Jimma University
Jimma Institute of Technology
Faculity of Electrical and Computer Engineering

Constructing a Model for Kafi-Noonoo Word Sequence Prediction Using Transfer Learning Approach

A Thesis Submitted to the School of Graduate Studies of Jimma University in Partial Fulfilment of the Requirement for the Degree of Master of Science in Computer Engineering

Presented by:

TEGENE GAREDEW ABATE

Jimma, Oromia, Ethiopia, March 2022.

Jimma University
Jimma Institute of Technology
Faculity of Electrical and Computer Engineering

Constructing a Model for Kafi-Noonoo Word Sequence Prediction Using Transfer Learning Approach

Thesis carried out by:

TEGENE GAREDEW ABATE

Advisor: Dr. Kinde Anlay(Assistant Professor)
Co-Advisor: Mr. Fetulhak Abdurahman(MSc.)

March 2022

Jimma, Ethiopia

Declaration

I declare that the work described in this thesis is entirely my own. No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or institute. Any help or source information, which has been availed in the thesis, has been duly acknowledged

Researcher: Tegene Garedew Signature:_____ Date:_____

This thesis has been submitted for examination with our approval as the university advisers.

Main Advisor: Dr.Kinde Anlay(Assistant Professor) Signature:_____ Date:_____

Co-advisor: Mr. Fetulhak Abdurhman(MSc.) Signature:_____ Date:_____

The thesis had been examined by:

External Examiner: Dr. Million Meshesha Signature:  Date:__March 15, 2022__

Internal Examiner: Mr. Kris Calpotura Signature: _____ Date:_____

Chair-person: Dr. Sirinivasan Trs. Signature: _____ Date:_____

JIMMA UNIVERSITY
JIMMA INSTITUTE OF TECHNOLOGY
FACULITY OF ELECTRICAL AND COMPUTER ENGINEERING

(Signatures)

.....
TEGENE GAREDEW ABATE

Abstract

This study presents word sequence prediction Language model for Kafi-Nono words. Text generation, in particular, next-word prediction, is convenient for users because it helps to type without errors and faster. Therefore, a personalized text prediction system is a vital analysis topic for all languages, primarily for Kafi-Nono, because of limited support for the Kafa language tools. Language model (LM) gives probability of how likely a sequence of words might appear in a particular order in a sentence, and they are an essential part of word prediction and natural language processing (NLP) systems. Language models have significantly advanced after the invention of Neural Network Language Models (NNLMs) called Transformers. Transformers have become the state-of-the-art language modeling tools for many NLP tasks because of their superior performance compared to N-gram models. Similarly, word prediction systems have improved at a considerable pace over the past decade. Though Kafi-Nono is spoken by a significant number of people, there is no word prediction system developed sofar for the language; this thesis is a first attempt to develop a word prediction language model for Kafi-Nono. We have applied cross-lingual transfer technique on the latest deep learning transformer model to develop the system. The main objective of this study is to develop word prediction language model that can predict next word/phrase for Kafi-Nono. The corpus was collected from books, news, cultural documents, and history of the language speaking societies for the sake of this study only. In order to develop the model, we have used unsupervised machine learning method called transfer learning. The transformer model used in this work is Generative Predictive Transformer(GPT2), which is made of 12 layers of NN, 768 Hidden units per layer, 768 Code context length, 768 Embedding dimension and 12 Attention heads. Transfer learning helped to overcome the problem of data scarcity for this language and enabled us to adapt the power of neural networks to get reasonable result with less effort. The idea behind the approach is to overcome the problem of a low-resources of the language, and handle the vast morphological inflection behaviour and scarcity of training data for Kafi-Nono by using unsupervised approach. This makes our approach effective for prediction problems where there is lack of resources for the language. For our experiment we have divided the dataset in to training and evaluation parts each with a dataset of 80% and 20% respectively.A separate testing data is also prepared to evaluate the final model. To evaluate the performance of our model, we have implemented two types of evaluation metrics; human(extrinsic) evaluation and automatic(intrinsic) evaluation called perplexity. The result claimed that our model yields an accuracy of 89% in human evaluation and 4.7 in perplexity. The achieved result was encouraging;. However; we trained our model with a mix of data from all dialects and also tonal features are not treated in our data.So, handling this two problems in the data can bring better result.

Keywords

Language Modeling,Word Predictions,Transfer Learning,Transformers,Kafi-nono

Acknowledgements

First of all, all praise be to the Almighty God, for giving me the blessing, the strength, the chance and endurance to complete this study. Next, I would like to express my sincere gratitude to my advisor Dr. Kinde Anlay for his time, generous guidance, patience and encouragement throughout the whole thesis work, from which I have learned a lot regarding my title. I would also like to acknowledge Mr. Fetulhak Abdurrehman for his suggestive and constructive comments which strengthen this research work. He has been working day in day out with me in the all over work. To my family who have been with me in every ups and downs during all my study period. I extend my thanks to the colleagues for providing me the necessary data that is important for training and testing of the prototype developed. Finally, my gratitude goes to all my classmates for the discussion we have and for the ideas we share which was very helpful for the successful completion of this work.

