THE DEVELOPMENT OF A TEXT GENERATION MODEL FOR SEPEDI LANGUAGE USING TRANSFORMER-BASED MACHINE-LEARNING TECHNIQUES

by

MAHLODI MERCY MOILA

DISSERTATION

Submitted in fulfilment of the requirements for the degree of

MASTER OF SCIENCE

In

DEPARTMENT OF COMPUTER SCIENCE

In the

FACULTY OF SCIENCE AND AGRICULTURE

(School of Mathematical and Computer Sciences)

at the

UNIVERSITY OF LIMPOPO

SUPERVISOR: MR MJD MANAMELA

CO-SUPERVISOR: DR TI MODIPA

2025

DEDICATION

This one is for me. I would like to thank myself for not giving up when it got tough.

DECLARATION

I declare that **THE DEVELOPMENT OF A TEXT GENERATION MODEL FOR SEPEDI LANGUAGE USING TRANSFORMER-BASED MACHINE LEARNING TECHNIQUES** is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references and that this work has not been submitted before for any other degree at any other institution.

Surname, Initials (Ms) **Date**

ACKNOWLEDGEMENTS

I would like to acknowledge the following people for their assistance and contribution in ensuring that I complete this study:

- Davis Sialumba, thank you for pushing me to further my studies to this level. If it were not for you, I would have found comfort in only having an undergraduate degree.

- I would like to thank Mrs Rene Kotze for ensuring that I did not quit when things got tough due to the lack of finances. Because of you, I was able to get NRF funding through the National Institute for Theoretic and Computational Sciences (NITheCS). If it was not for NITheCS, I would have dropped out.

- My Dad, who always encouraged me to give it my all and never give up when things got tough, thanks Pa.

- I also want to pass my sincere gratitude to my supervisor, Mr. MJD Manamela, and Co-Supervisor Dr. TI Modipa. It has been a bumpy road towards completing this research project, but you never gave up on me.

- To Rethabile, thank you for being the pillar of my strength during this journey.

- To my daughter Luna, in my life, you gave me so many reasons not to give up when things got tough. Your presence became my encouragement to succeed.

- To everyone who has supported me, I thank you for your time and encouragement.

ABSTRACT

The transformer-based machine learning technique is a deep learning model that processes the sequential input data using an encoder-decoder process. Transformers process the input data simultaneously using a parallelism approach while paying attention to each word at the time by applying an attention mechanism to each unit text being processed. The transformer-based model has been known to provide more state-of-the-art performance in natural language processing (NLP) tasks than a recurrent neural network (RNN) such as Long Short-Term Memory (LSTM). RNNs have the drawback of suffering from the problem of vanishing gradients and exploding gradients in implementation.

The GPT-Sepedi transformer-based model has shown great success in dealing with the process of text generation for the Sepedi language. This has led to a limited text generation system developed using a transformer-based model for the under-resourced African language, namely, the Sepedi language. This research project aimed to develop a text generation model for the Sepedi language using transformer-based machine learning techniques. The LSTM-Sepedi Attention-based model and the GPT-Sepedi Transformer-based model were developed and trained using a National Centre for Human Language Technology (NCHLT) Sepedi text corpus. The models were compared based on the results that they generated. A GPT-Sepedi Transformer-based model was used to generate the text.

The generated text was then compared with the Sepedi language vocabulary to determine the validity of the text. It was found that 61% of the text within the generated texts is found in the Sepedi language vocabulary. The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score was used to compare the model-generated text to human-written text. The ROUGE score result indicates that the GPT-Sepedi Transformer-based text generation model was able to generate words that humans can write with 83% precision. Even though the precision results indicated a better percentage, the text generated cannot be comprehensible with the

recall percentage of 0.05% and 0.1% F1-score results.
Keywords: Machine learning; Transformer; Sepedi; Text generation; GPT

TABLE OF CONTENTS

List of tables

List of Figures

1. CHAPTER 1: INTRODUCTION

1.1 Problem Statement

Natural language processing (NLP) is defined as the practice of creating machines that can manipulate human language or data that mimics human language (in written, spoken, and organised format) (Reshamwala et al., 2013). This process includes the development of different applications and models that can recognise, understand, interpret, and generate text or audio, such as in machine translation and spoken dialogue (Hirschberg & Manning, 2015). NLP is composed of natural language understanding, which involves the process of getting computers to understand written or spoken instructions provided by the end-user and natural language generation, which involves the process of generating the written or audio output after processing the input given by the end user in natural languages (Su et al., 2020).

Several studies have explored the text generation system's ability to generate coherent and semantically meaningful text. Text generation, a key task in NLP, aims to produce a readable and understandable text, as first highlighted by Mann (1982). This study focused on the ability of the system to generate coherent and semantically meaningful texts, thus text generation. This is done by using computational linguistics to produce texts that cannot be distinguished from human-written texts (Li & Jian-Tao, 2009). The process of text generation enables computing machines to generate meaningful text phrases and sentences for various purposes, including novel storytelling in a particular natural language (Hajdik et al., 2019). However, text generation models face several challenges, including syntactic structure, meeting human comprehension, readability, ambiguity challenges, and semantic connections, which can degrade a text generation model's performance (Stent et al., 2005).

Words can be identified in a text sequence using statistical analyses of text corpora. Furthermore, every word in a sentence depends on its context for meaningful and appropriate interpretation. Machine learning algorithms are often employed to predict appropriate words in text sequences (Agarwal & Arora, 2007). Machine learning-based systems, such as text generation message systems, have been developed for text generation in major languages of the world, such as the European Union languages. Not many automatic text-generating models are available for most under-resourced languages, such as the Sepedi language (Moila & Modipa, 2020). One of the NLP's strong achievements towards the realisation of the fourth industrial revolution implementation will, amongst others, be the development of automatic text generation models that support under-resourced languages the world over.

To develop a text generation model, lots and lots of curated text data is required. Various methods of developing text generation models have evolved (Welleck et al., 2019; Alsmadi et al., 2022). These authors discuss the different ways of developing models using neural network architecture. Deep Neural Networks (DNNs), Gated Neural Networks (GNNs), Gated Recurrent Units (GRU), and Long Short-Term Memory (LSTM) are stable techniques for dealing with sequential data (Islam et al., 2019). Furthermore, deep learning methods have been relatively successful in processes such as machine translation, dialogue response generation, summarisation, and other text generation tasks.

Despite their successes, these models have limited or low accuracy when analysing long sentences (Suman & Andreas, 2016). The DNNs, GRU, and LSTM methods use sequential processing of the text input instead of a parallel processing approach. These processing approaches tend to lead to exploding and vanishing gradients (Bin et al., 2021). The transformer-based models can process long-term dependency far more efficiently than the Recurrent Neural network (RNN) models (Radford et al., 2019). By focusing on the current word, transformers permit a better way of handling RNN issues than RNNs do by using an attention mechanism (Bin et al., 2021).

Transformer-based approaches provide cutting-edge results in NLP tasks by combining parallelism, attention, embedded space, and position encoding (Vaswani et al., 2017). The transformer-based model's ability to handle complex language structures and multilingual input and its state-of-the-art performance make it a promising candidate for automatically generating text in the Sepedi language. The Sepedi language is described as a tonal, agglutinative African indigenous language, which means that complex words with multiple morphemes, each carrying a specific meaning or grammatical function, are formed by adding prefixes, suffixes, and infixes to a root (Malope, 2006). The transformer-based model's ability to capture long-range dependencies and generate sequences of words is useful for modeling this complexity.

This study developed a text generation model for the Sepedi language using transformer-based machine learning techniques. The Sepedi language is an agglutinative African language with a distinct linguistic structure, which poses challenges for NLP researchers in developing language models that effectively accommodate its complexities. This has resulted in the language being underrepresented in the global NLP landscape. The development of open-source annotated corpora requires a significant amount of data, which is essential for building state-of-the-art models for underrepresented African languages like the Sepedi language. (Orife, et al., 2020). This technique was chosen because it can capture long-range dependencies and generate coherent and contextually appropriate text. Among other tasks, transformer-based approaches have proven to be highly effective in language translation, sentiment analysis, and text generation tasks. Using the power of self-attention mechanisms, transformer-based models are well suited to understanding relationships between words and capturing the global context of a text. They can handle complex grammatical structures and semantic nuances, which makes them ideal for developing a text generation model for the Sepedi language. In addition, they can easily incorporate large amounts of training data, allowing the model to learn from a variety of representative language patterns. To achieve accurate and fluent text generation in the Sepedi language, transformer-based machine learning methods hold great promise. Therefore, this study addresses the lack of computational tools for text generation for the under-resourced Sepedi languages.

1.2 Motivation

The text generation process involves creating coherent sentences using textual data, often through machine learning algorithms. It is, to an extent, quite challenging due to syntactic rules, semantic meaning, underlying intent, context, and resolving ambiguity accurately. Additionally, considering factors such as style, tone, and domain-specific knowledge adds to the difficulty. Hajdik et al. (2019) discuss how the semantic structure of a language can affect the context and content of text, which may cause a model to produce ungrammatical sentences.

The text generation process has been explored using various models, such as RNNs and LSTMs. These approaches have limitations in processing long input sentences (Xie, 2017). The issue with RNN models is that they cannot support the content and context of long sentences. The context of the previous words must be used to predict the next word in the generation. Dai et al.. (2019) discuss how LSTM can solve the long dependencies and the vanishing gradient problem in RNN by introducing a gating mechanism that selectively controls the flow of information through the network. This allows the network to remember relevant information over long sequences and avoid losing useful information through repeated multiplication of small gradients (Hochreiter & Schmidhuber, 1997). However, LSTMs, too, have limitations in that they can only effectively process sequences of a limited length. This is because the memory cells and gates in the network can become saturated or unstable when processing long sequences, leading to degraded performances. In practice, LSTMs have been found to perform well on sequences of up to a few hundred words, but beyond that, their performance tends to deteriorate (Dai et al., 2019).

The transformer-based models have been introduced, which use an attention mechanism to process input data all at once, resulting in better performances in NLP tasks (Vaswani et al., 2017). In other words, all input data is processed at once, and each word is analysed individually. Different transformer-based models are used in different NLP tasks, including text generation tasks such as novels, story writing, customer service, and much more. These models produce state-of-the-art performances based on the settings of their parameters (Topal et al., 2021). Generative Pre-trained Transformer 3 (GPT-3) produces a text that is readable without fine-tuning, whereas Bidirectional Encoder Representations from Transformers (BERT) use bidirectional and fine-tuning to produce fine results (Xia et al., 2020; Dehouche, 2021).

Transformer-based models have a better way of dealing with problematic RNN issues because they use an attention mechanism to eliminate the sequential processing of a text that RNN performs by focusing on the current word (Bin et al., 2021). The important issue in text generation is to have a model that will use the given input text dataset and produce an output text that humans can understand (Vaswani et al., 2017). The great need for such a model is that it will generate syntactically correct and meaningful Sepedi language sentences, which is essential since there are limited studies on automated text generation for the Sepedi language.

In recent years, there has been growing interest in developing NLP models for under-resourced languages such as Sepedi (Schneider et al., 2021). One approach that has shown promise in successful model development is using deep learning models, such as GPTs and LSTM, stacked with the attention mechanism for text generation tasks. By using these models, a high-quality text can be generated that mimics the language patterns and structures of humans (Cheng et al., 2016; Radford et al., 2019).

By developing a Sepedi language text generation model using these deep learning models, we can provide a valuable resource for Sepedi language speakers. The model can be used to generate text for a range of applications, such as creating educational materials, generating news articles, and supporting online communication (Hedderich et al., 2021). Moreover, this project can serve as a starting point for further research and development of NLP models for similar under-resourced languages, contributing to the broader goal of digital inclusion and language preservation (Kaddari et al., 2021)

In summary, the development of a Sepedi text generation model using GPT and LSTM stacked with the attention mechanism can provide a valuable resource for the Sepedi language speakers, and also help bridge the digital divide between well-resourced and under-resourced languages by developing useful systems and contribute to the broader goal of language preservation and digital inclusion.

1.3 Aim

The study aimed to develop a text generation model for the Sepedi language using transformer-based machine learning techniques.

1.4 Objectives

The objectives of this study were to:

i. Perform data exploration on the National Centre for Human Language Technology (NCHLT) Sepedi dataset.
ii. Train the LSTM-Sepedi Attention text generation model.
iii. Train a Generative Pretrained Transformer (GPT)-Sepedi Transformer text generation model.
iv. Evaluate the performance of the Long Short-Term Memory (LSTM)-Sepedi Attention text generation model and (GPT)-Sepedi Transformer

text generation model for the Sepedi language.

v. Perform text-generated evaluation using the ROGUE score.

1.5 Scientific Contribution

This study contributes to the development of a text generation model for the Sepedi language to support NLP applications that require large text corpora. This model will aid in assessing the suitability of transformer-based language models for African indigenous languages such as the Sepedi language. Text generation models, such as the GPT-n series, can be used to generate novel written content in African indigenous languages, such as the Sepedi language, one of the local under-resourced languages.

To generate coherent text, these models can be trained on a large corpus of text data, including books, articles, and websites. Using text generation models in the Sepedi language can significantly enhance the availability and quality of Sepedi language resources and contribute to the ongoing promotion and preservation of African languages. To utilise text generation models effectively, high-quality training data and computing resources to train the models are needed.

The development of a text generation model for Sepedi has important implications for several areas, including education, communication, and cultural preservation. With a text generation model, it would be possible to automatically generate large volumes of domain-specific text in Sepedi, which could be used to develop educational materials, improve communication in the language, and promote its use in various domains. Overall, the use of GPT-2 techniques has the potential to significantly enhance the availability and quality of Sepedi language resources and contribute to the ongoing efforts to promote and preserve important African languages.

1.6 Ethical Clearance

The study did not require ethical clearance as it did not involve the use of humans or animals in the collection of data. The study used a secondary dataset named NCHLT Sepedi text corpus (Puttkammer & Martin, 2014) to develop a transformer-based model for generating the Sepedi text.

1.7 Dissertation Structure

This dissertation is structured into five chapters, each of which serves a distinct purpose and contributes to the overall research objectives.

- Chapter 2 is the background and literature review that explores relevant theories, themes, and gaps in the existing literature related to the research topic and the Sepedi language.
- Chapter 3 presents the research methodology, including data preprocessing, research design and approach, and analysis procedures.
- Chapter 4 is dedicated to presenting results and discussions of research, including a description of the research findings and interpretation of results.
- Chapter 5 presents the conclusion of the research, summarising the main findings and their implications, discussing the contributions of research to the field, outlining the study's limitations, and providing recommendations for future research.

2. CHAPTER 2: BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

Language plays a critical role in communication and identity formation, and for many communities, their native language is an essential part of the survival of their culture. (Skutnabb, 2000). Unfortunately, some African Indigenous languages in South Africa, such as the Sepedi language, are considered under-resourced, meaning that they have limited linguistic resources as far as corpora, computational tools, and linguistic expertise for language-related tasks are concerned. (Mahlangu, 2018). This scarcity of language-related resources can make it extremely challenging to develop useful NLP applications for under-resourced languages.

One of the critical tasks in NLP is text generation. It involves a collection of texts in a specific language for training machine learning models to generate coherent texts in that language. The RNNs, such as the LSTM model, have been widely used for text generation tasks (Dhall et al., 2020). However, recent technological advances in transformer techniques, including the popular generative pre-trained transformer (GPT) series, have shown promising results in text generation tasks.

In this chapter, we discuss the Sepedi language and provide the background to text generation, including the nuances of applying various NLP models and techniques, such as RNNs, LSTMs, and GPT-2, for under-resourced languages such as Sepedi. We shall also explore the benefits and limitations of each method and provide examples of how one of the GPT models can be used to generate a high-quality text in the Sepedi language.

2.2 Sepedi Language

The Sepedi language, also called Sesotho sa Leboa, is one of the 12 official languages of South Africa. It is predominantly spoken in northern South Africa, including in the provinces of Limpopo, Mpumalanga, and Gauteng. (Molepo, 2014). Approximately 10% of South Africa's population (roughly 6.57 million) speak the Sepedi language, making it the fourth most spoken language.[1]. However, under-resourced languages, such as the Sepedi language, have not and continue to not receive the same priority as English in South Africa in terms of requisite development and intended official usage, resulting in limited digital resource development for most of these indigenous languages. (Turner & Wildsmith-Cromarty, 2014). Therefore, African indigenous languages are continuously having to contend with a limited number of resources required for their digital development. (Mkhize & Balfour, 2017).

South Africa still faces significant challenges due to the perceived disparity between segregationist and assimilationist ideologies, as students are in most cases taught in English instead of their mother tongue languages at most institutions of learning. (Sidik, 2019). Furthermore, despite being spoken by a significant number of people in South Africa, there is a serious lack of electronic digital resources and tools available in the Sepedi language, such as NLP models (Marivate et al., 2020). The lack of adequate digital resources will always lead to suboptimal performance when applied to the Sepedi language. As a result, the speakers of this language often do not have access to the same level of electronic digital services and information as the speakers of the more well-resourced languages.

Regarding language structure, there are syntactic differences between the languages belonging to the Nguni and Sesotho classes of Bantu Languages. In the Nguni class, a conjunctive system of representation is employed, where words are combined to create a lemma. In contrast, the Sesotho class, and to some degree the Tshivenḓa and Xitsonga languages, adopt a disjunctive system, where words

[1] These are the most spoken languages in South Africa – BusinessTech

are divided into distinct lemmas (Van Wyk, 1995). Most of the Bantu languages, with an agglutinating morphological structure, use several prefixes and suffixes built around a root form, such as a noun stem or verb root (Baerman et al., 2017). For example, the expression "I am fine" (English) in Sepedi, a Bantu language, consists of two orthographic words "*Ke gona*", whereas in isiZulu, a Nguni language, it is represented by a single orthographic term "*Ngikhona*" (Taljard & Bosch, 2006). Furthermore, the rules of derivation vary among languages, meaning that not all verbal and noun extensions can be combined with every verb root (Allwood et al., 2010; Grover et al., 2010).

When dealing with NLP tasks, it is crucial to consider the morphology and orthographic structure of agglutinating languages. The agglutinating languages are synthetic and are characterised by a high number of morphemes (Mngomezulu et al., 2019). Therefore, even when words are combined, the meaning of each morpheme remains the same, making it important and necessary to analyse the morphology and structure of each morpheme to accurately capture the intended meaning of the word. However, there are various representations of orthographic language units in individual linguistics, such as phonology, semantic properties, or statistical properties, which can pose a challenge in computational morphology (Pretorius et al., 2009; Kosch, 2006). In the Sepedi language, words with the same spelling can have different meanings as they have different graphical representations, and a single word can have multiple meanings, resulting in heteronyms and polysemous function words (Mothapo et al., 2021). For instance, the word "*bona*" in the Sepedi language can mean either "to see" or "to refer to a group of people" (Faass et al., 2009). The manner of articulation or pronunciation of this word and its context determines the correct or intended meaning whenever it is used.

The number of people who speak a language in their geographic location directly impacts an individual's ability to communicate in that language. Thus, the lack of resources for certain languages in NLP reflects the larger societal issue of resource scarcity. While there is a lack of diversity in NLP research, the question of scaling NLP to low-resource languages remains unresolved, as low-resource language problems are complex and systemic. To obtain basic corpus statistics, morphology or syntactic parsing tools are not necessary. Word frequency tables can provide insight into spoken language by showing the frequency of words, as opposed to written language (Allwood et al., 2010).