TEGENE GAREDEW ABATE

Contents

List of Figures

List of Tables

Abbreviations

AAC Augmentative and Alternative Communication

ACL Association for Computational Linguistics

AI Artificial Intelligence

BERT Bidirectional Encoder Representations

BiLSTM Bi-directional Long-Short Memory

BPE Byte-Pair Encoding

FFNNs Feed Forward Neural Networks

GPT Generative Pre-Training

HMM Hidden Markov Model

KN Kafi-Noonoo

KNGPT2 Kafinoonoo Generative Pretraining

KSS Key Stroke Per Second

LMs Language Models

LSTM Long-Short Term Memory

NLP Natural Language Processing

NMT Neural Machine Translation

OOV Out-of-vocabulary

POST Parts-of- Speech Tag

RNN Recurrent Neural Network

SOV Subject-Object-Verb

TPU Tensor Processing Unit

WP Word Prediction

XLMR Cross-Lingual Model - Representation

Chapter One : Introduction

1.1 Background

Language is one of the distinct characters of human being and is an important component of our lives. In written form it serves as a tool to transfer knowledge from one generation to the next. In spoken form it serves as a primary means of communication with others to accomplish our day-to-day activity. Natural Language Processing (NLP), therefore, is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to carry out useful tasks [3]. Some of these useful tasks are word prediction, Automatic Speech Recognition (ASR), Machine Translation (MT), optical character recognition (OCR), question answering (QA) and so on.

Ethiopia is a linguistic diverse country with more than 80 spoken languages. The development of human language technologies (HLTs) for these languages can contribute meaningfully for multilingualism and language development.

So far considerable effort from government universities, and some private educational institutions and individuals have been made to develop NLP resources for Ethiopian languages, but most of the languages are still considered low-resource for being with little data that can be used to develop NLP applications and technologies. The continuing development and advance of core technologies is a key step towards getting the goals set by the government policies, while also being a precondition for downstream NLP systems, such as machine translation. Therefore, it is vital that these development efforts are in agreement with international best practices and trends.

Language Models(LM) give a probability of how likely a sequence of words might appear in a certain order in a sentence. LMs are applied to a wide range of modern natural language processing (NLP) applications. LMs are important for automatic speech recognition (ASR), machine translation, and spelling correction systems, to name a few. LMs form a significant part of NLP applications because many tasks depend on the quality of the LMs used. The quality of LMs is determined with perplexity (PPL), which tells how well they can predict the correct sequence of words[4].

The task of next word prediction is finding the most possible next word(s) based on words around it. This is the archetypal prediction problem in NLP. Most NLP tasks require

prediction of most possible word, part of speech tag (POST) [5] or any other token, when given the context. For example, word-sense disambiguation, speech recognition, accent restoration, word choice selection, identifying discourse markers and context-sensitive spelling correction. Most methods implemented to these problems are based on n-gram and the like.

1.2 The Trends of Language Technologies

Approaches for modelling language are broadly classified as statistical and neural models. The models have their own advantages and shortcomings. For example, statistical models such as N-gram are limited only on the available dictionary words, that mean N-grams cannot produce new words. And though neural models are far better than statistical methods, they require a vast sum of training data making it difficult for low-resource languages.

The trend of language technologies has shifted from rule-based systems, which were popular until the 1990s [6, 7]], to data- driven, statistical, or supervised machine learning based methods like as Hidden Markov Models (HMMs), memory-based learning and decision trees [7, 8],6], and over the last decade the trend shifted again from machine learning to neural networks [8, 9].

As of 2011, [10] showed that neural networks could overtake other machine learning methods available at that time on POST, NER, phrase chunking, and semantic role labelling tasks. Later in 2013, Neural Machine Translation (NMT) [11, 12, 13] has proven itself as the new state-of-the- art machine translation. NMT systems have achieved improvements of up to 20 BLEU points [14] over statistical language modeling systems [15], which is currently in use by technology giants like Google [15] and Facebook [16]. Recently, deep learning based pre-trained transformer models have proved outstanding results in numerous language tasks.

Most recently, transfering the pre-trained transformer models like ELMo (Embedding from Language Models) [17], GPT (Generative Pre-Training) [18], GPT-2 [11] and BERT (Bidirectional Encoder Representations from Transformers) to low resource languages become the dominant practices for state-of-the-art NLP results. GPT2 is the successor of GPT. Although both GPT-2 and BERT are capable of text generation, Wang and Cho [7] shown that GPT-2 based generations are better in quality. Actually, the powerfulness of GPT-2 is claimed to be so high that it can be the risk of malicious or dangerous use. For this reason, OpenAI did not release the largest model to give more time to discuss about

the concerns.

1.3 Transfer learning

Though Neural language models require a vast sum of training data making it difficult for low-resource languages, they often outperform traditional language modeling methods. To tackle the problem of Neural language modeling systems shortfall for low-resource languages, transfer learning has been proven to be an effective approach.

Transfer learning is using knowledge from a source model trained with high resource language (such as English) to improve the performance on a target low resource language [19]. A low-resource language is defined as one for which there are few, if any, documenting resources such as corpus, grammars, or written texts. For communities who speak rare and under-served languages, these kinds of materials are important for preserving and promoting their culture, linguistic heritage, and identity.

In this thesis, we have used the state-of-the-art (SOTA) transformer model by applying transfer learning technique which best fit for morphologically rich and low resource languages, since Kafi-Nono is morphologically complex.

Generally, we have observed that how a few training steps are needed for our model to predict the first text that looks like a Kafi-Nono text and the coherence and complication of the generated words are very remarkable, although not all text is equally generated in terms of quality.

1.4 Statement of the Problem

The field of Artificial Intelligence (AI) has changed many aspects of human living, specially the human communication. The foundation for communication is language. This day there are a number of AI based communication applications that are used on a daily basis. For example, we practice AI based navigation systems like Google Maps and Waze to find the shortest way to our destination; we use AI assisted search engines like Google and Bing to search for relevant information; and AI based personal assistant systems like Siri and Alexa to organize our daily schedules, among many other things [18]. And all of these applications are based on language technology.

NLP primarily improves the development of languages through digital resources development and efficient language usage in digital communication. In the modern world of today the significance of computers and handheld devices is immense. Texts are the primary communication mechanism in the modern digital devices.

Unfortunately, a significant proportion of people all over the world have not benefited from these AI technologies, primarily because of lack of access to these technologies. In particular, this lack of access to AI technologies is very common to low-resource communities, communities which are suffering from financial and social impoverishment [18]. Kafi-Nono is one of the previously marginalized and highly low-resource languages of Ethiopia. However, it is currently spoken and officially working language by about three million people.

Kafi-Nono is a language which is spoken by around 3 million people in south western region of Ethiopia. Emphasis was not given to the language before 1987 by the previous governments. The language started as official working language of the Kafa zone since 1987 and offered as independent course both at primary and secondary school level in Kafa zone [61,85].

One of the existing problems with Kafi-Nono language is lack of automatic word completion system. Some speakers of the language may not be familiar with how to spell some words in Kafi-Nono because of pronunciation difference due to different pronunciation in different zones of the regions language hence they are writing as they are pronouncing which may not be correct.. The problem is more severe for peoples where Kafi-Nono is their second language. The lack of word completion, creates multiple problems with Kafi-Nono texts literary works such as journals, Books, fictions and some newspapers. This may undermine the wide usage of the Kafa language, limits the amount of linguistic research in the region and impacts image of Kafi-Nono for new generation due to misspelling the words. This has a potential to restrict readers to enjoy books published using Kafi-Nono language due to wide range of inconsistencies in spelling. As a result, it can be safely assumed that books and other knowledge produced about the Kafa people may lead to varying interpretations and hence the future generation may have little knowledge of culture, custom, religion, etc. The problem of misspelling is magnified when non-Kafa language history Speakers try to create document using text editing software.

Word sequence prediction, being the product of NLP and AI, is one of the most broadly used systems to enhance communication in augmentative and alternative communication [16]. A number of word sequence prediction systems exist for different local languages to assist users on their text entry. Amharic [20, 21, 22], Afan Oromo, from some of the local languages and Swedish [23, 24], English [25], Italian [26], Persian [27], Bangle [16] from international languages are some of word prediction studies conducted

lately. These studies contribute in reducing the error, effort and time to write a text for typesetting, and for people who are not able to use a conventional keyboard. As to the researcher knowledge there is no attempt to come up with word prediction for Kafi-Nono language.

Kafi-Nono has most of the features of agglutinative languages where all certain forms (morphemes) are affixes (mostly suffixes and sometimes infixes). In agglutinative languages like Kafi-Nono most of the information is communicated through affixes (infixes and suffixes) attached to the roots or stems. For example, the verb daamo which mean take can be inflected in to more than 90 forms like daamme, daamite, daamaache, etc. to mean, we will take, taken, not taken, etc. Since Kafi-Nono is morphologically very rich, derivations and word formations in the language involve a number of different linguistic features including affixation, reduplication and compounding [28]. The difference in length of both vowels and consonants induces difference in meaning. For example, the Kafi-Nono word baaroo means Holiday while baroo means another or forehead and baaro means corn or maize.