Under-resourced African indigenous languages face challenges in NLP tasks due to limited numbers of small datasets, with approximately 2000 languages spoken on the African continent. As a result, artificial intelligence techniques are not commonly and easily used for these languages, leading to further technological disparities (Cunliffe et al., 2022; Shikali & Mokhosi, 2020). Resources for language technology implementation are lacking, leading to concerns that many African indigenous languages tend to fall behind within the broad language engineering for the emerging fourth industrial revolution.

2.3 Text Generation Techniques

This section provides an overview of a text generation process, including its definition, the background of various techniques used for text generation, and their applications in developing different text generation tools.

2.3.1 Definition of Text Generation

In the field of NLP, text generation involves leveraging computational linguistics to produce a text that appears indistinguishable from a human-written text. Computational techniques are widely used to generate a text that is coherent and linguistically appropriate and that closely mimics a human-written text in terms of style and content. (Xie, 2017). Most NLP tasks rely on pre-trained models to overcome the problem of data scarcity faced by under-resourced languages. Rather than creating models and addressing challenges, the creation of datasets is found to be more urgent and useful in addressing data scarcity problems (Adelani et al., 2021). The availability of language corpora is crucial for African indigenous and/or other/minority languages to keep up with technological advances resulting from global language competitions. However, pre-trained models require additional data and high-performance computing systems to operate smoothly and effectively (Allwood et al., 2010).

In a recent survey paper (Iqbal & Qureshi, 2022), RNNs, Convolutional Neural Networks (CNNs), and Generative Adversarial Networks (GANs) are discussed as deep learning models for text generation. Artificial intelligence techniques enable computers to learn how to communicate like humans, giving them the ability to process various types of information, such as images, words, speech, and other forms of data. According to Woolley (1969) and McKeown (Kathleen), an application such as language translation uses both text and audio data as input types for categorising text. To ensure readability, it is important to consider the syntactic structure of the input data when generating text. Semantic structure affects not only language content but also multiple linguistic contexts. According to Hajdik et al. (2019), this may cause the model to produce sentences that are not grammatically correct if not attended to carefully.

Understanding language syntax, i.e., how words are grouped to form sentences, is crucial in the process of generating a text from a trained language model. It is important to establish a semantic link between sentences to ensure coherence and understanding in the resulting text. Therefore, when creating a system that generates a text using computational tools, it is also necessary to develop the system comprehensively from the start to determine the content and desired aesthetic form of the text (Iqbal & Qureshi, 2022)

Deep learning systems can efficiently handle vast amounts of data and computations at a remarkable speed while simultaneously processing them by utilising more resources and deep learning models, including RNNs, CNNs, variational AutoEncoders (VAEs), and GANs, have proven to be effective solutions for NLP problems (Fatima et al., 2022). The challenge of generating natural language in text generation research cannot be denied. The current limitation of deep learning models is their inability to capture all the intricate details, technicalities, and interpretations of natural language. Because of these details, longer texts are created in diverse circumstances. The RNNs, general artificial neural networks, and transformers are some examples of conditional text generation techniques that have been proposed for generating texts. (Bin, et al., 2021)

The utilisation of GANs through a reinforcement learning approach enables the creation of logically structured and contextually relevant text. This approach has shown considerable progress in enhancing the quality of text generation (Subramanian et al., 2017). Moreover, a larger number of sentences generated through this method leads to even better outcomes in terms of precision and readability. When working with text data, gradients cannot be easily propagated. This is due to the nature in which the NN processes sequential text data by not being able to learn the previous data. As a result, a previous GAN solution faced challenges related to producing content lacking meaningful information and encountering infrequent scalar reward signals while generating lengthy textual outputs. To address this issue, Guo et al. (2018) proposed a method for utilising adversarial training with LeakGAN to generate longer texts. By using the discriminator feature as a guiding signal, the generator is more effectively guided when generating long texts. However, this method encounters issues if the sample text exceeds 200 words.

Before developing a coherent text generation system that can generate meaningful and understandable texts, various factors need to be considered. The words in a sentence must be syntactically correct and make sense independently before they can be understood within a sentence. To generate coherent sentences, it is beneficial to consider the preceding and succeeding sentences. To achieve a coherent text, the previous and following sentences must be semantically related in some way. Without this step, the preceding discourse may influence the generated text as it will make it difficult for readers to comprehend the intended meaning and context of the generated text due to poor quality, coherence, and understandability. The initial objective of any text generation model is to organise words and phrases in the most appropriate order for each sentence.

During the past few years, interest in text generation in Bantu languages has grown, with researchers exploring various methods to generate text in these languages. One of the primary challenges in generating text in Bantu languages is the fact that they are highly inflected with complex grammar and syntax (Byamugisha et al., 2017). Some of the studies conducted on the Bantu languages include, but are not limited to, the study conducted by Laura and Jade (Martinus & Abbott, 2019), which focuses on translating English language into Bantu, Nguni, and Afrikaans languages spoken in South Africa. Bantu languages have a complex noun class and often use angulations, where words are formed by combining their morphemes. Nguni languages have a distinct use of click sounds and have unique phonological and grammatical structures. The researchers used a neural machine translation technique to create a model that could generate coherent and grammatically correct sentences in the language.

Although further research in this direction is needed, studies provide evidence that advanced machine learning methods can be used to better generate text in Bantu languages. This advancement may significantly impact the growth of NLP technologies, which can leverage it to facilitate communication in these languages spoken by millions of people throughout the African continent. In a study conducted by researchers at the University of Cape Town, the potential of a transformer-based language model called XLM-R for text generation (Conneau et al., 2020) found that the model could generate coherent and meaningful sentences in Swahili, despite the language's complex grammar and syntax. The researchers trained multilingual models using parallel data, which showed the effectiveness of the model performance across different cross-lingual tasks with limited linguistic resources. The research presented a significant contribution to the field of NLP, particularly in the context of low-resource languages.

2.3.2 Deep neural network techniques

RNNs are neural networks that process sequential data, such as time series data, natural language texts, or audio signals. These networks maintain a "memory" of previous inputs, allowing them to process each input in the context of the previous ones. The RNNs operate by taking in one input at a time, processing it, and passing the output back into the network to process the next input. In this way, the network can "memorise" the input sequence and use that information to generate predictions or outputs (Lipton et al., 2015). Fig 2.1 Illustrates the basic structure of an RNN, which consists of one recurrent neural whose output feeds back into its input at each time step. This recurrent connection allows the neuron to maintain a memory of its previous state, which can be used to inform its processing of subsequent inputs. In practice, RNNs are typically implemented as a series of these recurrent neurons, each processing one input in the sequence. The outputs of the neurons are fed forward to the next neuron in the sequence and back to the input of the current neuron at the next time step (Goodfellow et al., 2016).

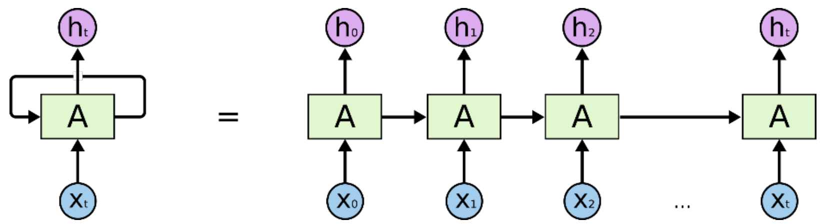


Figure 2.1 Recurrent Neural Network architecture[2]

According to a recent review of deep learning in this field, sentence representation is a crucial task for many NLP applications (Young et al., 2018). RNNs encode a sentence into a fixed-length feature vector, which can be used for multiple downstream tasks, such as machine translation or sentiment analysis. Bahdanau et al. (2015) discuss that the study uses an encoder-decoder structure with RNNs as the basic building block for both the encoder and the decoder. The process involves the use of a bidirectional RNN to encode the source sentence into a sequence of vectors and a unidirectional RNN to generate the translated sentence from the encoded vector representation. The RNN has been identified as one of the most effective algorithms for modeling sequential data, particularly in the field of NLP. Moreover, since RNNs have some kind of internal memory, they can remember both past and present inputs, which simplifies sequence modeling. (Sherstinsky, 2020). The sequential structure of RNNs makes them suitable for analysing text data in general. As a result, natural language text can be viewed as some kind of sequential data containing specific relationships between contexts.

Research has shown that RNNs outperform some language models, such as N-gram models, in determining sentence structure accuracy (Doval & Gómez-Rodríguez, 2019). Additionally, statistical language models that use backpropagation can maintain context and capture the structural properties of language. As a result of the recurring memory of the RNN, it can retain some details of the context, which assists in the capture of the inherent linguistic structure when training models based on RNNs for text generation (Bin et al., 2021). This can lead to difficulties in capturing

[2] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

long-term dependencies, which are particularly challenging for neural networks. As a result, the algorithm may assign high weights to certain connections, which can result in two types of gradients: the exploding gradient, which can prevent the model from learning, and the vanishing gradient, which can prevent it from learning when gradients are too small (Salehinejad et al., 2017).

One of the problems with RNNs is that long-range dependencies within a sequence or across multiple sequences can be lost. For instance, if a word at the beginning of a sequence is important for a word at the end of the sequence, the model may forget the first word by the time it reaches the last word (Jozefowicz et al., 2015). To overcome this limitation, the RNN model can be designed in such a way that it focuses on certain parts of input data while disregarding others to improve their performance. This approach can help mitigate the challenges posed by longer sequences in RNNs.

To address this limitation, attention mechanisms can be added on top of RNN architecture, according to Bahdanau et al. (2015). Attention mechanisms are a fundamental concept in the field of machine learning, particularly in the domain of neural networks and NLP. In general, attention allows the network to focus on different parts of the input sequence at different times, updating its memory when needed. By integrating attention mechanisms, RNNs exhibit an enhanced capability to process extended input sequences, thereby facilitating a more adept capture of the interrelations among distinct tokens within the sequence (Cho et al., 2014). To solve a problem more efficiently, the model is trained to focus on certain parts of the input data while leaving out the less important parts.

A neural network reads a text from left to right and updates its state after processing each word. However, by the time a sentence is finished, much of the information from earlier words may be lost (Gupta & Gupta, 2019). This is partly because of the disappearing and exploding gradients occurrence, which can hinder RNNs' ability to detect long-term dependencies. To solve these issues, LSTMs were created.

Long-Short-Term Memory: The LSTM technique is an artificial neural network type that increases language modeling accuracy. The research by Klaus et al. (2017) has highlighted the importance of memory cells, input, output, and forget gates within the LSTM architecture. These LSTM components enable the network to selectively retain or forget aspects of context, allowing the model to make informed decisions about which information to keep and which to discard. As a result, the LSTM can more accurately predict the next word in a generated text, making it a valuable tool for NLP tasks. (Santhanam, 2020). Hong et al. (2018) demonstrated the effectiveness of LSTMs in language modeling and the importance of its components in achieving high performance. As a result, the LSTM can decide with greater knowledge what data to retain and anticipate the following word in the generated text with greater precision.

Barman and Boruah (2018) used RNNs with LSTM cells to predict the next word in a sentence because on the previous words. Specifically, the model aimed to predict the next word in the Assamese language, a language used for e-communication, to improve time management. The results of the model were impressive. However, the Assamese language posed a challenge due to its numerous synonyms for common words such as ‘you’. To address this, the researchers utilised LSTM and IPA.[3] Charts.

Moila and Modipa (2020) employed the LSTM method to create a text generation model for the Sepedi language. The model was trained using the NCHLT Sepedi dataset. During the study, the overall performance of the model achieved 50% accuracy. It was noted that the arrangement of certain generated words in the text led to challenges in understanding them due to their sentence structure.

Figure 2.2 Shows the structure of the LSTM model for the text generation process. By providing state nodes to the RNN’s process, the LSTMs are designed to address

[3] https://www.internationalphoneticalphabet.org/

the problems caused by memory constraints, temporal dependency, and gradient propagation. As the RNN processes the input, it also receives state information. The next stage is to get a new input and some state information from the LSTM before the output. State refers to input gates, forget gates, and output gates. In the flow of the sequence, forget gates determine whether state information stored in the processing state can be forgotten, input gates determine which state information needs to be updated into the processing state, and the output gate decides that all information should be removed. From the processing state, it is saved, and part of it is printed at time *t*. All states use sigmoidal activation and *tanh* functions. The sigmoidal activation function provides smooth processing while the model remains differentiable, and the *tanh* function spreads and warms the gradients, preventing them from vanishing/exploding and letting the cell state flow for a while.

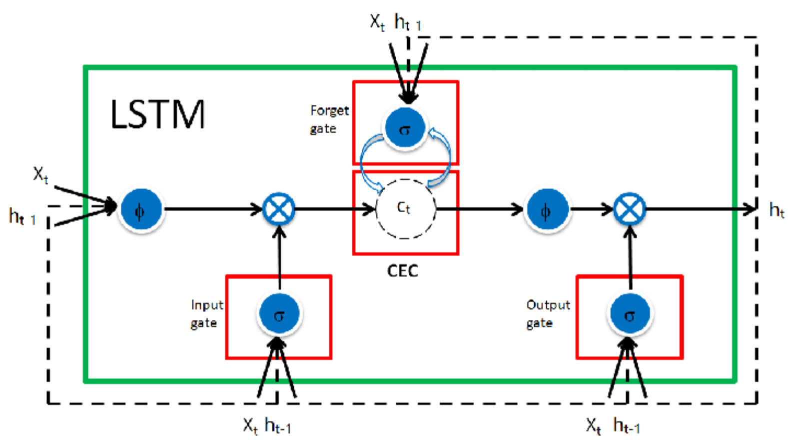


Figure 2.2 Long-Short-Term Memory Architecture (Su & Kuo, 2019).

It is unequivocally evident that LSTM models perform better than n-grams in high-resource environments, but it remains unclear whether the adoption of LSTM architectures will yield similar improvements in the specific context of under-resourced South African languages (Tüske et al., 2018). In our study, we aimed to evaluate LSTM stacked with attention on the Sepedi language. LSTM stacked with attention is the integration of multiple layers of LSTM to the attention mechanism that can capture multiple and complex patterns in sequential data. This process is set to improve the performance of the text generation model by providing the model with the ability to capture long dependencies while focusing on the relevant information within the input sequence. The LSTM model's performance is improved by providing the model with the hierarchical representation of the input sequence, having an improved information flow and memory, and helping the model by reducing the information overload by focusing on only relevant information to avoid information overload.

2.3.3 Transformer Technique

The transformer technique is a deep learning model that processes the input data simultaneously using the encoder-decoder technique to process text input based on a parallelism approach using transformer models (Vaswani et al., 2017). This means that all input data is processed simultaneously. NLP tasks such as generating text for novels or customer service frequently use transformer-based models (Topal et al., 2021). Compared to sequential neural networks, transformers are more effective in addressing issues with RNNs because they use an attention mechanism instead of sequential processing (Bin et al., 2021). Transformer models, which are based on artificial neural networks, have recently achieved state-of-the-art results in language modeling (Radford et al., 2019).

Transformer models overcome the varnishing and exploding gradient problem by using self-attention to memorise the entire sequence. As a result of the Vaswani et al. (2017) paper, sequence-to-sequence models have been improved for a wide range of tasks in NLP. The introduction of the transformer architecture and self-attention mechanism have become standard tools in the NLP toolkits and have paved the way for further research in the field. The transformer is an essential tool as it overcomes the challenges of parallelisation and loss of contextual information encountered in earlier sequence-to-sequence models like RNNs and LSTMs.

The transformer model differs from other models in that it exclusively employs self-attention to derive the representation of a sequence of words or sentences by linking various words within the same sequence. Transformers adopt an encoder-decoder architecture with two primary constituents. While the encoded uniform shape is being decoded, a sequence of outputs is being generated by the decoder. Encoders and decoders can process the entire sequence of embedded words simultaneously, which reduces the time needed for training the model. Furthermore, this parallel processing allows for context to be computed from all words, creating a more comprehensive context. With transformer-based models, various tasks can be accomplished effectively in the field of NLP, such as Question Answering, reading, and using NLP to communicate with customers.

Fig 2.3 Shows the process of transformer architecture (Vaswani et al., 2017). Autoregressive models, including transformers, use all previous inputs in a sequence to forecast the subsequent output. However, unlike traditional sequence-to-sequence schemes such as RNNs, which process sequences one element at a time and maintain an internal hidden state that encodes information about the entire history of the sequence, transformers process the entire input sequence at once (Dai et al., 2019).

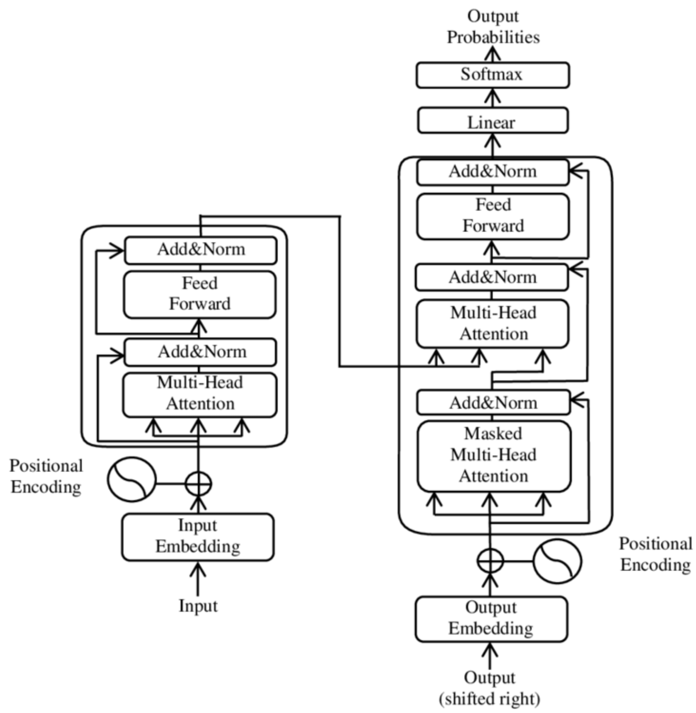


Figure 2.3 Transformer Architecture (Vaswani, et al., 2017)

For processing sequences, transformer models use the encoder and decoder components with stacked attention mechanisms. This approach, which employs encoder-decoder networks with attention, has demonstrated its efficacy in addressing various sequence-to-sequence challenges. Specifically, the encoder converts the input sequence into a one-dimensional hidden vector (Cho et al., 2014). For a decoder to produce the corresponding output sequence, the hidden vector is converted into symbols or tokens, which represent words for tasks such as machine translation or generation of text. Encoders and decoders do this by stacking self-attention layers and connecting them in a pointwise manner. The input for the encoder is a list of vectors, which is processed through a self-attention layer and feed-forward neural network before being passed on to the next encoder.

The input in the transformer model is first passed through a word embedding layer as the initial step. The input dataset is tokenised, resulting in a sequence of tokens with a fixed length. These tokens are generally represented as integer indices within a vocabulary dataset. To provide the model with information about word positions without relying on recurrent processing, positional embedding is added to each embedding in the bottommost encoder. This embedding process is limited to the bottom-most encoder. The resulting text is then represented as a vector in a space where words that are close to each other are grouped (Sonkar et al., 2020)

To better comprehend the significance of a word in diverse sentences, the subsequent move is to insert positional information into the embeddings. The positional embedding in the Encoder stage involves converting a word into a vector representation and understanding the spacing between words in a sentence, transformers use position embeddings. Thus, a vector is produced that includes the positional relationship between the words in the sentences (Dar et al., 2023). In transformers, embedding algorithms are used to represent each word as a vector, resulting in a token. It is crucial to maintain the position of each word within a sentence to enable the transformers to add a vector to each input embedding that adheres to a pattern learned by the model, aiding in determining the position and distance of each word in the sentence. This process facilitates the model's ability to extract the meaningful distance between the embedding vectors after they are projected to the attention stage.

As soon as the words of the input sentence are embedded, each vector is processed through two layers of encoders, multi-headed attention, and then a fully connected network. A residual connection is also implemented around each of the two sublayers, followed by layer normalisation. The transformers have a crucial feature, parallelism, to enhance the processing speed, unlike RNNs that process the data sequentially, consuming more time. This means that each word at every position has its path in the encoder. Unlike in RNNs, where the data flows sequentially, consuming more time. Therefore, transformers are much faster due to their parallelisation property (Sutskever et al., 2014). Using stacked self-attention and pointwise, fully connected layers, transformers employ a process for both the encoder and decoder to be able to complete the process of text generation.

During the encoding process, vectors are input into an encoder, and then a feed-forward neural network is applied to the vectors. In turn, the output is passed on to the next encoder. During the encoder's processing, each item in the input sequence is compiled into a context vector. Encoders transfer context vectors to decoders once they have processed the entire input sequence. The process of the decoder structure operates by generating the output sequence one item at a time. The word embedding algorithm transforms words into vector spaces that capture many semantic and contextual meanings (Cho et al., 2014).

- Generative Pre-trained Transformer

GPT models, as reported by Topal et al. (2021) perform at a state-of-the-art level. As already mentioned, the benefits of the transformers are that they eliminate the sequential processing of a text that RNNs perform by focusing on the current word, which is the process that transformers use (Bin et al., 2021). Having a model that utilises input datasets to produce output texts that humans can understand is critical in text generation (Vaswani et al., 2017).

Fig 2.4 Illustrates the differences between the GPT series of language models. Deep learning language models such as GPT-1 were developed using a transformer architecture to generate human-like text and were first introduced in 2018, followed by GPT-2 in 2019 and GPT-3 in 2020 (Brown et al., 2020). Unsupervised learning techniques allow GPT models to understand patterns and structures in human language from large amounts of text data captured from many sources, including books and articles. There are different parameters such as weights, biases of weights, embeddings, and encoders GPTs language model in GPT-1 total 117 million; those in GPT-2 total 1.5 billion; and those in GPT-3 total 175 billion (Radford et al., 2018; Radford et al., 2019).

This pre-training has resulted in the GPT models becoming highly capable language generators, which can be used in a variety of applications. These include language translation, chatbots, content creation, summarisation, question answering, and sentiment analysis, among others (Zhang et al., 2020). Each application utilises the language generation capabilities of GPT models in different ways, depending on the specific use case.

In summary, the GPT series of language models have evolved, with increasing numbers of parameters and improved language generation capabilities. In various applications, these models have been trained on large amounts of text data using unsupervised learning techniques.

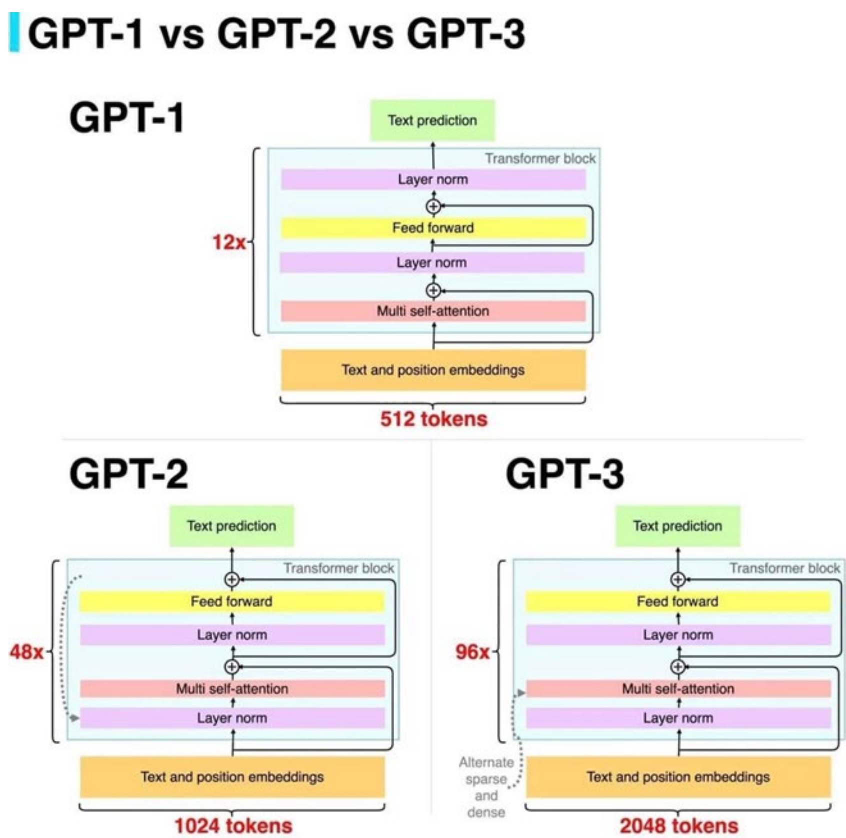


Figure 2.4 GPT 1, 2, 3 Architecture (Damien , 2023)

Due to the limited number of studies on automated text generation for the Sepedi language, a model that can generate meaningful sentences in this regard is essential. Recently, BERT and GPT have been proposed. BERT is a stack of encoders, while GPT is a stack of decoders based on transformer architecture. BERT was introduced by Google AI Language (Devlin et al., 2019), while GPTs were introduced by OpenAI (Radford & Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020).
It has been found that pre-trained transformer language models like GPTs (1,2,3) and BERT provide better results when generating text (Schramowski et al., 2021). As a result of these models, it was impossible to distinguish between machine-generated and human-generated text (Xia, 2020). As far as transformer architecture is concerned, GPT-1, GPT-2, and GPT-3 are very similar and differ mostly in the size of the training data and the number of transformer blocks about the incoming tokens.

Several studies have demonstrated that fine-tuned pre-trained language models, such as GPTs, can generate texts that are comprehensible and readable when they have been trained correctly, as reported by Mager et al. (2020). GPT-2 is a 1.5B parameter transformer that provides state-of-the-art performance. As a result, the model produces not only readable but also correct text. The GPT-2 language model uses a decoder transformer architecture to generate text sequences by predicting the next word based on the previous ones. GPT-2 does not require an initial representation of the input from an encoder. Due to this design, GPT-2 is well-suited for tasks that require generating output sequences purely based on input sequences, such as language modeling, text generation, and question-answering. In contrast, encoder-decoder transformers typically use both an encoder and a decoder to generate output sequences from input sequences (Dai et al., 2019)

Du et al. (2021) used traditional RNN models and the pre-trained language GTP-2 to assist communities with adapting to online systems by generating text for essays and summaries, among other tasks. The GPT-2 report indicates that the community was able to read the text generated by the language model. The GPT-2 model relies on a single transformer decoder block. As a result of unidirectional self-attention masking, each token can only attend to previous tokens (Li et al., 2021; Brown et al., 2020). It has been shown that large language models can perform well when trained on sufficiently large and diverse datasets (Radford et al., 2019). A high-capacity model trained to maximize the likelihood of a sufficiently varied text corpus begins to learn how to perform a surprising number of tasks without explicit supervision in a zero-shot setting.

Having systems that can give meaningful responses is important when creating dialogue systems (Budzianowski & Vulić, 2019). This practice will allow humans to engage in the system the same way they engage with each other. The process of training in this kind of system requires a large amount of data. The questions raised about GPT (1 & 2) base models revolved around their ability to effectively support a task-oriented dialogue application. When the model was fine-tuned, it achieved a success rate of only 50% in generating appropriate dialogue responses.

- BERT Language Model

The BERT language model is designed to pre-train and fine-tune deep bidirectional representations from the unlabeled text by conditioning both the left and right context in all layers. It focuses on the unsupervised tasks of the Masked Language Model (MLM) and Next Sentence Prediction (NSP), which BERT uses in the pre-training process. The masked language model (MLM) masks some tokens from the input at random (Toutanova, 2019). Predicting the masked word's original vocabulary is based solely on its meaning. In contrast to pre-training a left-to-right language model, the MLM objective allows the representation to fuse the left and right meaning, allowing us to pre-train a deep bidirectional transformer. The next sentence prediction pre-trains the text-pair concurrently.

2.4 Conclusion

The literature review indicates that very little research has been conducted on text generation for under-resourced languages using pre-trained language models. To conclude, the Sepedi speed language is one of the under-resourced languages that lack most NLP applications. It is necessary to develop technologies such as text generation that can be used to create new content in the Sepedi language. RNNs and LSTM networks are well-known deep-learning models that are used in text generation. However, these models are limited in generating coherent and contextual text.

The Encoder-decoder models are currently dominating the research field. These models include pre-trained models such as GPTs, GNAs, and BERT. The GPT series of language models generates texts that are human-like using a transformer architecture, a state-of-the-art deep learning model. The model requires a high volume of data, allowing it to learn the relationship between the text and then generate valid words. Some of the benefits of the transformer model include but are not limited to summarisation, classification, content creation, and novel. With its high-quality language generation capabilities, transformers can be used to support the development of the Sepedi language by generating new content in the language. Transformer techniques have opened new possibilities for the development of under-resourced African languages.

Table 2.1 Indicate the review of the existing literature where the performance of the models was measured using Loss for the developed text generation models for under-resourced languages. These studies highlight significant challenges faced when training deep learning models, particularly transformer-based models.

Table 2.1 Review of the existing literature

Study	Model Type	Loss Result	Data Size Requirements	Key Findings
Moila & Modipa (2020)	LSTM	1.78	Requires smaller dataset	LSTM models perform better than transformer-based models in text generation for under-resourced languages.
Ramalepe et al. (2025)	Transformer-based Transformer-based	2.78 3.46	Requires a larger dataset	Transformer models need larger datasets to perform effectively in text generation tasks.

The findings show that models using LSTM architectures perform better compared to transformer-based models, as evidenced by a lower loss result of 1.78 for the LSTM model compared to loss values of 2.78 and 3.46 for the transformer-based models. This difference is attributed to the data size requirements, as training transformer models requires a larger dataset than training LSTM models (Moila & Modipa, 2020; Ramalepe et al., 2025).

3. CHAPTER 3: METHODOLOGY

3.1 Introduction

The methodology for this study involved and started by using a dataset of Sepedi language text. The NCHLT Sepedi text dataset (Eiselen & Puttkammer, 2014), is used as the secondary data source. It is a collection of texts in Sepedi that has been curated and annotated for language processing tasks. The NCHLT is a project that was initiated to develop speech and technologies for South African official languages (11 Languages) (Badenhorst & De Wet, 2022). The Human Language Technology Index is useful in making comparisons between the HLT development for each of the then 11 official languages of South Africa (Grover et al., 2010). The NCHLT Sepedi dataset is a text corpus of the Sepedi language, which is a member of the Bantu language family spoken in South Africa. The limited size of the Sepedi dataset can pose a challenge in the training of machine learning models, particularly those that necessitate a substantial amount of data, such as deep neural networks. (Cunliffe et al., 2022; Bansal et al., 2022).

The next step was the development and training of a Sepedi text generation model using LSTM stacked with attention and GPT-2 techniques. For the evaluation of the performance of the Sepedi text generation models trained using LSTM stacked with attention and GPT, various metrics, such as loss and accuracy, were used to compare their performances on the Sepedi language dataset. Additionally, the quality of the output text was measured by the ROUGE score to determine the effectiveness in generating coherent, syntactically correct language.

3.2 Sepedi Text Data

In general, the availability of any form of data items allows one a chance to process these items towards making an informed decision either for research or business purposes. Different analysis methods, text analysis, and data mining are used daily to analyse the data to help with decision-making. (Ashirwadam, 2014). Text analysis is used to find the inside of the text dataset, such as NLP, and white data is used to discover the relationship and patterns within a larger dataset that can be used in tasks such as regression and predictive modelling. For this study, text analysis was used to analyze the text dataset. The data can be represented in either structured, semi-structured, or unstructured forms (Hänig et al., 2010).

The dataset is one of the most essential tools in NLP. To train a language model, an appropriate dataset is required. The more language and speech technologies evolve through focused development initiatives, the less the fundamental digital resources for African languages suffer from the data scarcity phenomenon. Most African indigenous languages are found to be highly under-resourced. The dataset includes a range of text genres, such as news articles, social media posts, and literature, providing a diverse sample of the language. Before using the dataset for model development, the process of preprocessing is required to clean and prepare the data for the further processing stage. To perform these preprocessing tasks, removal of data anomalies, removal of special characters, and any non-text value is important when developing a model. Then, the data can be tokenised into units of tokens that can be analysed. Data preprocessing is meant to contribute towards the improvements of the performance of the model; that is, it has a significant effect on the model's outcome. This includes aspects such as the removal of any punctuation, as well as the removal of emails and websites. The process of text analysis is performed after the process of preprocessing to understand the insight of the dataset, such as the number of words and their frequencies and how long the words and sentences are within the dataset. Analysing the data allows the process of determining the total number of unique words and word frequency.

For this study, the main goal was to analyse the unstructured NCHLT Sepedi text dataset and then apply the process of word tokenisation. The next task was to perform data visualisation, the graphical representation of the dataset. Several factors can influence the outcome of the model developed when dealing with unstructured data, such as its nature, semantic complexity, and improper preprocessing of the data (Ramzan et al., 2016; Gurusamy & Kannan, 2014).

3.2.1 Text data preprocessing

Preprocessing the data is crucial before feeding it into deep learning algorithms as it can significantly impact the model's performance. The first step involves replacing special characters with spaces, as they are not typically part of the language used in the study. However, letters with the diacritic such as "š" may be included in the language and should not be removed. The second step involves breaking up sentences based on punctuation marks. This helps to make the text more manageable and allows the model to process each sentence separately. Punctuation marks are used to conclude sentences in sentence structures, which makes this step important for model execution.

The third step in data preprocessing is to remove non-alphabetic words, including numbers. This helps the machine recognise and construct full sentences using all the words in a sentence. Finally, to reduce vocabulary size, all words are converted to lowercase. In NLP, vocabulary size is an important factor, and data preprocessing plays an important role in downstream NLP applications (Anandarajan et al., 2019; Chai, 2023). This process enhances efficiency, accuracy, and other relevant scores, allowing any deep learning technique to train the model accurately.

3.2.2 Data Analysis

Understanding the structure of an unstructured dataset is crucial for effective data analysis. As a result, it is necessary to gain an understanding of the word lengths within the dataset, as well as how each sentence is structured in terms of the sentence composition. Several factors influence how well a language model performs, including the length of words and sentences. Various approaches have been employed by different authors to analyse the effect of word and sentence length on the corpus and its influence on the performance of language models (New et al., 2006; Lewis & Frank, 2016). Understanding these factors can lead to a more effective analysis and model development in NLP.

In any human language, sentences are considered the fundamental unit of language, and their length has traditionally been measured by the number of words in each sentence. Each sentence is expected to convey a complete message or idea, as indicated by research studies (Denner et al., 2018). Taylor (2015) examined the effect of word and sentence length on language using various approaches. The research also explored how these factors affect the performance of language models. Words comprise a few characters, while sentences comprise a few words, and both words and sentences can vary in length. Factors such as syllables, phonological measures, and orthographic neighbors can influence the length of words, the recognition of words within a sentence, and the meaning of words within a sentence (New et al., 2006).

Different linguistic properties can provide different views on words, regardless of their length. In line with Zipf's laws in language study, the shortest words tend to be the most frequent. Therefore, assigning word lengths based on frequency can be both highly efficient and concise (Zipf, 1935; Kanwal et al., 2017). The length of a morpheme, the smallest part of the word that cannot be divided into smaller parts, is a better predictor of its content than its frequency of use in a language. The word distribution represents how often certain words appear in the Sepedi dataset.

Table 3.1 represents how the dataset was divided according to words and sentences. The Sepedi dataset comprised approximately 2,117,598 words. The total number of words comes after the preprocessing process. The training dataset consisted of 1,483,422 words, while validation and testing consisted of 314,959 and 319,217, respectively. The table also provides insightful information about the number of sentences each set has. The total number of sentences after the process of preprocessing was 167,353. The training dataset consisted of 117,130, the validation dataset had 25,101, and the testing dataset had 25,122.

Table 3.1 Word distribution among data sets.

NCHLT Sepedi dataset

Datasets	%	Total words	Total Sentences
Training	70	1,483,422	117,130
Validation	15	314,959	25,101
Testing	15	319,217	25,122

This process provides an understanding of the distribution of words in the dataset. Fig 3.1 Shows the frequency of the words in training, validation, and testing datasets. The top five (5) most frequent words in all three (3) datasets consisted of the same words, namely "*go*, *le*, *ke*, *ya*, *a*". Regardless of whether it is from the dataset train, validation, or test, in training and validation datasets, "*go*" appears the most frequently with 94198 in the training dataset and 19766 in the validation dataset.

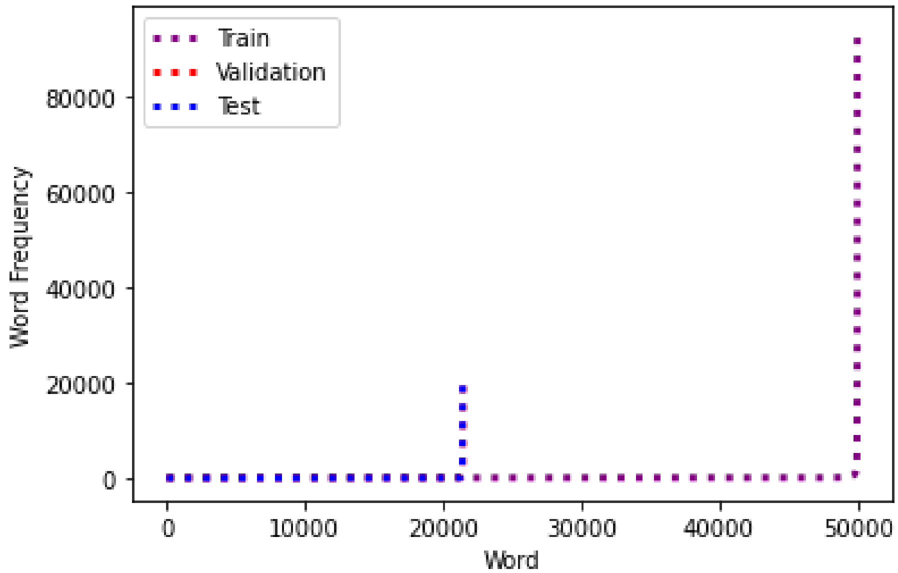


Figure 3.1 Word count frequency for train, validation, and test sets

There are about 26,364 words in the training dataset that appear only once, 12,083 such words in the validation dataset, and 12,007 in the testing dataset. The training dataset has 23,511 words with frequencies between 2 and 999. In the validation and testing dataset, 9,340 and 9,385 words fall within the same frequency range. Then the words with a frequency count of range 1000 to 95000 in the training dataset is 129 and, in the validation and testing dataset, the set is 30, respectively. Words appearing only once dominate the training dataset at 52%, followed by words with a frequency of 47% for occurrences between 2 and 99. The less occurring words in the dataset are words with a frequency of 2.5 % for words with occurrences between 1000 and 95000. Observing the training dataset, we notice that the validation dataset is primarily characterised by words that appear only once at 56%, followed by words with frequencies between 2 and 999 at 44%. The less occurring words in the validation dataset are words with a frequency of 0.1 % for words with occurrences between 1000 and 20000. The pattern is the same as for the testing dataset. The datasets do not have a fair distribution because less frequent words, especially those occurring only once, dominate all three of them.