In addition, by using word sequence prediction the swiftness of typing could be greatly advanced when people write Kafi-Nono texts and at the same time correct misspelled word that create miscommunication between Authors and readers. A Single letter may change the meaning of the word if misspelled. Furthermore, lack of Kafi-Nono word auto completion impacts non-native speakers from learning Kafi-Nono language in its proper form. Due to misspelling and low speed of typing, the new Kafi-Nono speakers could develop low-esteem that prevents them from practicing the language. Therefore, this study is undertaken to solve the above listed problems by providing Kafi-Nono word auto completion. The purpose of this study is to design and develop word sequence prediction language model for Kafi-Nono with inclusion of context information. The developed model can be used in predictive text entry systems and writing aids. In this research we are therefore interested in answering the following main Research questions:

- how to prepare data set of Kafi-Nono for transfer learning?
- which model of transfer learning is suitable for Kafi-Nono word prediction?
- To what extent the proposed Kafi-Nono word sequence prediction works?

1.5 Objectives

1.5.1 General Objective

The general objective of this research is to plan and develop a word sequence prediction model for Kafi-Nono language.

1.5.2 Specific Objectives

The specific objectives of this thesis are to:

1. Analyze and study the morphology and structure of Kafi-Nono language
2. Construct Kafi-Nono corpus.
3. Develop word sequence prediction model.
4. Evaluate the performance of word sequence prediction model using collected data.

1.6 Benefit/Significance of the Study

Word Prediction is mainly useful for users with motor losses. The prediction of the best probable words benefits to minimize the number of keystrokes needed to type a text and frequently becomes vital for users with speech and language disabilities or dyslexia: it has been proven that such systems assist users representing an Alternative and Augmentative Communication (AAC) technique, [8]. Specifically, our proposed system will benefit in such a way that:

- Increase NLP resource for the target language.
- Contribute for the development of Kafi-Nono language.
- assist physically disabled individuals who have typing difficulties
- speed up typing by decreasing keystrokes for mobile phone, computer and other hand-held device users.

- Suggesting correct words, hence giving error free text entry.

Furthermore, this study will be a stepping stone for other NLP researches such as Machine translation, speech recognition etc.

1.7 Scope and Limitation of the Study

The scope of this study is to construct word sequence prediction model to investigating word prediction for Kafi-Nono words at word and phrase level which study the actual prediction of the word given in corpus.The technique used was an unsupervised machine learning approach particularly based a deep learning approach, using transfer learning. This study is conceptually limited to developing prototype for Kafi-Nono words that auto complete words. Due to the absence of standard training and test corpus, we have prepared mixed dialect training and test sets data for the experimentation which needs further development for other purposes.

1.8 Outline of the Thesis

This chapter begins with the discussion about the background of the research. This is followed by the problem statement and then the objective of this research and the methodology. Finally, the scope and the significance of the study are described. The outline of the remaining chapters is given below:

CHAPTER 2: Presents the literature and related works related to language modeling and next word prediction where it starts with Language model, overview of Artificial Neural Networks, Machine Learning for Natural Language Processing, Selective Focusing, Attention Mechanisms, Transformers, Pre-Trained Transformer for Text Generation, Corpus, Architecture of GPT, Performance of transformers, Scale-Up for Better Performance, Training, Restrictions and Partial Release, Limitations of transformer, Applications and Subsequent Research on transformers is discussed.

Comparison of Multilingual Vs Monolingual Models for Cross-Lingual Transfer is presented and the Transformer Model algorithm is overviewed. And also we have provided an overview About Kafi-Nono Language, Linguistic Characteristics, Dialect, Vowels, Kafi-Nono Part of Speeches. Evaluation of the reviewed methods for the Suitability to Kafi-

Nono Language is discussed. Finally Related Works of Local languages and Other African Languages are summarized.

CHAPTER 3: The Chaper discusses the Methodology, the Model Selection, Transformer Language Modeling Algorithm, Datasets and Data Quality; Representative Data and Deduplication. And the Proposed Approach briefly. Also the KN WPS System Architecture, Text Data preprocessing- The Embedding Component- Data Processing, Sentence Segmenting, Tokenization, The Pre-Trained Transformer Model Input Encoding Byte Pair Encoding (BPE) are presented.

CHAPTER 4: In this chapter the Experiment, Results and Discussion on Dataset, Experimental Setup, The Prototype, Fine-Tuning Hyper-Parameters, Training-Hyper parameters, Training and the experiment on How Fast Transformer Model Learns the KN Language Pattern, the Training Loss in Fine-Tuning and Evaluation are presented. Discussion on on the results are also presented

CHAPTER 5: Presents the conclusion and recommendations. This chapter also includes the summary of contributions and future research works.