The process of performing text analysis on the Sepedi dataset provides an understanding of how each Sepedi word is structured in terms of size. The longest word in the training dataset is "*modulasetulomohlahlobimogolobatho,*" with a length of 33 characters. Respectively, the longest words in the validation and testing datasets are "*motlatšamongwaledipharephare* and *motlatšamopresidente*". Fig 3.2 Shows the word length distribution in training, validation, and testing datasets. The words with a length between 0 and 5 characters were found to contribute 14% and 16%, respectively. Words with a length between 5 and 15 characters contribute 84% and 82%, respectively. The longest words in the training, validation, and testing dataset contribute about 2% and 1% of the datasets. In terms of popularity in the Sepedi words, the datasets are fairly distributed, with more than 80%. This means that the percentages of words and word length in train, validation, and test sets provide nearly the same percentages.

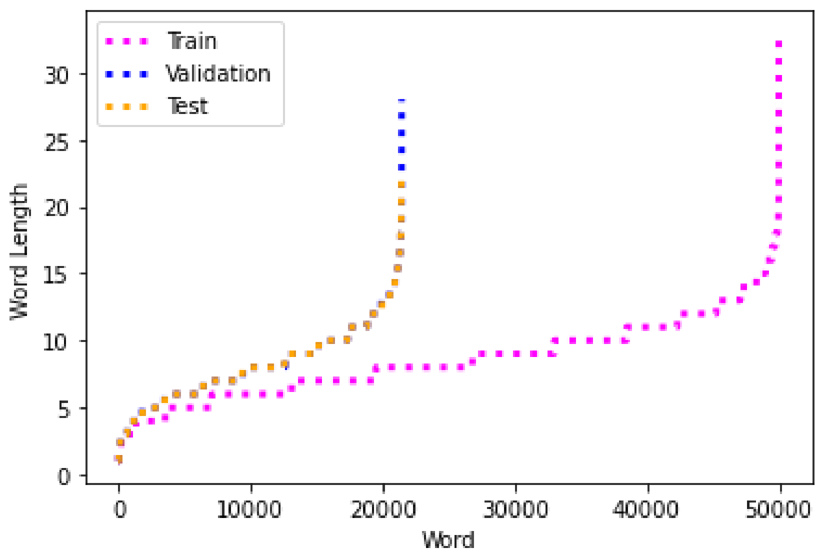


Figure 3.2 Word-by-word length graph for train, validation, and test sets

Fig 3.3 Depicts the word length frequency in the training, validation, and testing datasets. The word length is used to determine the effects of these datasets in effectively training text generation mode. The combination of words with the length of one character in a training and validation dataset has a frequency of 697792 and 143231, respectively. The combination of the frequency of other words in the datasets adds up to 171728 for the training dataset and 785628 for the validation set. These frequencies are made from a total of 26- and 23-character words in the training and validation dataset.

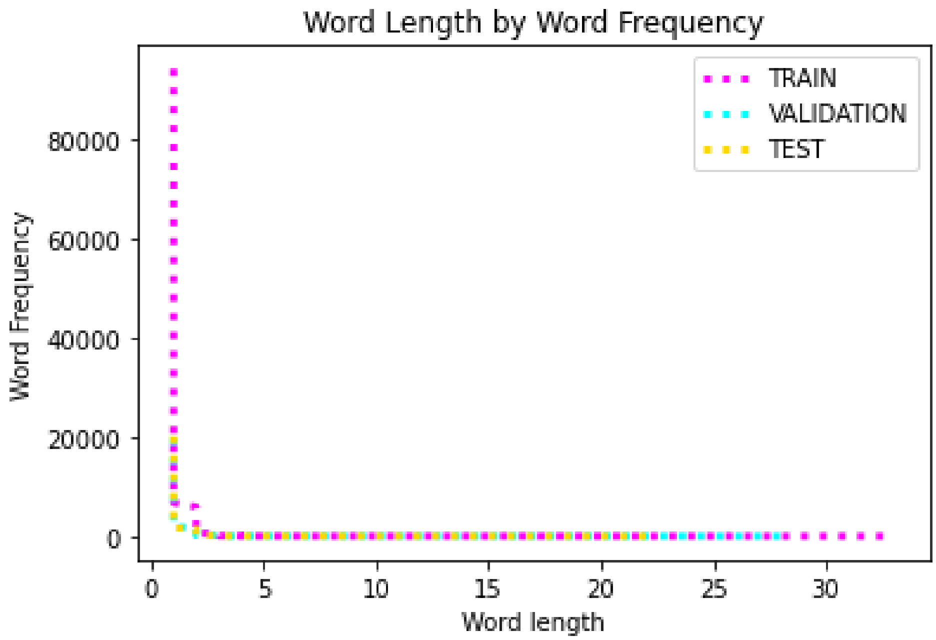


Figure 3.3 Word length by word frequency for training and validation and test sets.

- Sentence distribution

Fig 3.4 Indicates how many times each sentence appears within the dataset. The dataset consists of a total of 167,353 sentences. The training dataset comprises 70% of the entire dataset, while the testing and validation datasets each account for 15%. Most of the sentences occur once in the training, validation, and testing dataset, respectively. The phrase “*le ge go le bjalo*” appears 248 times in training, 53 times in the validation dataset, and 46 times in the testing dataset. Fig 3.4 indicates the percentage of the sentences that are repeating in the dataset. This sentence is often translated as “However”.

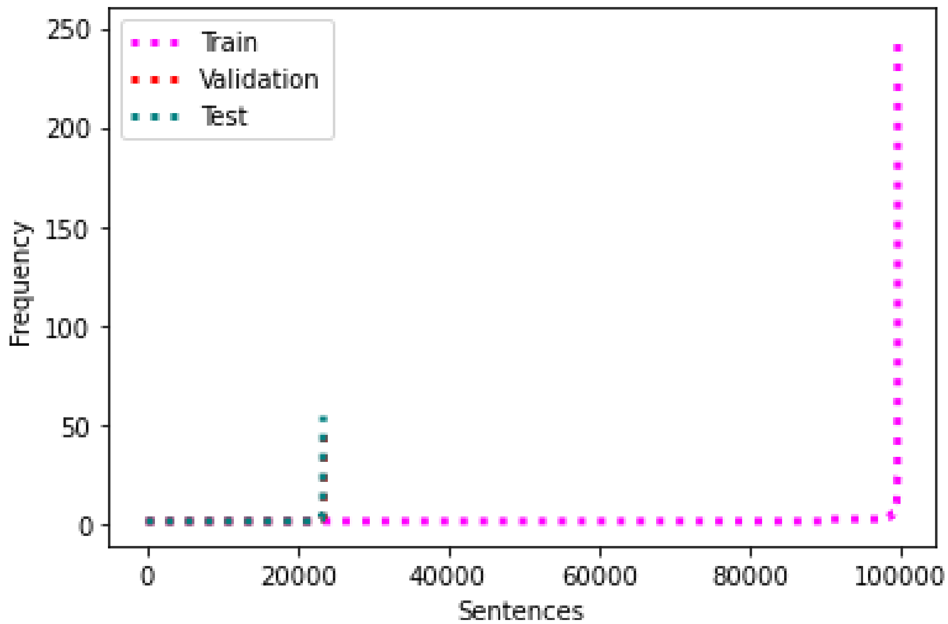


Figure 3.4 Sentence frequency for train, validation, and test sets.

Fig 3.5 Shows the length of the sentences in the dataset. The shortest sentence in the training dataset, validation dataset, and testing dataset are of length 1, respectively. The longest sentences are 98 for the training dataset, 87 for the validation dataset, and 76 from the testing dataset. By looking at the datasets, one can gain an idea of how long or short the sentences can be generated by the trained model. The sentences that consist of 40 or more words in the training dataset contribute to 1%, 3%, and 2% in the testing dataset.

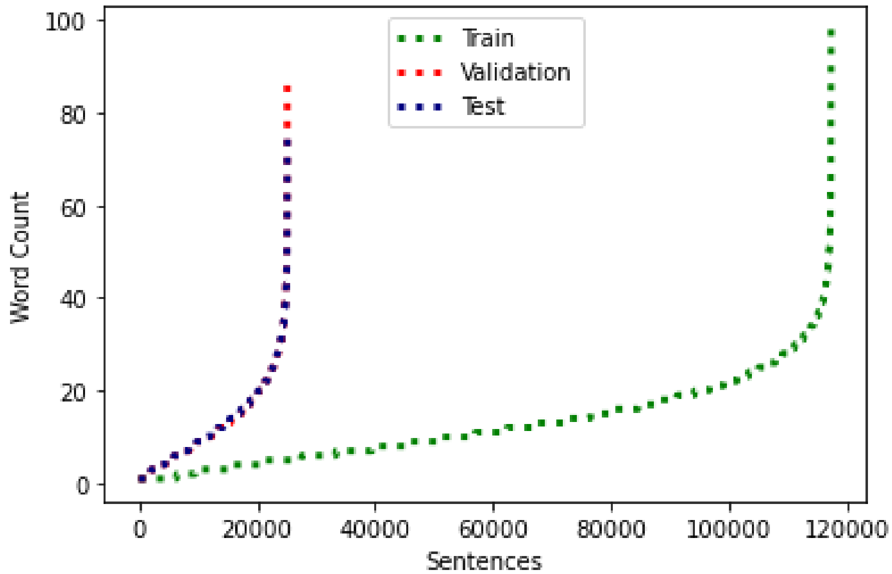


Figure 3.5 Sentence length for train, validation, and test sets.

Research has shown that sentences with an average length of more than ten words tend to convey more meaning in text-to-speech synthesis compared to sentences with an average length of less than five words (Chu & Chiou, 2016). Additionally, longer sentences are more likely to be understood than shorter ones (Just & Carpenter, 1980).

Fig 3 6 illustrates the frequency of the sentence length across the datasets. The graph indicates the flat line between a sentence length of 40 and a sentence length of 100. This indicates that the sentences with these lengths appear between the range of 1 – 185 in the training dataset, 1 – 42 for the validation dataset, and 1 – 37 for the testing dataset. The training dataset has the highest frequency of 6465 for a sentence length of 5. For example, sentences such as "*thuto ke sehlare sa bafsa*".

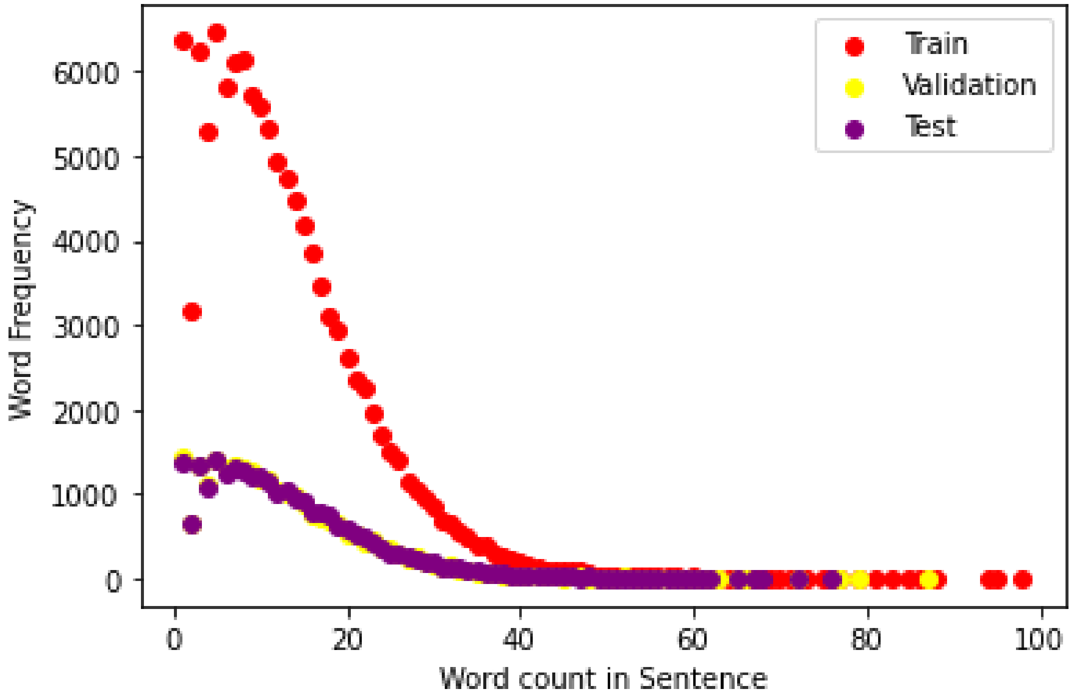


Figure 3. 6 Word count in sentence frequency across the datasets

3.3 Model Development

The development section discusses several key steps, including tokenisation, defining the LSTM-Sepedi Attention model, fitting the LSTM-Sepedi, defining the GPT-Sepedi Transformer model, and fitting the GPT-Sepedi Transformer model. Tokenisation was the first step in the development process, which involved breaking down the input text data into individual tokens or words. This was achieved using the tokenisation module provided by the TensorFlow library. Next, in the development, we defined the LSTM-Sepedi Attention model. The design consisted of specifying the architecture of the model, including how many layers of LSTM-Sepedi there should be, how many neurons should be in each layer, and what activation function should be used in each layer. The goal of the LSTM-Sepedi model was to learn the word patterns and relationships in the input text data and generate coherent and syntactically correct text. Once the LSTM-Sepedi model parameters were properly defined, they were fitted to the training data using the fit() method. During training, the model adjusted its internal parameters to minimise the difference between the predicted and actual output, gradually improving its ability to generate accurate and natural-sounding text. Then, the GPT-Sepedi Transformer model was defined by specifying the architecture of the model, which was based on transformer neural network architecture. Using transfer learning, the GPT-Sepedi Transformer model was pre-trained using the Sepedi dataset. Finally, the GPT-Sepedi Transformer model was fitted to the training data using the fit() method. During training, the model adjusted its internal parameters based on the input data and gradually improved its ability to generate coherent and syntactically correct text.

3.3.1 Tokenisation

Tokenisation involves breaking down a text into smaller units, called tokens, which are used for NLP tasks such as classifying text, recognising entities, and translating languages. (Jurafsky & Martin, 2022). There are types of tokenisation. Word-level tokenisation is where words are the basic unit of text. Using this tokenisation method, each word is treated independently. However, tokenisation can also be done at the character level, the sentence level, or the sub-word level. There are various techniques for tokenisation, including rule-based tokenisation, statistical tokenisation, and neural network-based tokenisation. (Rai & Borah, 2021). The tokeniser used for the LSTM-Sepedi Attention Model was a word-level tokeniser by default, as defined by the Tokeniser class. The text was split into words based on spaces and punctuation, and each unique word was assigned an integer. Whereas the GPT-Sepedi Transformer Model used a custom character-level tokeniser, which converted each character into a different integer token and then back. The character-level tokeniser provides the mode with the ability to process the data in a flexible way, which gives the model the ability to learn any combination of characters. The tokeniser used was a custom character-level tokeniser, which converted each character into a different integer token and then back. The character-level tokeniser provides the mode with the ability to process the data in a flexible way, which gives the model the ability to learn any combination of characters.

3.3.2 LSTM-Sepedi Attention Model

The LSTM-Sepedi Attention model is designed for generating text in the Sepedi language. It utilises an LSTM layer within a deep neural network architecture. The model is configured with various parameters that enable it to process input text data, convert it into a list of tokens, and subsequently generate the desired output. The process of embedding layers is used to convert the text into a vector. The model architecture and sizes of the deep learning model have several layers with different output shapes. The first layer is an input layer that takes a sequence of tokens followed by an embedding layer. The embedding layer converts the input sequence, such as words or tokens, into continuous, high-dimensional vectors (dense vectors). Within the embedding later, some parameters represent the numbers of weights and biases that need to be learned during training. The next layers are the LSTM layers. The model consists of two LSTM layers, which are stacked with several attention heads to be able to focus on the important parts of the input vectors during training. The dense layer was used to connect all inputs as some fully connected neurons then pass them to the activation function. Once this process was complete, the activation function was used to convert the vectors back into tokens as output sequences. All these parameters are trainable, which means that they were updated during training to improve the model’s performance on the text generation task.

3.3.3 GPT-Sepedi Transformer Mode

The GPT-Sepedi Transformer model is designed in such a way that it can generate text in the Sepedi language. The model was developed using the GPT technique, which uses transformer architecture. The first step involves importing the necessary libraries, including TensorFlow and the Transformers library. Loading the train and validation data files follows, requiring them to be prepared in data preprocessing. While developing the GPT system, the model was created using Pytorch and different parameters. The parameters used during the development of the model include block size, batch size, number of iteration steps, and the learning rate. The model then loads the text data files and creates a set of vocabulary of unique words. The vocabulary set will be converted into a set of tensors using Pytorch, which are used for both training and validation. The generation parameters are defined to train the model. The performance of the model was measured using loss at every iteration. After training the model, the model then generated the text based on the given input.

3.4 Evaluation

It is crucial to evaluate a language model's performance to determine whether it is effective in accomplishing its task or not. To evaluate a language model, it is necessary to compare its achievable accuracy and loss. In the process of developing a language model, loss functions calculate the difference between predicted and actual outputs. The goal of Equation 2 Is to minimise this difference during the training process. Meanwhile, Equation 1 measures the percentage of samples that are correctly classified among the total number of samples (Goodfellow et al., 2016).

The LSTM-Sepedi Attention model will be evaluated based on loss and accuracy, while the GPT-Sepedi Transformer model was measured on loss value. As the loss decreases, the accuracy of the language model increases. The accuracy and loss metrics are, therefore, crucial for assessing language model performance. According to Goodfellow et al. (2016), the loss function is an important measure of a model's performance; the accuracy function, on the other hand, measures the percentage of correctly classified samples out of a total of samples. In summary, accuracy and loss are essential metrics that are used to evaluate the performance of a language model during its development.

The accuracy is calculated by:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Equation 1

The loss function is calculated by:

$$LOSS = -\sum_{i=1}^{output\ size} y_i \cdot \log \hat{y}_i$$

Equation 2

where an encoded vector y_i represents the true label, while a predicted probability vector $\hat{y}$ It is a vector of predicted probabilities for all classes. The loss is the negative log-likelihood of the true class.

When training a machine learning model, it is crucial to monitor the model's performance by examining its loss and accuracy metrics. Loss functions measure the difference between predicted and actual outputs during training. It is a mathematical function that calculates the error of the model's prediction for each training example.

To determine if a model is effective, we typically compare its accuracy and loss values between the first epoch and the nth epoch. The nth epoch refers to the final epoch of the training process. If the loss value at the nth epoch is lower than that at the first epoch, it indicates that the model has learned to make fewer errors. Conversely, if the loss value does not decrease over time, it indicates that the model is not improving. Additionally, an effective model should achieve its maximum accuracy by the nth epoch when compared to the initial epoch. In this sense, the model has learned to accurately predict the training data's outputs. When the loss value is low and accuracy values are high, the model makes relatively small errors. However, if the loss is higher, it may result in significant errors, which can be detrimental to the performance of the model.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) was used to evaluate the automatically generated text. ROUGE is a text evaluation score that is used to evaluate the automatically generated text against the human-written text. There are different kinds of ROUGE scores, including ROUGE_1, ROUGE_2, and ROUGE_N (Lin, 2004; Rosari & Noever, 2023). These rouges differ in terms of what kind of n-grams they consider when evaluating the text. It can be unigram, which only looks at individual words generated, and bi-grams, which look at a set of text or sentences with the generated text. ROUGE makes use of 3 metrics: precision, recall, and F-score. These metrics are used to evaluate the performance of the model to provide insight into how relevant the generated texts are. ROUGE 1 was used to determine the accuracy and relevancy of the Sepedi text generation model-generated text

Both models were trained using the following parameters: batch size, LSTM weights, dense layer and embedding size, learning rates, and attention heads. This is because the parameters used are essential for maximising model performance, particularly when training languages with limited resources. While LSTM weight and thick layers deal with pattern recognition and precise output generation, bit size affects learning stability and memory efficiency. A well-tuned learning rate ensures steady convergence, attention heads improve dependency modeling, and embeddings capture linguistic nuances while training a transformer model. To generate correct and relevant text, these parameters are necessary to balance computing efficiency, learning capacity, and flexibility to Sepedi's complicated language structure. It is impossible to ignore the impact of a limited dataset, though. The model's capacity to generalise effectively is limited by the availability of data, which frequently leads to overfitting or underfitting. This difficulty highlights how crucial parameter optimisation is to striking a balance between learning potential, computing efficiency, and flexibility in response to Sepedi's intricate linguistic structure. The model's ability to accurately represent linguistic diversity is further limited by small datasets, hence, it is essential to improve the training process via data augmentation or transfer learning strategies.

3.5 Conclusion

Based on the methodology described, it can be concluded that the development and evaluation of NLP models, particularly the LSTM-Sepedi Attention and GPT-Sepedi Transformer models, require a comprehensive and systematic approach. This approach should include the careful selection and preparation of appropriate datasets, such as the NCHLT Sepedi dataset, as well as thorough text data analysis. Furthermore, model evaluation is crucial to determining the effectiveness of the developed models in generating high-quality sentences. Evaluation metrics such as loss and accuracy provide insight into the model's performance and the ability to minimise errors during the training process. In addition to these metrics, the ROGUE score is a valuable tool for comparing the quality of sentences generated, allowing for an objective assessment of the effectiveness of each model.

4. CHAPTER 4: RESULTS

4.1. Introduction

This chapter is set out to discuss the results of a text generation model for the Sepedi language. The following sections provide insight into the results of the experiments conducted for the LSTM-Sepedi Attention model using small and large datasets, and experiments for the GPT-Sepedi Transformer model. The models were trained using different parameters, such as batch size, learning rate, activation function, embedding sizes, and number of attention heads, to optimise model performance.

4.2. LSTM-Sepedi Attention Text Generation Model Results

In the text generation task, the effectiveness of a model to produce the required results is often measured through its performance on training and validation datasets. In this section, the results are presented by exploring the visualisation of model training outcomes using various configurations, such as batch sizes, dense layers, and LSTM weights. Throughout the process of training, the results have indicated that different parameters can make the model act differently. Table 4.1 Show the parameters that led to the best-performing model for both the LSTM-Sepedi Attention model and the GPT-Sepedi Transformer model. The table has been referenced throughout the results discussions.

Table 4.1 Best-performing model parameters

Models	**Parameters**	
LSTM-Sepdi Attention Model	Batch size	512
	LSTM Weights	200
	Dense layer	200
GPT-Sepedi Transformer model	Batch size	256
	Embeddings	128
	Attention	16

	heads	
	Learning rate	

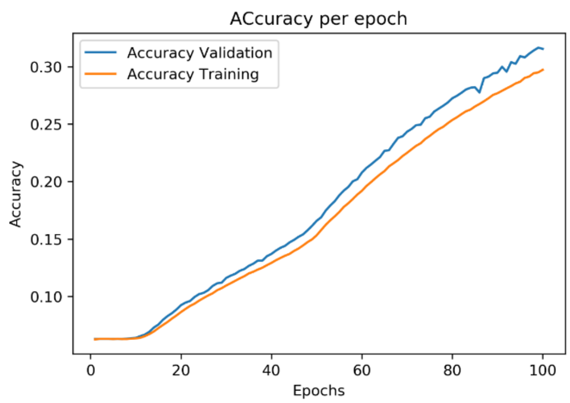
Table 4.2 shows the model comparative analysis of the LSTM-Sepedi Attention model and GPT-Sepedi transformer model. The LSTM-Sepedi Attention model does well on small datasets compared to the GPT-Sepedi transformer model, which requires a larger dataset. While LSTMs process sequences sequentially, leading to slower training times, they are more efficient on smaller datasets and can effectively learn from limited data. However, they may struggle with capturing Sepedi's rich morphology. In contrast, the GPT-Sepedi transformer model excels at handling complex morphological variations and processing sequences in parallel but demands substantial computational resources and larger datasets to perform effectively

Table 4.2 Comparison Analysis for LSTM-Sepedi Attention And - Models

ASPECT	**LSTM-Sepedi Attention**	**LSTM-Sepedi Attention**
Dataset	Perform better on small dataset	Perform better on large dataset
Data modeling	Processes sequences sequentially	Processes sequences in parallel
Training time	Slower for long sequences due to sequential processing	Faster with sufficient resources, due to insufficient resources, it took time to train
Handling Morphology	Struggles to capture Sepedi's rich morphology	Better at understanding morphological variations

4.2.1 Small dataset LSTM-based results

Different experiments were conducted using different dataset sizes. The portion of the dataset, which is 15% of the whole dataset was used as the training set, here referred to as a small dataset. The model was evaluated with the portion of the dataset which is the same amount (15%). The 15% dataset was needed to evaluate the model performance using a small dataset and a data hungry deep learning architecture. Fig 4.1 and Fig 4.2 Show the loss and accuracy results obtained from training the model using a 64-batch size with 100 epochs. The Fig 4.1 Shows a sharp increase in accuracy as the number of epochs increases. The highest accuracy obtained is 31% at epoch 100 while Fig 4.2 Shows a decrease in loss per epoch with a loss value of 3.2 at epoch 100.



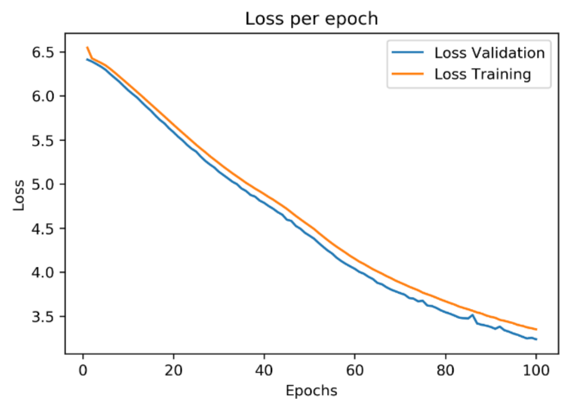
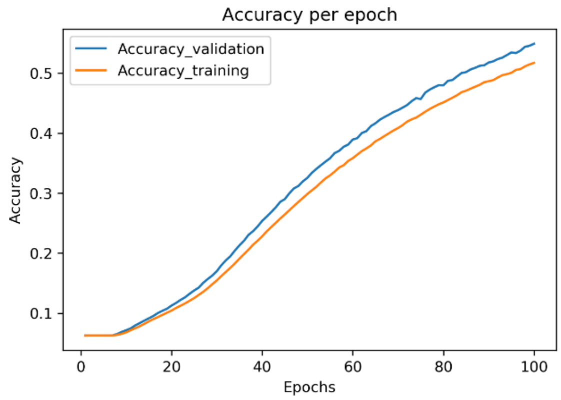


Figure 4.1 LSTM-Sepedi Attention Model Accuracy *Figure 4.2 LSTM-Sepedi Attention Model loss results*

Fig 4.3 and Fig 4.4 shows the accuracy result of the model trained on 128 batch size with 100 epochs. At the beginning of training the model indicated an accuracy of 6.3% at epoch 1 for the validation dataset. The improvement in accuracy indicates that the model was able to learn the relationships within the data. This is shown by the increase in accuracy from epoch 1 to epoch 100 with an accuracy of 55% for the validation at epoch 100.



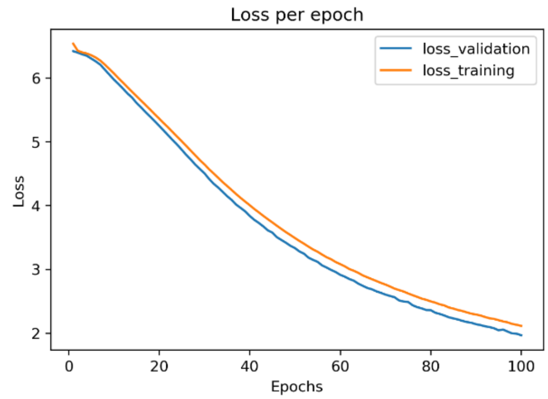


Figure 4.3 LSTM-Sepedi Attention accuracy results. *Figure 4.4 LSTM-Sepedi Attention Loss results*

On the other hand, the loss result shown in Fig 4.4, the loss decreases from 6.4199 at epoch 1 to the lowest validation loss of 2.11 at epoch 100. The results show that the improvement in accuracy is very slow. One of the major factors that may have caused the behaviour of the model is the time the model takes to complete one epoch. This may be due to the complexity of the model. Fig 4.1 has the highest accuracy of 31% at epoch 100, while Fig 4.3 shows the accuracy results of 55%. The results indicate that adjusting the batch sizes did improve the model results. The batch size affects the model in such a way that increasing the size of the batch size increases the training speed. At the same time, small batch sizes are good for generalising the performance of the model, while large sizes may lead the model to overfit.

Fig 4.5 Depicts the overlapping of the common words that are found in both small datasets that were used for results in Fig 4.1 and Fig 4.2 . The graph shows the top 50 common words found in both the training and validation datasets. These common words cause the training and validation results to overlap. Thus, the model behaves the way it does in terms of training and validation results.

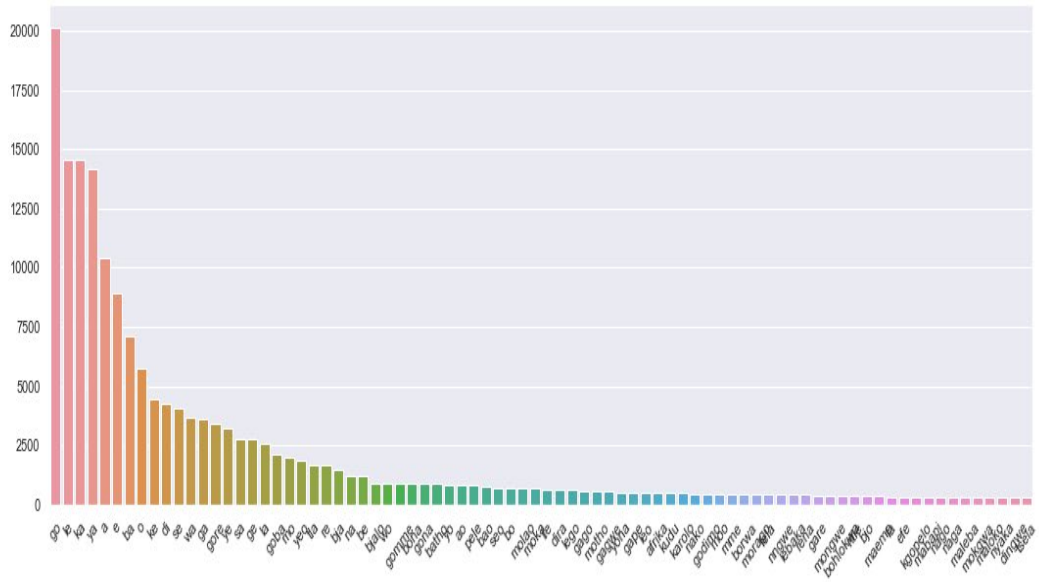
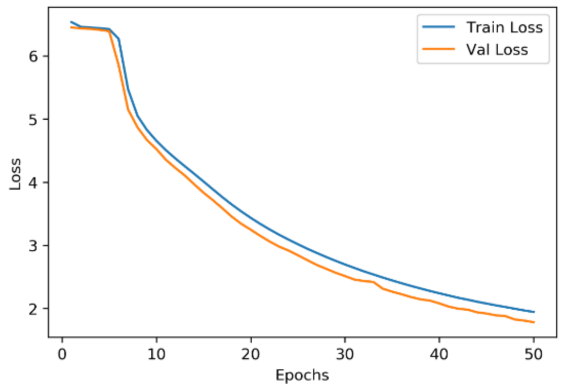


Figure 4. 5 Overlapping common words in a small dataset

4.2.2 LSTM-Sepedi Attention model results using a large dataset.

The following section presents a graphic representation of a text generation model for the Sepedi language. Each visualisation illustrates the model's performance under different scenarios in the graphical representation. The variation in batch sizes, the choice of LSTM weights, and the selection of dense layer configurations all contribute to building a good-performing model.

Fig 4.6 and Fig 4.7 Loss and accuracy graphs are generated using a batch size of 512, 200 LSTM weight, and a Dense layer of 200. According to Fig 4.6 and Fig 4.7, the validation loss decreased from 6.44 to 1.78, and validation accuracy improved from 6% to 62%. Specifically, Fig 4.7 Started flat from epochs 1-5, then showed a steep increase in accuracy of 6%-24% from epochs 6-8. From epoch 9-50, the graph increases in accuracy until epoch 50.



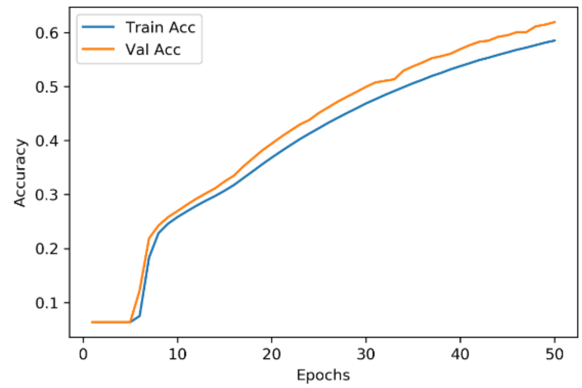
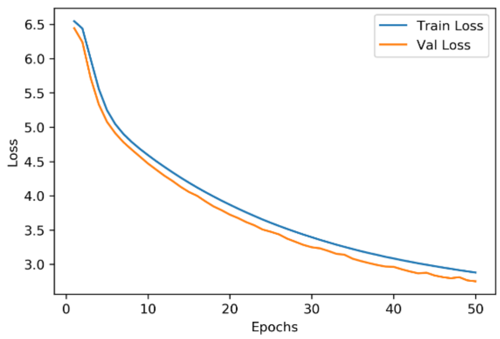


Figure 4.6 LSTM-Sepedi Attention Loss results. *Figure 4.7 LSTM-Sepedi Attention Model Accuracy results.*

The results show that the loss is decreasing per epoch, which indicates that the model is learning from the dataset. For training purposes, completing one epoch takes approximately 1221 seconds using a GPU, which can be about 20 minutes per epoch. On the other hand, accuracy is set to increase over the epoch, which indicates that the model is showing improvements in terms of performance, thus the model to make predictions on the training dataset. The loss and accuracy derived from the validation dataset indicate that the model can generalise to a new dataset, as it has learned patterns that are not specific to the training data alone. Comparing Fig 4.6 and Fig 4.7 results with the results conducted using a small dataset, it can be observed that the model has performed better with the bigger dataset. The model has the lowest loss of 1.78 using a large dataset compared to 3.34 and 2.00 loss using a small dataset. The model shows the highest accuracy of 62% compared to the accuracy of 31% and 52% using a small dataset, respectively.

Fig 4.8 shows the loss results obtained by training the model on 100 dense layers and 100 LSTM weights. At epoch 50 the loss results indicate the loss of 2.75 from a loss of 6.5 at epoch 1. The performance of the model improves over each epoch. The accuracy of the model shows a gradual increase. Fig 4.8 and Fig 4.9 shows that the model accuracy starts from approximately 6% on the validation set on epoch 1-2. There is a steady improvement from epochs 2-4. Then the graph indicates that the model keeps improving over each epoch using validation sets with an accuracy of 46% on validation.



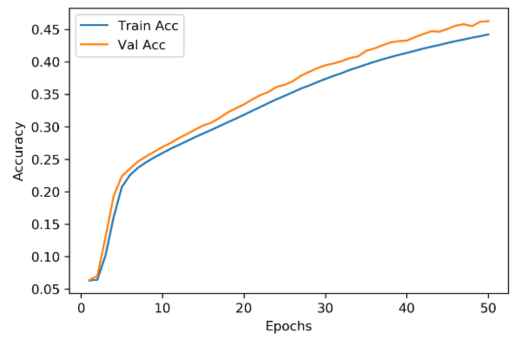
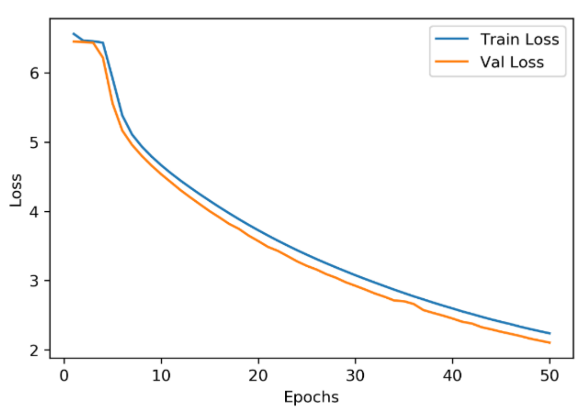


Figure 4.8 LSTM-Sepedi Attention Model Loss results. Figure 4.9 LSTM-Sepedi Attention Model Accuracy Results

Fig 4.10 and Fig 4.11 depict the model, which was trained on 200 weights and 200 dense layers using a batch size of 1024. The model obtained the validation loss results of 2.1, and the model provides a validation accuracy of 58%, as shown by Fig 4.10 and Fig 4.11. The figures show that the loss and accuracy improve over each epoch on the training set. Note that the validation accuracy improves but at a slow pace. This could be due to the size of the dataset and the adjustment of the parameters, and the model is not capturing the patterns properly. These issues will be part of the future work.