This section we will examine momentarily on the chosen themes that are identified with our work.

2.1 Language Models

The field of Natural Language Processing (NLP) has been growing quickly over the previous decade. A portion of this advancement has come about because of advancement of computational power, such as, bigger memories and powerful Graphical Processing units (GPUs). These has empowered bigger datasets processing and computationally heavy estimations [29], [30], [31]. These computational performance enhancements have likewise empowered scientists to utilize a lot more significance in preparing corpus for language Modeling (LM), a sub-field of NLP [4].

One can consider language modeling as a task of giving a probability to given sentences. Practically speaking, this would imply that when one presents a sequence of words into a language model, the language model will yield a likelihood of how possible these words will show up in the provided request. This section will present the significance of language models in current society and clarify how the traditional (non-neural) language models work.

2.2 Need for Language Models

Language modeling in current times started during the 1980s when the primary models of some importance were created [32]. These first models were intended for written words, and, from that point forward, the model has been adjusted and improved to incorporate spoken words. Today, language models are expected to speed up and upgrade the communication between people just as for the association between people and PCs. Some substantial and apparent use cases for language models are smart keyboards, response suggestions for emails [33], auto-correction for spelling, and remote helpers (virtual assistants).

Figure 2.1: *An example of accelerated human-computer interaction by Google* [99]

2.2.1 Accelerating communication

The most apparent language model implementation, concerning the speed increase of communication between people, is the auto-completion apparatus on a smart cell phones. Google previously began utilizing auto completion in its web index 2.1 in 2004 to improve and speed up the communication among people and PCs.

These days, both Google and Apple are utilizing language models in their product to predict subsequent words when composing messages 2.2. This forecast of resulting words should be possible with supposed measurable language models or neural network language models (NNLMs). Today, nonetheless, the preference is more for the utilization of NNLMs because of their prevailing performance, which will be discussed in this section.

(a) *iPhone: apple inc.* (b) *android: by google*

Figure 2.2: *An example of accelerated interaction between people taken from* [100]

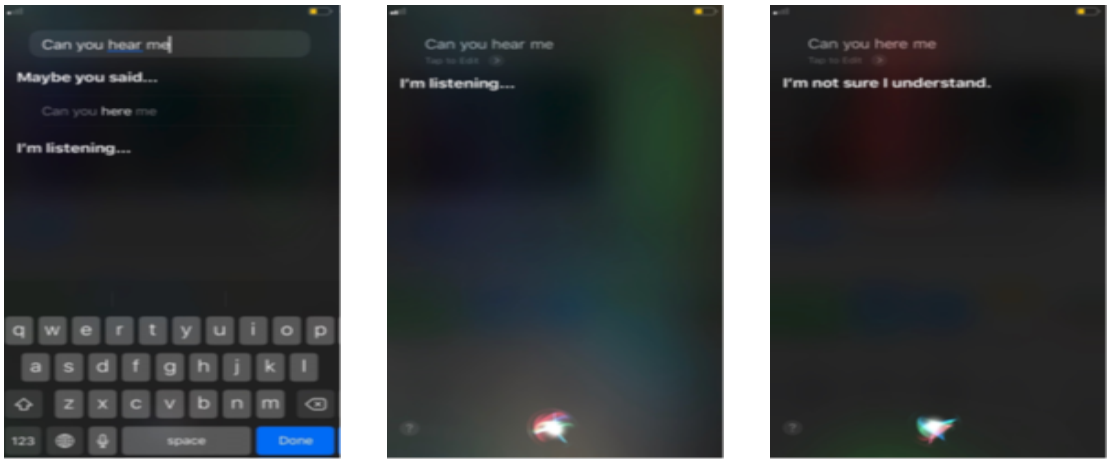
2.2.2 Human-computer Interaction

Language models undertake a basic part in human-PC collaboration with speech recognition system (ASR) frameworks when ASR frameworks look to comprehend the set-

ting of what the human is attempting to say. For people, this coordinating of speech or voice to words is simple since we have learned, through experimentation, to naturally coordinate with the right words and expressions for the duration of our lives. Although, PCs come up short on the context oriented information that is important for successfully preparing spoken communication. For instance, if a human inquires: *Can you hear me?* Now, as the pronunciation of “hear” and “here” are so similar, the ASR may inaccurately interpret the question as: *Can you here me?* This inquiry would look bad for the individual with background information that assists with comprehension or "predict" the proposed expression. The language model is the instrument that tells the speech recognition system, which of the given arrangement of potential expressions, is the most likely. Thus, the LM is fundamental for the presentation of a speech recognition framework as it probably won’t be certain which words have been said. For instance, because of comparative pronunciations, poor hearing of the voice, or loud background noise.