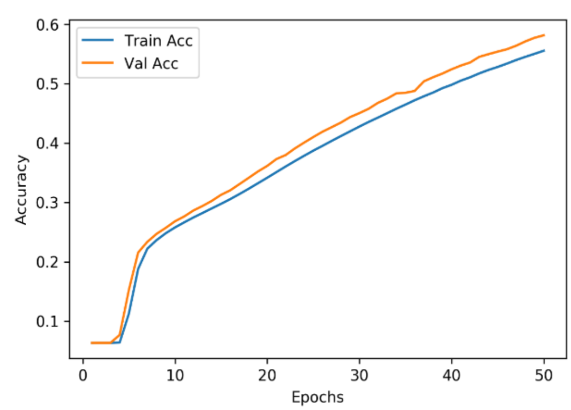


Figure 4.10 LSTM-Sepedi Attention Model Loss results. Figure 4.11 LSTM-Sepedi Attention Accuracy results

The last experiment conducted using LSTM with attention was done using 2048 batch size 200 LSTM weights and 200 dense layers. Fig 4.12 and Fig 4.13 Show the loss and accuracy results obtained after training the model on the 2048 batch size. It can be observed that at epoch 50, the model indicates 2.75 validation loss. It can also be observed that the validation accuracy is 48%. It can be observed that the validation loss reduces as the model trains. The results show that the model can generalise and learn from the data. Also, the accuracy of the validation set improves over the epoch, which shows some performance. The final accuracy of 48% indicates that even if the model shows an improvement, 48% is not the desirable result for generating the text.

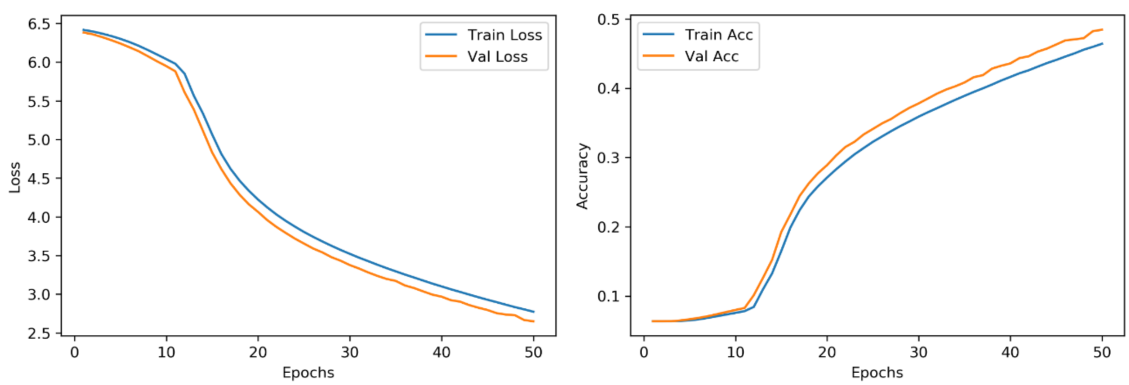


Figure 4.12 LSTM-Sepedi Attention model Loss results. Figure 4.13 LSTM-Sepedi Attention Accuracy results

What is observed throughout LSTM-Sepedi Attention model experiments is that batch size affects the performance of the model. Based on the results, the more one increases the batch size while keeping the LSTM layer and dense layer constant, the less the model will perform; thus, the model will provide less and less accuracy. Training of the model takes time and requires high-performance systems. Thus, training the model using the overall dataset came with constraints when trained on 100 epochs using a large dataset. This includes the time taken to complete each epoch, which indicates the requirements for more computational resources.

The highest-performing model from all the results is provided in Fig 4.6 and Fig 4.7. The model was trained with 512 batch size on 200 LSTM weights with training accuracy of validation accuracy of 62% and loss validation loss of 1.78, respectively. This model is the one considered for generating text.

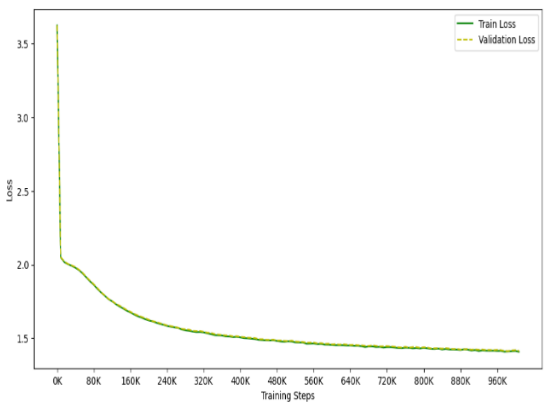
4.3. GPT- Sepedi Transformer Model

Transformer models are sophisticated machine learning models specifically designed to handle sequence data input. These models utilise an encoder-decoder mechanism and an attention mechanism to process the input data. The attention mechanism allows the model to focus on one word at a time while simultaneously processing the entire sequence, thereby enabling efficient parallelism. The experiments conducted involve training the transformer model using different parameters ranging from batch sizes, number of steps, embedding sizes, number of attention heads, and learning rates. The progress and performance of the model were monitored using the training and validation loss.

4.3.1 GPT-Sepedi Transformer model training

Each experiment is done using different parameters while keeping other parameters constant, where at each step interval, the training and validation loss is recorded. The lower the validation loss value, the better the model performance. Fig 4.14 and Fig 4.19 show the results, which were conducted using 64,128 and 256 batch sizes, 64 and 128 embeddings, and 8 and 16 attention heads. Throughout the process of training the GPT-Sepedi Transformer model, the learning rate of 1e^-3 was kept constant.

Fig 4.14 Details the loss values for the validation and training datasets at different steps. The results were conducted using batch size 64, 64 embedding sizes, and 8 attention heads. The loss starts at 3.62 at step zero and shows a significant decrease at step 8000 with a loss of approximately 2.05. Deep in training the model, the model seems to be doing well, with a final loss of 1.44. The consistent decrease in loss indicates that the model is learning and improving its performance during the training.



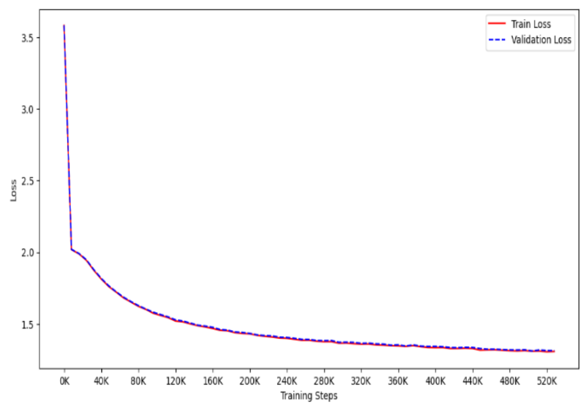
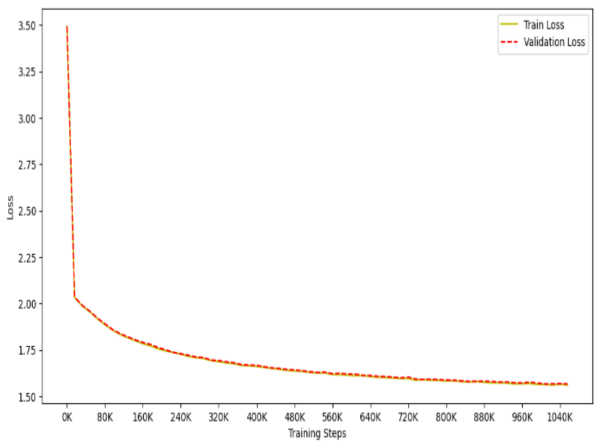


Figure 4.14 GPT-Sepedi Transformer Loss 64 embedding. *Figure 4.15 GPT-Sepedi Transformer Loss 128 embedding*

On the other hand, the graph in Fig 4.15 Shows the same behaviour with the loss gradually decreasing. The rate at which the loss decreases in Fig 4.15 Indicates that the increase in embedding size and attention heads has a significant impact, as shown by the consistent reduction in loss value.

Fig 4.16 visualises the training and validation loss of the results conducted using batch sizes 128, 64 embedding sizes, and 8 attention heads, while Fig 4.17 shows the loss results conducted using batch sizes 128, 128 embedding sizes, and 16 attention heads. Both graphs show a steady decrease in loss where the slope of the loss curve decreases over time. Thus, the curve decreases towards the end, indicating that the model is improving. The loss results provide a better training process with better improvement in loss values where the initial loss was 3.5. At step

16000, both graphs show the loss value, which dropped drastically to 2.04 and 2.01 from the initial loss. Fig 4.17 shows better results compared to Fig 4.16. This indicates that an increase in embedding size and attention heads had an impact on the performance of the model with a final loss of 1.57 and 1.26, respectively.



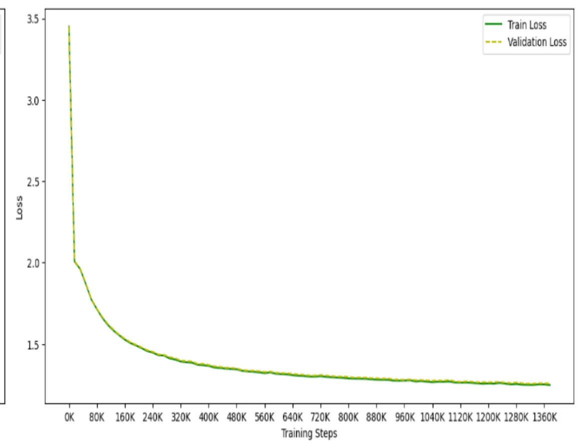
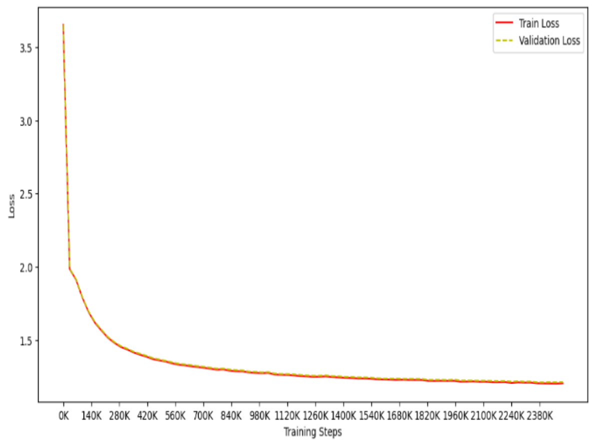


Figure 4.16 GPT-Sepedi Transformer Loss 64 Embeddings Figure 4.17 GPT-Sepedi Transformer Loss 128 embedding

Fig 4.18 and Fig 4.19 Shows the results conducted using Batch size 256 with 64 and 128 embedding sizes and 8 and 16 attention heads. Both loss values start at 3.67 and 3.65, respectively. The graph drops to 2.02 on Fig 4.18 and Fig 4.19 With a loss of 1.99 at step 16k. The loss results depict that the increase in model parameters has a significant impact on the model as the final loss on Fig 4.18 being 1.35 and Fig 4.19 With 1.21.



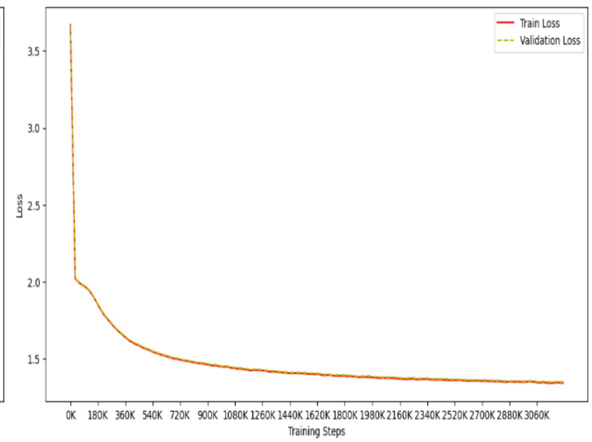


Figure 4.18 GPT-Sepedi Transformer Loss 64 Embeddings   Figure 4.19 GPT-Sepedi Transformer Loss 128 embedding

The take from training these models is that in terms of the impact of the model, capacity is that the model with larger embeddings and more attention heads tends to have a high effect on how the model trains. It improves the model performance, while the model trained with low embeddings and attention heads tends to improve at a slow pace. Looking at the loss trend, the loss seems to be decreasing from Fig 4.14 to Fig 4.19. As much as the loss was decreasing in all figures, the model with 128 embeddings and 16 attention heads seems to have a loss decrease better than models with 64 embeddings and 8 attention heads. The learning rate C seems to be the best rate for training the GPT-Sepedi Transformer model, as the loss results show a steady decrease with no visible instabilities. To avoid overfitting, the dropout rate of 0.1 was used. The dropout assisted the training by ensuring that when the model failed to learn, it stopped the training to avoid overfitting. Thus, an increase in model capacity may assist with enhancing the model's ability to capture and represent the complexity of the dataset and be able to generalize. The outmost performed model is the model results provided in Fig 4.17.

batho mo gabe batho le mmušo wena go kgokagega se ipeagadi ye e bona mo e ka swanelwa go fiwa o nyaka ditragalotšu tšeo di lego go laetšwego ka boso bja bafahloko
ka tsenyo tšeo di felago bonyanengwageng le tša setšwa go tšwetšwa bja bonalo ka šetša diphina tšeo di osekhwamo se tla saenya meepo ya ka fase ye e ebilwego
ge o phelana ga mehola
ka magareng ga ditemošo tša ditšhelete
gagwe medipelo di na le difostšo
ke ka fase gabo
ge a se bangwalela ka go bahlokwa le setsi
go maula dinete tšeo
boa

Figur 4.20 Generated Sepedi words

4.3.2 GPT-Sepedi Transformer model text-generated

The generated text will be compared to the Sepedi language vocabulary to check if the generated texts are valid. This will give an inside look at how the models performed. The text was generated using the model, trained using batch size 256, with hyperparameter 128 embeddings and 16 attention heads. This model was chosen to generate text due to the low loss value at the end of the training.

Fig 4.20 Shows the text generated by the GPT-Sepedi Transformer model. The models were compared based on the batch size 64,128 and 256 together with 64 and 128 embeddings and, lastly, 8 and 16 attention heads. Most of the words alone have a meaning but cannot make a comprehensive statement. Thus, the results we obtained when evaluating the text using the ROUGE Score in section 4.4.

Looking at Fig 4.21, it shows the frequency distribution of Sepedi Valid words from the generated text in Fig 4.20. The graph shows that "*ka"* is the most frequent word, which occurs 8 times, and "*go, le, and tleo" appear 4-5 times*. Most words appear once; thus, the distribution shows a small number of highly frequent words while most words appear with low frequency.

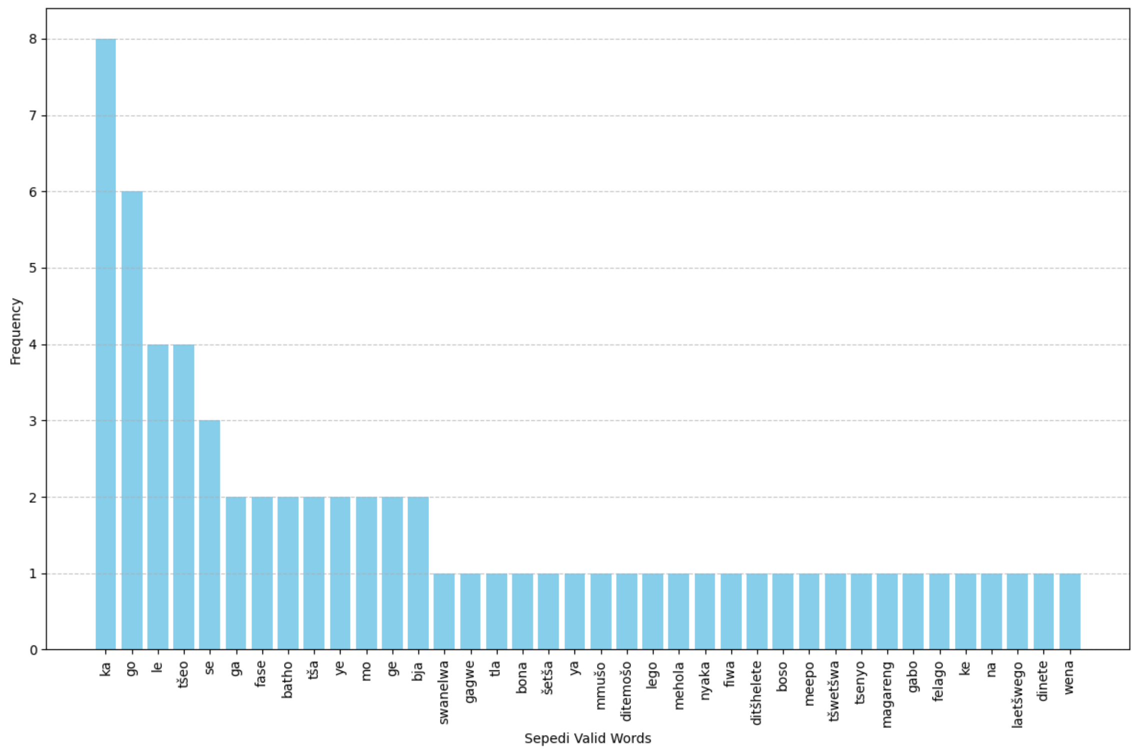


Figure 4.21 Generated Valid Sepedi words

Fig 4.22 represents Sepedi words by their frequency in the generated text in Fig 4.20. Most of these words appear only once, highlighting their uniqueness. These words may be considered special words that within the current vocabulary dataset, are not available. These words may remain relatively unknown or can be considered as special words.

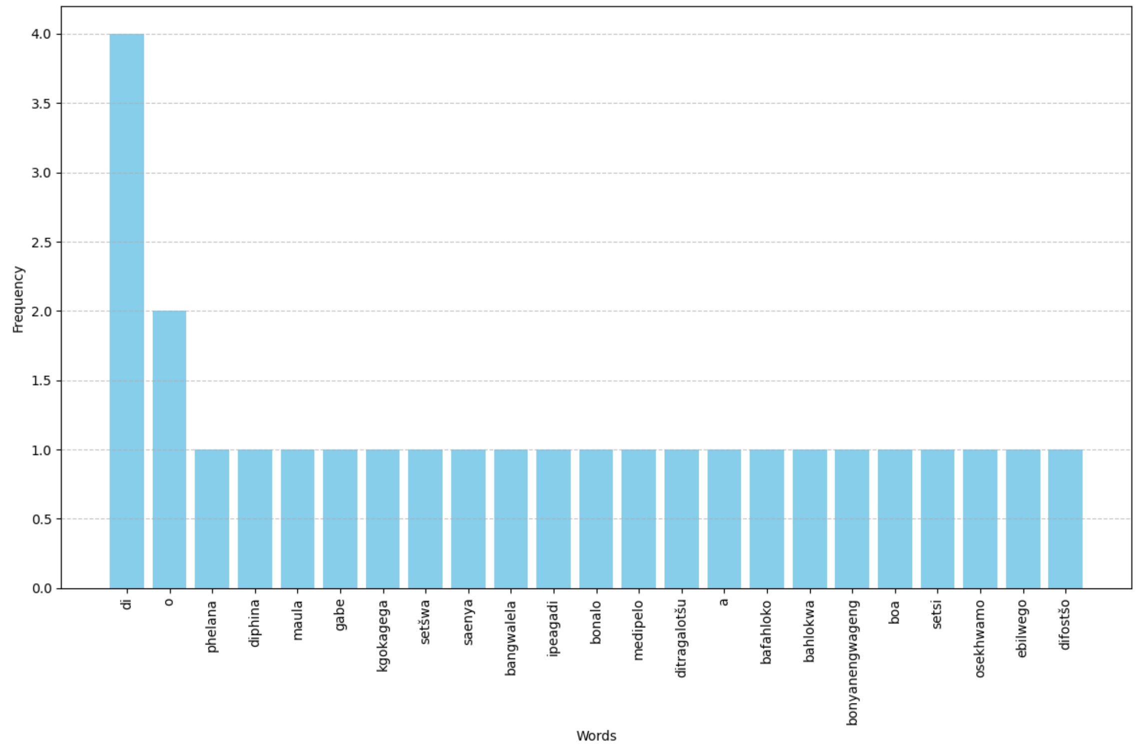


Figure 4.22 Generated unique and unknown Sepedi words

The bar graph in Fig 4.23 Shows that across all the batch sizes, the model with 128 embeddings and 16 attention heads consistently outperforms the model with 64 embeddings and 8 attention heads. This is an indication that increasing the embedding size and attention head enabled the model in a positive way to be able to have the ability to generate valid Sepedi words. The best-performing model uses a batch size of 256, 128 embeddings, and 16 attention heads, achieving a score of 61.29%.

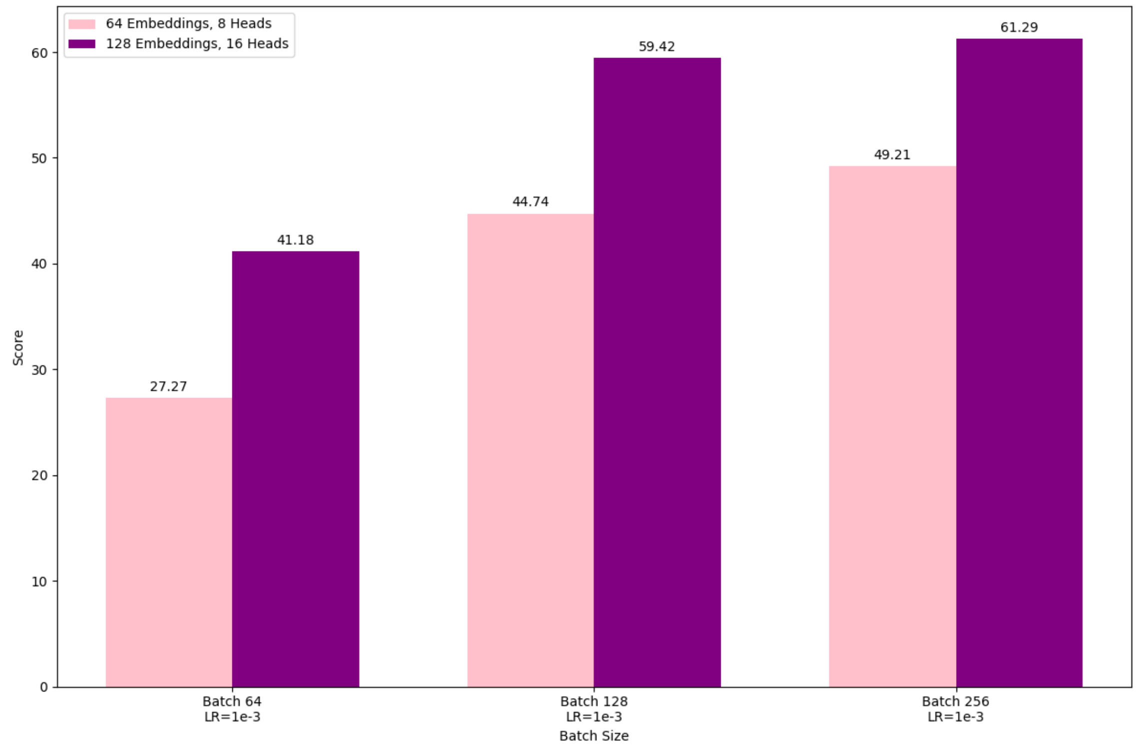
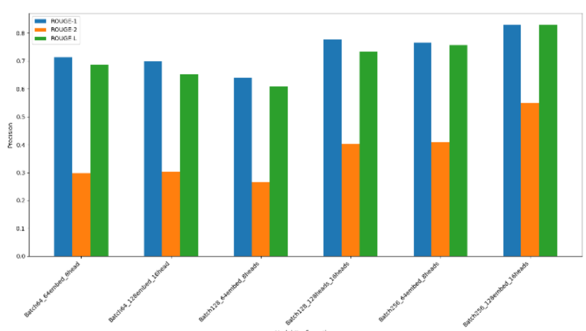


Figure 4.233 Percentage of Valid Sepedi words per Batch size

4.4. Text-generated evaluation using ROUGE Score

The accuracy and validity of the generated text were evaluated using the ROUGE score. The ROUGE score focuses more on semantic relevance, which was the primary requirement of this study, rather than the syntactic correctness of the language. For under-resourced languages such as Sepedi, it is not ideal to use evaluation metrics like BLEU or METEOR, as they emphasize the grammatical correctness of the generated text. For languages with agglutinative and morphologically rich structures, focusing solely on grammatical correctness may not accurately reflect meaningful text generation. ROUGE aligns well with the focus of this study. ROUGE is helpful for under-resourced languages because it checks how much important content is included and compares it to the original text.

Fig 4.24 to Fig 4.26 show the ROUGE results of precision, recall, and F-score, respectively. The graphs indicate that the precision has a high value compared to Recall and F-score. The precision percentage for the model configured with batch size 256, 128 number of embeddings, and 16 attention heads depicts that 83% of the unigrams in the generated text are found in the test data. Looking at metric Recall results, it shows that 0.05% of the text in the test data that are found in the generated text data. This implies that generated text data is lacking more of the words found in the test data. Then looking at the F-score, 0.10% is a low percentage, which indicates a poor balance between precision and recall. Examining the well-balanced models, those configured with a batch size of 64, 128 embeddings, and 16 attention heads, as well as those with a batch size of 128, 64 embeddings, and 8 attention heads, demonstrate a good balance between precision and recall. This balance results in a higher F-score. However, the model configured with batch size 256, 128 embeddings, and 16 attention heads provided a high percentage of valid Sepedi words compared to other models. The overall results indicate that as much as there are more relevant generated words in the test data, the generated data lacks comprehensiveness.



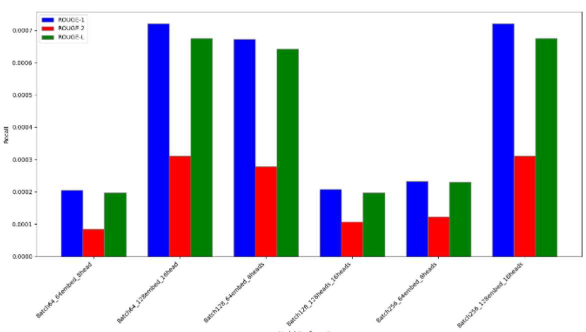


Figure 4.24 ROGUE Score Precision

Figure 4.25 ROGUE Score Recall

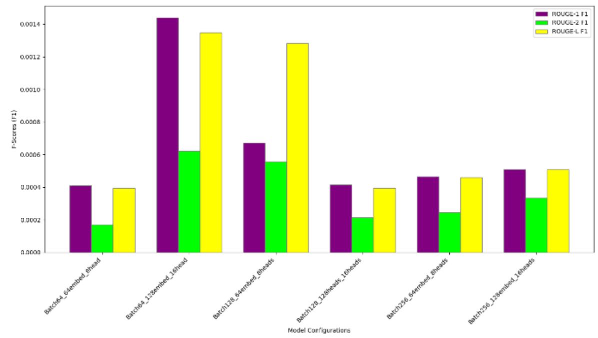


Figure 4.26 ROUGE F-Score results

4.5. Discussion

Chapter 4 discusses the two Sepedi text generation models, which are the Sepedi LSTM model and the Sepedi Transformer-based machine learning model. The results were obtained using different parameters that affect different aspects of training the model. The parameters range from batch sizes, number of embeddings, attention heads, different LSTM and Dense layers, input layer, weights, and learning rate. The LSTM model was trained using two sets of data while keeping the LSTM layer to 200.

The experiments for the Sepedi LSTM model were conducted using a different dataset. The experiments were conducted using a batch size of 64 to a batch size of 2048 using 50 epochs. Due to the system requirement to train the model using a large training dataset, optioning for 50 epochs was considered more than 100 epochs to run the experiments. The results are shown in Fig 4.6 to Fig 4.12. The number of epochs and LSTM weights batch size were changed per the experiment. The model that indicated better performance for the Sepedi LSTM model trained with 512 batch size using 50 epochs, as shown by Fig 4.6 and Fig 4.7. The validation accuracy at epoch 50 is 62%. The results indicate that the LSTM-Sepedi Attention model, which was trained with a batch size of 512 and LSTM weights of 200, can capture and generalize the language and generate coherent text. The results also suggest that even though the model obtained a moderate accuracy, it still faces some challenges in fully capturing and understanding the entire Sepedi language structure, such as its agglutination and rich morphology structure. While keeping into consideration the limitation, the model seems best suited for the generation of text for this study for the LSTM Sepedi Attention model.

Training of the GPT-Sepedi Transformer model involves training the model using a large dataset. The model was evaluated using Loss value. The learning rate was used as a key parameter during experiments. Different experiments were conducted using different model configurations, such as bath size, embeddings, and attention heads, while keeping the learning rate 1e^-3 constant. The loss results seem to be decreasing as the embeddings and attention heads increase. The model trained with batch size ‘256, 128 embeddings, and 16 attention heads outperformed all other models, getting the lowest loss results of 1.21, as shown in Fig 4.19. Additionally, the model indicated superior performance on the testing dataset, obtaining a test loss result of 1.21, which validates its generalisation ability and robustness in learning on unseen data. The combination of the chosen parameters helps with the optimization of the model’s capabilities in terms of learning and generalizing the text, particularly in the context of the Sepedi language’s complex linguistic structures. The lowest loss indicates that the model has a better chance of generating an output text that aligns with the expected output results. Thus, the combined parameters allow the model to generalise effectively and minimise errors.

The model was then used to generate the text, which was then compared to Sepedi vocabulary. The generated text provided a high percentage compared to Sepedi vocabulary, which indicates that the transformer model can generate valid Sepedi words. Fig 4.24 to Fig 4.26 show the ROUGE score. The ROUGE score was then used to evaluate the quality of the generated text. The percentage of rogue indicates that the model was able to generate 83% of the valid words found in the dataset. However, there were not enough words in the dataset that were found in the generated text. The F-score percentage shows that as much as there are many of the generated words within the reference data, the data has limited comprehensiveness. Thus, the data is not balanced and not enough to allow the model to generate accurate Sepedi language

Training both models had its own challenges that affected their performance. This includes the high computational resources required when training the models. The attention mechanism applied to the LSTM model is computationally demanding because it requires calculating weighted sums over all input states, which can become expensive, especially for long sequences. This makes LSTMs with attention mechanisms efficient when handling long sentences but necessitates an increase in computational resources. Similarly, transformer-based models also incur significant costs to train. Due to the nature of the data used, the models faced issues such as overfitting and stability. To reduce overfitting, dropout and early stopping were employed. Additionally, the use of Google Colab GPU A100 assisted in training both models efficiently, although it came with a considerable financial burden, requires hours and hours of training the models.

5. CHAPTER 5: CONCLUSION

5.1 Introduction

This chapter presents the outcome of the development of a text generation model for the Sepedi language using a transformer-based machine learning technique. This includes data exploration on the NCHLT Sepedi dataset, training an LSTM-attention text generation model, and training a Pretrained Transformer 2 text generation model. Evaluate the performance of the Generative Pretrained Transformer 2 text generation model for the Sepedi language and evaluate their performances using ROUGE metrics.

The Sepedi text generation model was developed using LSTM stacked with attention and transformer language model techniques. The results obtained when training using a transformer model with batch size 256, 64 number of embeddings, 8 number of attention heads, and learning rate 1e^-3 showed impressive outcomes. The results show that increasing the embeddings, attention heads, and learning rate while keeping the batch sizes constant at batch size 64 contributes to enhancing the model's performance. The batch size can either increase or decrease the model training speed, depending on the size. As batch sizes affect how the model trains, embedding assists with capturing the insight of the data to get their semantic meaning and the number of attention heads, each dealing with different aspects of the words within the sentence to learn the dependencies and relationship. This is to help the model to be less overfitting.

The model transformer provides the best results compared to the LSTM model. (Moila & Modipa, 2020). This indicates that the transformer model can learn Sepedi language data better than LSTM. Due to the results obtained, the model was then used to generate Sepedi words. The words generated were then compared to Sepedi's vocabulary. There was a high percentage of Sepedi words that were valid. By using ROUGE metrics, the results showed that the generated text is not that far from the original Sepedi. Further training is needed to generate enough text that is not only coherent but also comprehensive. This means the goal is to produce text that provides meaningful and complete information, rather than just text that makes sense in isolation. The overall results show that the GPT-Sepedi Transformer-based model performs better than the LSTM-Sepedi Attention model.

5.2 Scientific Contributions

Text generation aims to create a system that can automatically learn from input data and generate text that closely resembles human-written text. This technology has numerous applications, including chatbots, generating large datasets, and writing stories and novels. Currently, there is a lack of available datasets for under-resourced languages such as Sepedi. Most existing datasets for text generation are limited, leading to the need for large models that may overfit and perform poorly. Training in deep learning and transformer models requires a lot of resources. Training models with limited datasets can fail to understand the relationship between context and words, thereby producing subpar output text.

5.3 Future Work and Recommendations

While the LSTM-Sepedi Attention and GPT-Sepedi Transformer model has provided significant results in generating Sepedi text, other factors still need to be investigated further. These factors include the size of the dataset and the limitations in computational resources when training transformer models. Training LSTM models with an attention mechanism also presents its own set of challenges, like training transformer models. Our primary, NCHLT Sepedi dataset is small, and this may affect model training in terms of model generalisation. Future work should involve augmenting the NCHLT Sepedi dataset with other data from other sources to enhance the model performance.

Making use of pre-trained language models has shown state-of-the-art performance in text generation tasks. Training in deep learning and transformer models requires a lot of resources, such as the use of advanced architectures and hardware systems, high-performance systems that have high processing units and GPUs. To measure the performance of this model, in future work, research should consider incorporating other evaluation metrics such as METEOR, BLEU, and human evaluation to provide a more comprehensive assessment of the generated text's quality than using only ROUGE. By deploying the text generation model in various real-world applications, such as automated news article generation, educational tools for Sepedi language learners, and digital assistants. Future work should focus on adapting the model for these specific applications, ensuring robustness and reliability in practical settings. Transformer models require high-performance systems, this has been a problem.

5.4 Research Summary

The aim of the study was introduced in Chapter 1. The study aimed to develop a text generation model for the Sepedi language using a transformer-based machine technique. Data exploration was performed on the National Centre for Human Language Technology (NCHLT) Sepedi dataset. The LSTM-Sepedi Attention text generation model and Generative Pretrained Transformer (GPT)-Sepedi Transformer were then trained. The performance was evaluated based on loss and accuracy values. The validity of the generated was evaluated using the ROGUE score. To fulfill the aim of the study, the following objectives were outlined:

- Objective 1: A secondary Sepedi NCHLT Corpus was acquired from the NCHLT system. The Python programming language was used to perform the preprocessing of data (Section 3.2). Objective 2: Open-source TensorFlow Python libraries were used during the process of training the model. Different parameters, ranging from batch size, embeddings, LSTM weights, and attention heads, were defined as part of training purposes (Section 3.3.2). Objective 3: For the training of the GPT-Sepedi Transformer text generation model, Pytorch libraries were used during the training stages. The parameters such as batch size, learning rate, embedding sizes, and number of attention heads were used during the training. This includes defining the GPT language model (3.3.3). Objective 4: The loss and accuracy functions were used as a model evaluation measure (Section 3.4). The LSTM-Sepedi Attention model was trained using different parameters; the observed results indicate that the LSTM-Sepedi Attention model provides better results compared to LSTM models (Section 4.2). (GPT)-The Sepedi Transformer text generation model for the Sepedi language was evaluated using the loss function (Section 3.4). The results indicated that training the model with large embedding and a greater number of attention heads seems to have more effect on the performance of the model. This was shown by the low loss value and percentage of valid generated Sepedi words (Section 4.3).

- Objective 5: ROUGE Score was used to evaluate the quality of the generated words (Section 3.4). The GPT-Sepedi Transformer model was selected to be the one that is used to generate the text due to its low loss value. The ROUGE results for the generated text indicated a high percentage of precision, which indicates that the model was able to generate valid words (Section 4.4)

Bibliography

Adelani, D. I., Ruiter, D., Alabi, J. O., Adebonojo, D., Ayeni, A., Adeyemi, M., . . . España-Bonet, C. (2021). MENYO-20k: A Multi-domain English-Yorùbá Corpus for Machine Translation and Domain Adaptation. *CoRR, abs/2103.08647*. doi:https://arxiv.org/abs/2103.08647

Agarwal, S., & Arora, S. (2007). Context Based Word Prediction for Texting

Language. In *Large-Scale Semantic Access to Content (Text, Image, Video, and Sound* (pp. 360–368). Paris, FRA: LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Allwood, J., Hammarstro, H., Hendrikse, A., Ngcobo, M. N., Nomdebevana, N., Pretorius, L., & van der Merwe, M. (2010). Work on Spoken (Multimodal) Language Corpora in South Africa. *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10).* Valletta, Malta: European Language Resources Association (ELRA). Retrieved from http://www.lrec-conf.org/proceedings/lrec2010/pdf/438_Paper.pdf

Alsmadi, I., Aljaafari, N., Nazzal, M., Alhamed, S., Sawalmeh, A. H., Vizcarra, C. P., . . . Al-Humam, A. (2022). Adversarial Machine Learning in Text Processing: {A} Literature Survey. *IEEE Access, 10*, 17043-17077. doi:10.1109/ACCESS.2022.3146405

Anandarajan, M., Hill, C., & Nolan, T. (2019). Text Preprocessing. In *Practical Text Analytics: Maximizing the Value of Text Data* (pp. 45 - 49). Springer International Publishing. doi:10.1007/978-3-319-95663-3_4

Ashirwadam, J. (2014). Communication Research Methods Methods of Data Analysis. *Tamilnadu Theological Seminary*, 1 - 6.

Badenhorst, J., & De Wet, F. (2022). NCHLT Auxiliary speech data for ASR technology development in South Africa. *Data in Brief, 41*. doi:https://doi.org/10.1016/j.dib.2022.107860

Baerman, M., Brown, D. P., & Greville, C. (2017). *Morphological Complexity.* Cambridge University Press. doi:10.1017/9781316343074

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In Y. Bengio, & Y. LeCun (Ed.), *3rd International Conference on Learning Representations* (pp. 1 - 15). San Diego, CA, USA: corr. Retrieved from http://arxiv.org/abs/1409.0473

Bansal, A., Sharma, R., & Kathuria, M. (2022). A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications. *Association for Computing Machinery, 54*(10), 1-29. doi:https://doi.org/10.1145/3502287

Barman, P. P., & Boruah, A. (2018). A RNN-based Approach for next word prediction in Assamese Phonetic Transcription. *Procedia Computer Science, 143*, 117 - 123. doi:https://doi.org/10.1016/j.procs.2018.10.359

Bin, G., Hao, W., Yasan, D., Wei, W., Shaoyang, H., Yueqi, S., & Zhiwen, Y. (2021). Conditional Text Generation for Harmonious Human-Machine Interaction. *ACM Trans. Intell. Syst. Technol., 12*, 50. doi:10.1145/3439816

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., . . . Amodei, D. (2020). Language Models Are Few-Shot Learners. *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems. 159*, pp. 1877–1901. Red Hook, NY, USA: Curran Associates Inc.