A language model can identify phrases that make no sense to a human speaker because they have learned the probability of sentences by seeing vast amounts of written text. A language model, trained with quality data, should not have seen a sentence "Can you here me?" but it has most probably seen some combination of a sentence "Can you hear me?". Hence, if a speech recognition system asks the language model which one of these is most likely to appear, it will give a higher probability to the "Can you hear me?" sentence. This way, LMs help ASR systems understand contextual information, which improves accuracy and performance in speech recognition. If speech recognition systems were not using a language model in deciding what has been spoken, it would generate much more sentences that would not make sense.

Some voice aides can even show all potential understandings of a human’s spoken expression. For example, Apple’s Siri is one such voice assistant, as displayed in 2.3



(a) (b) (c)

Figure 2.3: *Possible interpretations of a spoken phrase* [12]

2.3 Classic Language Models

The historical background of language models is rich. It began from the classic methodologies dependent on statistical language modeling, such as, n-gram models that utilized diverse smoothing procedures to deal with concealed n-grams [34]. One late synopsis of this set of experiences of language modeling is done by [35]. The most recent advancements in the quickly arising field of language modeling are discussed in this work.

2.3.1 Statistical Language Modeling

In this work, since there are many statistical methods, we will only see the most recent one, N-gram models One key capacity of NLP has been Statistical Language Modeling, which is basic for speech recognition and machine interpretation [32].

Statistical Language Modeling is intend to become familiar with the probability P(w1, ..., wn) of a grouping of words w1, ..., wn [2], [36], [28]. The chain rule 2.1 of probability can be used to ascertain this likelihood

$$P(w_1, w_2, .., w_n) = \prod_{i=1}^{n} P(w|w_1, w_2, .., w_1) \tag{2.1}$$

As there is much variation in the number of words that may precede a given word, and also due to the complexity of calculating P(wi | w1, ..., wi-1) for many words i, the probability of a word is typically conditioned on a window of m previous words 2.2.

$$P(w_1, ..., w_n) \sim \prod_{i=1}^{n} P(w_i|w_i - m, ..., w_i - 1) \tag{2.2}$$

There are various approaches to utilize a language model. For instance, the model can expect and foresee succeeding words; it can likewise give probabilities to sentences. The accompanying model exhibits this. The language model may figure that the sentence: **that is when I saw the three big giants walking towards me** has a greater likelihood of appearing in a text than the same sentence with a different ordering of the words **walking big that saw is the me three when I giants towards**

In addition to other things, this is utilized for tasks that recognize words in vague settings, such as recognizing human speech, where the information is noisy. The accompanying Equation 2.2 presents one of the settled in traditional language models (called

n-gram models) that have been utilized by the analysts for quite a long time.

2.3.2 N-Gram Models

The N-gram model is a language model with low difficulty, which is just an arrangement of N words. For example, an arrangement of two words is a bigram (or 2-gram): "I saw" or "walking towards". Adding a word to the grouping makes a trigram (or 3-gram) that contains three words; "walking towards me". In a trigram model case, the likelihood of a grouping of words w1, ..., wn would be determined in the accompanying manner 2.3

$$P(w_1, ..., w_n) \sim \prod_{i=1}^{n} P(w_i | w_i - 2, ..., w_i - 1) \tag{2.3}$$

A trigram model can be generalized because it observes the two previous words in a given sequence, so it can be calculated as an N-gram that takes into account N-1 words (2.4)

$$P(w_1, ..., w_n) \sim \prod_{i=1}^{n} P(w_i | w_{i-N+1}, ..., w_{i-1}) \tag{2.4}$$

Here we use the Markov assumption, which is the term for the basic assumption that the probability of a word is only dependent on a limited number of previous words. An easy way of calculating trigram or N-gram probabilities is by using maximum likelihood estimation (MLE) [28]. The estimate given by MLE for the N-gram probability of a word wi given a previous sequence of words h = wi | wi-N+1, ..., wi-1 can be calculated by summing the number of times wi appearances in the context h, and normalizing this by dividing every observation with h 2.5 [36], [37].

$$P(w_i | w_{i-N+1}, ..., w_{i-1}) = \frac{count(w_{i-N+1}, ..., w_{i-1}, w_i)}{count(w_{i-N+1}, ..., w_{i-1})} \tag{2.5}$$

For instance, consider that the words "three big" gives the context of those words h, and we wish to forecast the likelihood that the next word in the sequence w will be "giants". A training corpus offers a trigram model the ability to count the number of times "three big" was followed by "giants" and calculate 2.6

$$P(w_i | w_{i-N+1}, ..., w_{i-1}) = \frac{count(\text{threebiggiants})}{Count(\text{threebig})} \tag{2.6}$$