Budzianowski, P., & Vulić, I. (2019). Hello, It's GPT-2 - How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems. *Proceedings of the 3rd Workshop on Neural Generation and Translation* (pp. 15 - 22). Hong Kong: Association for Computational Linguistics. doi:10.18653/v1/D19-5602

Byamugisha, J., Keet, M. C., & DeRenzi, B. (2017). Toward an NLG System for Bantu languages: first steps with Runyankore (demo). *Proceedings of the 10th International Conference on Natural Language Generation* (pp. 154 - 155). Santiago de Compostela, Spain: Association for Computational Linguistics. doi:10.18653/v1/W17-3523

Callison-Burch, C., Osborne, M., & Koehn, P. (2006). Re-evaluating the Role of {B}leu in Machine Translation Research. *11th Conference of the {E}uropean Chapter of the Association for Computational Linguistics* (pp. 249 - 256). Trento, Italy: Association for Computational Linguistics. Retrieved from https://aclanthology.org/E06-1032

Chai, C. P. (2023). Comparison of text preprocessing methods. *Natural Language Engineering, 29*, 509 - 553. doi:10.1017/S1351324922000213

Cheng, J., Dong, L., & Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading. *Proceedings of the 2016 Conference on Empirical*

Methods in Natural Language Processing (pp. 51-561). Austin, Texas: Association for Computational Linguistics. doi:10.18653/v1/D16-1053

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724 - 1734). Doha, Qatar: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Doi: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (pp. 103 – 111). Doha, Qatar: Association for Computational Linguistics. doi:10.3115/v1/W14-4012

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., . . . Stoyanov, V. (2020). Unsupervised Cross-lingual Representation Learning at Scale. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440 - 8451). Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.747

Cunliffe, D., Vlachidis, A., Williams, D., & Tudhope, D. (2022). Natural language processing for under-resourced languages: Developing a Welsh natural language toolkit. *Computer Speech & Language, 72*, 1 - 20. doi:https://doi.org/10.1016/j.csl.2021.101311

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., & Salakhutdinov, R. (2019). Transformer-{XL}: Attentive Language Models beyond a Fixed-Length Context. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 2978- 2988). Florence, Italy: Association for Computational Linguistics". doi:10.18653/v1/P19-1285

Damien, B. (2023, February 01). *The ChatGPT models family*. Retrieved from

NEWSLETTER The AiEdge: https://newsletter.theaiedge.io/p/the-chatgpt-models-family

Dar, G., Geva, M., Gupta, A., & Berant, J. (2023). Analyzing Transformers in Embedding Space. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Abs/2209.02535*, pp. 16124 - 16170. Toronto, Canada: Association for Computational Linguistics. doi:10.18653/v1/2023.acl-long.893

Dehouche, N. (2021). Plagiarism in the Age of Massive Generative Pre-Trained Transformers (Gpt-3). *Ethics in Science and Environmental Politics, 21*, 17-23. doi:10.3354/esep00195

Denner, V. S., Sergio, P., & Renio, M. S. (2018). Robustness of sentence length measures in written texts. *Physica A: Statistical Mechanics and its Applications, 506*(https://doi.org/10.1016/j.physa.2018.04.104), 749 - 754. doi:https://doi.org/10.1016/j.physa.2018.04.104

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171 - 4186). Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1423

Dhall, I., Vashisth, S., & Saraswat, S. (2020). Text Generation Using Long Short-Term Memory Networks. *Micro-Electronics and Telecommunication Engineering. 106*, pp. 64 - 657. Singapore: Springer. doi:doi.org/10.1007/978-981-15-2329-8_66

Doval , Y., & Gómez-Rodríguez, C. (2019). Comparing Neural- and N-gram-based Language Models for Word Segmentation. *J. Assoc. Inf. Sci. Technol., 70*, 187 – 197. doi:10.1002/asi.24082

Du, H., Xing, W., & Pei, B. (2021). Automatic text generation using deep learning: providing large-scale support for online learning communities. *Interactive Learning Environments, 0*, 1-16. doi:10.1080/10494820.2021.1993932

Eiselen, R., & Puttkammer, M. (2014). Reykjavik, Iceland. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)* (pp. 3698-3703). European Language Resources Association (ELRA). Retrieved from http://www.lrec-conf.org/proceedings/lrec2014/pdf/1151_Paper.pdf

Faass, G., Heid, U., Taljard, E., & Prinsloo, D. (2009). Part-of-Speech Tagging of Northern Sotho: Disambiguating Polysemous Function Words. *Proceedings of the First Workshop on Language Technologies for African Languages* (pp. 38–45). Athens, Greece: Association for Computational Linguistics.

Fatima, N., Imran, A. S., Kastrati, Z., Daudpota, S. M., & Soomro, A. (2022). A Systematic Literature Review on Text Generation Using Deep Neural Network Models. *IEEE Access, 10*, 53490–53503. doi:10.1109/ACCESS.2022.3174108

Feldman, R., & Sanger, J. (2006). Text Mining Preprocessing Techniques. In *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data* (pp. 57–63). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511546914.004

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press. Retrieved from http://www.deeplearningbook.org

Grover, A. S., van Huyssteen, G. B., & Pretorius, M. W. (2010). The South African Human Language Technologies Audit. *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10).* Valletta, Malta: European Language Resources Association (ELRA). Retrieved from http://www.lrec-conf.org/proceedings/lrec2010/pdf/454_Paper.pdf

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., & Wang, J. (2018). Long Text Generation via Adversarial Training with Leaked Information. *Proceedings of the AAAI Conference on Artificial Intelligence* (p. 8). New Orleans, Louisiana, USA: AAAI Press. doi:doi.org/10.1609/aaai.v32i1.11957

Gupta, S., & Gupta, S. K. (2019). Abstract Summarization: An Overview of the

State of the Art. *Expert Systems with Applications, 121*, 49–65. doi:10.1016/j.eswa.2018.12.011

Gurusamy, V., & Kannan, S. (2014). Preprocessing Techniques for Text Mining, (pp. 1 - 6).

Hajdik, V., Buys, J., Goodman, M. W., & Bender, E. M. (2019). Neural Text Generation from Rich Semantic Representations. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1235

Hajdik, V., Buys, J., Goodman, M. W., & Bender, E. M. (2019). Neural Text Generation from Rich Semantic Representations. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Volume 1 (Long and Short Papers)*, pp. 2259-2266. Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1235

Hänig, C., Schierle, M., & Trabold, D. (2010). Comparison of Structured vs. Unstructured Data for Industrial Quality Analysis. *Intelligent Automation and Systems Engineering*, (pp. 20 - 22). Retrieved from https://api.semanticscholar.org/CorpusID:16766210

Hedderich, M. A., Lange, L., Adel, H., Otgen, J., & Klakow, D. (2021). A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 2545- 2568). Association for Computational Linguistics. doi:10.18653/v1/2021.naacl-main.201

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science, 349*, 261-266.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*, 1735- 1780.

Hong, M., Wang, M., Luo, L., Tan, X., & Zhang, D. (2018). Combining Gated Recurrent Unit and Attention Pooling for Sentimental. *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence* (pp. 99 – 104). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3297156.3297267

Iqbal, T., & Qureshi, S. (2022). The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences, 34*, 2515-2528. doi:https://doi.org/10.1016/j.jksuci.2020.04.001

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An Empirical Exploration of Recurrent Network Architectures. *Proceedings of the 32nd International Conference on International Conference on Machine Learning. Volume 37*, pp. 2342 – 2350. JMLR.org}.

Jurafsky, D., & Martin, J. H. (2022). Speech and Language Processing. 3rd. https://web.stanford.edu/.

Kaddari, Z., Mellah, Y., Berrich, J., Belkasmi, M. G., & Bouchentouf, T. (2021). Natural Language Processing: Challenges and Future Directions. In *Artificial Intelligence and Industrial Applications* (pp. 236- 246). Cham: Springer International Publishing.

Kanwal, J., Smith, K., Culbertson, J., & Kirby, S. (2017). Zipf's Law of Abbreviation and the Principle of Least Effort: Language users optimise a miniature lexicon for efficient communication. *Cognition, 165*, 45 - 52. doi:https://doi.org/10.1016/j.cognition.2017.05.001

Klaus, G., Rupesh, S. K., Jan , K., Bas, S. R., & Jürgen, S. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems, 28*, 2222-2232. doi:10.1109/TNNLS.2016.2582924

Kosch, I. M. (2006). *Topics in Morphology in the African Language Context.* Unisa Press. doi:https://books.google.co.za/books?id=e26AoEvgttcC

Lewis, M. L., & Frank, M. C. (2016). The length of words reflects their conceptual complexity. *Cognition, 153*, 182 - 195. doi:https://doi.org/10.1016/j.cognition.2016.04.003

Li, J., Tianyi, T., Zhao, W. X., & Wen, J.-R. (2021). Pretrained Language Model for Text Generation: A Survey. In Z.-H. Zhou (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)* (pp. 4492 - 4499). International Joint Conferences on Artificial Intelligence Organization. doi:10.24963/ijcai.2021/612

Li, Z., & Jian-Tao, S. (2009). Text Generation. In Z. Li, & S. Jian-Tao, *Encyclopedia of Database Systems* (pp. 3048-3051). Springer US. doi:10.1007/978-0-387-39940-9_416

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *Annual Meeting of the Association for Computational Linguistics* (pp. 74 - 81). Barcelona, Spain: Association for Computational Linguistics. Retrieved from https://aclanthology.org/W04-1013

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR, abs/1506.00019*, 1 - 38.

Mahlangu, P. T. (2018). Challenges faced by Sepedi language in education: A case study of selected schools in Limpopo Province, South Africa. *Journal of Multilingual Education Research, 8*(19), 23-38.

Malope, M. (2006). Morphology and word formation in Sepedi: A study of a Bantu language. *Lexikos, 16*, 170-189.

Mann, W. (1982). Text Generation. *Comput. Linguist., 8*(2), 62–69.

Marivate, V., Sefara, T., Chabalala, V., Makhaya, K., Mokgonyane, T., Mokoena, R., & Modupe, A. (2020). Investigating an Approach for Low Resource Language Dataset Creation, Curation and Classification: Setswana and Sepedi. *Proceedings of the first workshop on Resources for African Indigenous Languages"* (pp. 15-20). Marseille, France: European Language Resources Association (ELRA). Retrieved from https://aclanthology.org/2020.rail-1.3

Martinus, L., & Abbott, J. Z. (2019). A Focus on Neural Machine Translation for African Languages. *CoRR, abs/1906.05685*. Retrieved from http://arxiv.org/abs/1906.05685

Mkhize, D., & Balfour, R. (2017). Language rights in education in South Africa. *South African Journal of Higher Education, 31*(6), 133-150. doi:10.28535/31-6-1633

Mngomezulu, J. R., Tonsing, K. M., Dada, S., & Bokaba, N. B. (2019). Determining a Zulu core vocabulary for children who use augmentative and alternative communication. *Augmentative and Alternative Communication, 35*, 274 - 284.

Moila, M. M., & Modipa, T. I. (2020). The Development of a Sepedi Text Generation Model Using Long-Short Term Memory. *Proceedings of the 2nd International Conference on Intelligent and Innovative Computing Applications.* Plaine Magnien, Mauritius: Association for Computing Machinery}.

Molepo, L. N. (2014). *The Designation of a Language Naming: Sepedi / Sesotho SA LEBOA.* University of Limpopo.

Mothapo, R., Tönsing, K., & Morwane, R. (2021). Determining the Core Vocabulary used by Sepedi-speaking Children During Regular. *Int J Speech Lang Pathol, 23*(3), 295 - 304. doi:10.1080/17549507.2020.1821774

New, L., Ferrand, C., Pallier , C., & brysbaert, M. (2006). Reexamining the word length effect in visual word recognition: New evidence from the English Lexicon Project. *Psychonomic Bulletin & Review, 13*, 45 - 52. doi:10.3758/BF03193811

Orife, I., Kreutzer, J., Sibanda, B. K., Whitenack, D., Siminyu, K., Martinus, L., . . . Bashir, A. (2020). Masakhane - Machine Translation For Africa}. *CoRR, abs/2003.11529*. doi:https://arxiv.org/abs/2003.11529}

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311 - 318). Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. doi:10.3115/1073083.1073135

Pretorius, R., Berg, A., Pretorius, L., & Viljoen, B. (2009). Setswana Tokenisation and Computational Verb Morphology: Facing the Challenge of a Disjunctive Orthography. *Proceedings of the First Workshop on Language Technologies for African Languages* (pp. 66-\-73). Athens, Greece: Association for Computational Linguistics. Retrieved from https://aclanthology.org/W09-0710

Radford, A., & Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training. Retrieved from https://api.semanticscholar.org/CorpusID:49313245

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog, 1*, 9.

Rai, A., & Borah, S. (2021). Study of Various Methods for Tokenization. In J. K. Mandal, S. Mukhopadhyay, & A. Roy (Ed.), *Applications of Internet of Things* (pp. 193 - 200). Singapore: Springer Singapore. doi:https://doi.org/10.1007/978-981-15-6198-6_18

Ramalepe , P. S., Modipa , I. T., & Davel, H. M. (2025). Pre-training a Transformer-Based Generative Model Using a Small Sepedi Dataset. *arXiv*.

Ramzan, T., Muhammad, K. H., Shaeela, A., & Fakeeha, F. (2016). Text Mining: Techniques, Applications and Issues. *International Journal of Advanced Computer Science and Applications, 7*, 11. doi:10.14569/IJACSA.2016.071153

Reshamwala, A., Mishra, D., & Pawar, P. (2013). REVIEW ON NATURAL LANGUAGE PROCESSING. *IRACST – Engineering Science and Technology: An International Journal (ESTIJ), 3*, 113-116.

Rosari, G., & Noever, A. D. (2023). Grading Conversational Response of Chatbots. *ArXiv*.

Salehinejad, H., Baarbe, J., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent Advances in Recurrent Neural Networks. *ArXiv, abs/1801.01078*.

Santhanam, S. P. (2020). Context-based text generation using LSTM networks. *ArXiv, abs/2005.00048*. Retrieved from https://api.semanticscholar.org/CorpusID:69429822

Schramowski, P., Turan, C., Andersen, N., Rothkopf, C. A., & Kersting, K. (2021). Language Models Have a Moral Dimension. *CoRR, abs/2103.11790*. Retrieved from https://arxiv.org/abs/2103.11790

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena, 404*, 132 - 306. doi:https://doi.org/10.1016/j.physd.2019.132306

Shikali, C. S., & Mokhosi, R. (2020). Enhancing African low-resource languages: Swahili data for language modeling. *Data in Brief, 31*, 1 - 7. doi:https://doi.org/10.1016/j.dib.2020.105951

Sidik, N. (2019). The language in Education Policy in South Africa: A gap between policy and efficacy. *Proceedings of the Eleventh Conference on Applied Linguistics (CONAPLIN 2018).* ATLANTIS PRESS. doi:10.2991/conaplin-18.2019.321

Skutnabb, K. T. (2000). *Linguistic Genocide in Education, Or Worldwide Diversity and Human Rights?* Taylor & Francis Group. doi:https://books.google.co.za/books?id=3ZqDQgAACAAJ

Sonkar, S., Waters, A., & Baraniuk, R. (2020). Attention Word Embedding. *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 6894 - 6902). Barcelona, Spain (Online): International Committee on Computational Linguistics. doi:10.18653/v1/2020.coling-main.608

Stent, A., Marge, M., & Singhai, M. (2005). Evaluating Evaluation Methods for Generation in the Presence of Variation. In A. Gelbukh, *Computational Linguistics and Intelligent Text Processing* (pp. 341-351). Springer Berlin Heidelberg.

Su, S.-Y., Huang, C.-W., & Chen, Y.-N. (2020). Towards Unsupervised Language Understanding and Generation by Joint Dual Learning. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 671-680). Online: Association for Computational Linguistics.

doi:10.18653/v1/2020.acl-main.63

Su, Y., & Kuo, C. J. (2019). On Extended Long Short-Term Memory and Dependent Bidirectional Recurrent Neural Network. *Neurocomput, 356*, 151–161. doi:10.1016/j.neucom.2019.04.044

Subramanian, S., Rajeswar, S., Dutil, F., Pal, C., & Courville, A. (2017). Adversarial Generation of Natural Language". *Association for Computational Linguistics* (pp. 241 - 251). Association for Computational Linguistics. doi:10.18653/v1/W17-2629

Suman , R., & Andreas, S. (2016). A comparative study of recurrent neural network models for lexical domain classification. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6075–6079). Shanghai, China: IEEE Press. doi:10.1109/ICASSP.2016.7472844

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (pp. 3104 – 3112). Montreal, Canada: MIT Press.

Taljard, E., & Bosch, S. E. (2006). A Comparison of Approaches to Word Class Tagging: Disjunctively vs. Conjunctively Written Bantu Languages. *Nordic Journal of African Studies, 15*, 428-442.

Taylor, J. R. (2015). *The Oxford Handbook of the Word.* Oxford University Press. doi:10.1093/oxfordhb/9780199641604.001.0001

Topal, O. M., Bas, A., & van Heerden, I. (2021). Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet. *CoRR, abs/2102.08036*. Retrieved from https://arxiv.org/abs/2102.08036

Turner, N., & Wildsmith-Cromarty, R. (2014). Challenges to the implementation of bilingual/multilingual language policies at tertiary institutions in South Africa (1995â€“2012). *Language Matters, 45*(4), 295-312. doi:10.1080/10228195.2014.961525

Tüske, Z., Schlüter, R., & Ney, H. (2018). Investigation on LSTM Recurrent N-gram Language Models for Speech Recognition. *Interspeech.* Computer Science.

Retrieved from https://api.semanticscholar.org/CorpusID:52191115

Van Wyk , E. B. (1995). Linguistic assumptions and lexicographical traditions in the African Languages. *Lexikos, 5*(5), 82-96. doi:10.10520/AJA16844904_292

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17.* Long Beach, California, USA: Curran Associates Inc.

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., & Weston, J. (2019). Neural Text Generation With Unlikelihood Training. *CoRR, 1908.04319*. doi:https://dblp.org/rec/journals/corr/abs-1908-04319.bib

Xia, C., Nguyen, H., Zhang, J., & Yu, P. S. (2020). CG-BERT: Conditional Text Generation with {BERT} for Generalized Few-shot Intent Detection. *CoRR, 2004.01881*.

Xie, Z. (2017). Neural Text Generation: A Practical Guide. *CoRR, abs/1711.09534}*. Retrieved from http://arxiv.org/abs/1711.09534

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent Trends in Deep Learning-Based Natural Language Processing. *IEEE Comput. Intell. Mag., 3*, 55 - 75. doi:10.1109/MCI.2018.2840738

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating Text Generation with BERT. *International Conference on Learning Representations.* Retrieved from https://openreview.net/forum?id=SkeHuCVFDr

Zipf, G. K. (1935). *The psycho-biology of language.* Houghton, Mifflin.