However, even N-gram models that have been trained with a large corpus are problematic. The reason for this is that it is challenging to calculate N-Gram probabilities. Like our previous example, numerous sequences of words typically appear very infrequently, only once, or not at all [36], [37], [32]. Let us consider the three-word sequence "walking towards me". What is the probability of having the word "me" following a sequence of words "walking towards"? A training corpus may contain not a single instance of that particular sequence. As a consequence, count (walking towards me) would be zero, and hence, P(me | walking towards) would also be zero. This is problematic as the sequence of words "walking towards" may appear a number of times in the corpus. Forecasting with Equation 2.7 would fail to correctly estimate the actual likelihood of the sequence appearing.

$$P(me|walkingtowards) = 0 \tag{2.7}$$

Thus, the use of a standard N-gram model would yield such inaccurate zero probabilities on far too many occasions, making the model's predictions very noisy. Therefore, in order to circumvent probabilities of zero, it will be necessary to apply smoothing techniques. These smoothing techniques remove some probability mass from frequent events, redistributing it to unseen events. For example, those that have been assigned zero probability by the N-Gram model. [36] [38], [37]. There are significant issues related to the use of N-gram models, even though they are effective in certain contexts with a limited range of words and phrases. Modern recurrent neural network language models (RNNLMs) can offer improved perplexities and error rates in speech recognition systems compared to these traditional n-gram approaches [2], [5], [39], [40]. The following section introduces such RNNLMs.

2.3.3 Word Prediction Using Frequencies of Words

The simplest word prediction method is to build a dictionary containing words and their relative frequencies of apparition. When the user starts typing a string of characters c, the predictor offers the n most frequent words beginning by this string in the same way they are stored in the system [41]. Then, the user can choose the word in the list he or she wanted to enter or continue typing if it is not in the list [41]. There are several studies about word frequencies in different languages, for instance [42] gives information about the frequency of word occurrence in English used by some disabled people. In this approach Using the frequency every single word or unigram word model, it is clear that some of the predicted words are not appropriate because context or history of words is not taken into account. This means word sequence history would provide a clue for

appearance of the next words. Nevertheless, in most case, it is difficult to calculate the probability of entire sequence. Based on Markov assumption in which only last n-1 word of the history affects succeeding word. Two common statistical models those provide a compatible technique to compute probabilities of next words though Markov model are N-gram and HMM [43] [44].

2.3.4 Word Prediction Using Probability

Another possibility is to use the relative probability of appearance of a word depending on the previous one. To implement this system a two-entries table is needed to store the conditional probability of appearing of each word Wj after each Wi. If the dictionary contains N words, the dimension of the table will be of N*N. That is, it will have N2 entries, but most of the values in the dictionary will be zero or close to zero. In some cases, it could be possible for the system to give proposals before entering the beginning of a word. The recentness of use may also be included in this approach This method is hardly adaptable to include the user preferred words because the dimensions of the dictionary cannot be changed. This difficulty leads to the design of modified versions, like the one that uses only the most probable pair of words.

2.3.5 Informed (Knowledge based) Models

Syntactic Word Prediction Using Probability

This approach takes into account the syntactic information inherent to the languages. In this way, the set of words that are candidates to be proposed by the predictor is restricted to the ones that match the most probable syntactic role in the current position of the sentence, thus increasing the hint rates. This syntactic vector dimension is smaller than the one used in the previous approach, and the proportion of probabilities which are close to zero is also smaller. Each entry in the dictionary will associate a word with its syntactic category, and its frequency of apparition. Words can be sorted by syntactic categories to facilitate the selection process. When a word is syntactically ambiguous, that is, when more than one category is possible for a given word, one entry for each possible category may be created. The table of conditional probabilities of syntactic categories has a fixed size and it is built before the use of the predictor. Adaptation to the user's lexicon is possible because there is no need to increase the size of the table. New words are included in the dictionary with a provisional syntactic category deducted from its use.

Later on, the system may require some help from the user to verify if the categorization was correct. wet could be also possible to add some morphological information in the dictionary to propose the words with the most appropriate morphological characteristics (gender, number). This could increase the hint rate of the predictor .

Syntactic Word Prediction by Using Grammars

In these approaches, the current sentence is being parsed using a grammar to get the most probable categories. Parsing methods for word prediction can be either top-down [45]] or bottom up [46].

So, there is a need to define the syntactic rules (typically LEFT <[RIGHT]+, usually being LEFT and RIGHT some syntactic categories defined in the system) that are used in a language. Within a rule, it could be possible to define concordance amongst the components of the right part (either in gender and/or in number). Then, the suggestions may be offered with the most appropriate morphological characteristics. wet is necessary to leave open to the user the possibility of changing the word's ending. For example, if there is a mismatch in the rule used by the system, it may be necessary to modify the end of an accepted proposal. The dictionary is similar to the one used in the previous approach with the addition of morphological information to allow concordance. The complexity of this system is also larger because in this case, all the words of the sentence that appear before the current word are taken into account, while in the previous approaches only one previous word was used. The adaptation of the system for the new words is made increasing the word frequencies and the weights of the rules. The inclusion of new words is similar to the one in the previous approach [41].

Semantic Word Prediction

These methods are not very used, because their results are similar to those of the syntactic approaches, but the increase in complexity is great. Maybe the simplest method that can be used is the semantic word prediction by using parsing methods. In this approach each word has some associated semantic categories, while in the previous one categories were purely syntactic. The rest of the features (the procedure, complexity, structure of the dictionary, adaptability...) are similar to the previous one. Nevertheless, the problem of giving semantic categories to the words is very complex and it results difficult to be programmed [41]. There may be other methods to treat the semantic information, but their complexity is going to be very great for a real-time system as the word predictors are intended to be, even the time requirements maybe a few seconds

between two consecutive keystrokes of an impaired person are not very strong for the computational capacities of today's ordinary equipment like hand held devices.

2.4 Neural Network Language Models

In this section, our purpose is to give an overview of the most important LMs. Each technique is described and along with its performance on LM is discussed. Our structured overview makes it possible to detect the most promising techniques in the field of LM.

Neural networks [12] are a collection of algorithms that is aimed at recognizing patterns. Their nature is very similar to the human brain. Machine perception, labelling, or clustering of source data helps to interpret sensory data. All real-world data (images, sound, text, or time series) must be converted into vectors for a neural network to recognize numerical patterns. Artifi- cial neural networks (ANN) consist of many deeply interconnected processing elements (neurons) that collaborate to solve a problem. ANN typically includes numerous processors that run parallel and are organized in tiers. The same as optic nerves get input in human visual processing first level gets the initial input. Each subsequent level accepts output from the level prefacing it, not from the raw input—similar to the neurons distant from the optic nerve that gets signals from those closes to it. The last level gives the system result. Recurrent neural networks [13] are generalizations of a direct transmission neural network that has internal memory. The RNN is repetitive in nature because it performs the same function for each data entry, but at the same time, the current output depends on the previous calculation. After receiving the original data, it is copied and sent back to the periodic network. The decision is based on an analysis of the current input and the output from the previous input. RNNs can use their internal state (memory) to process input sequences when direct communication neural networks cannot. The internal state helps them in duties such as speech recognition or unsegmented, connected handwriting recognition. Unlike RNN, other neural networks' inputs are independent of each other. All RNN inputs are interconnected. Drawbacks of the periodic neural network RNNs are as follows:

1. Gradient problems of disappearance and explosion

2. RNN training is a complicated task

3. In the case of tanh or relu activation function models cannot process very long sequences

Language models (LM) can be classified into two categories: count-based and continuous-space LM. The count-based methods, such as traditional statistical models,usually involve making an n-th order Markov assumption and estimating n-gram probabilities via counting and subsequent smoothing. The LM literature abounds with successful approaches for learning the statistical(count) based LM: modified Kneser-Ney smoothing, Jelinek-Mercer smoothing [47, 48] etc. In recent years, continuous-space LM such as feed-forward neural probabilistic language models (NPLMs) and recurrent neural network language models (RNNs) are proposed. This Neural Language Models (NLM) solves the problem of data sparsity of the n-gram model, by representing words as vectors (word embeddings) and using them as inputs to a NLM. The parameters are learned as part of the training process. Word embeddings obtained through NLMs exhibit the property whereby semantically close words are likewise close in the induced vector space. Moreover, NLMs can also capture the contextual information at the sentence-level, corpus-level and subword-level.

Neural language model (NLM) is also known as Continuous-space LM. There are two main NLM: feed-forward neural network based LM, which was proposed to tackle the problems of data sparsity; and recurrent neural network based LM, which was proposed to address the problem of limited context. Recently, recurrent neural network based approach have achieved state-of-the-art performance. The early proposed NLM are to solve the aforementioned two main problems of n-gram models. Subsequent works have turned to focus on sub-word modelling and corpus-level modelling based on recurrent neural network and its variant — long short-term memory network (LSTM)

2.4.1 Feed-Forward Neural Network Based Models

The first neural approach to LM is a neural probabilistic language model [49], which learns the parameters of conditional probability distribution of the next word, given the previous n-1 words using a feed-forward neural network of three layers. An overview of the network architecture is additionally given in Figure 1. In this architecture,

1. Build a mapping C from each word i of the vocabulary V to a distributed, real-valued feature vector $C(i)R^m$, with m being the number of features. C is a $|V|m$ matrix, whose row i is the feature vector $C(i)$ for word i.

2. A function g over words maps the input sequence of feature vectors for words in context $(C(w_(t-n+1)),,C(w_(t-1)))$to a conditional probability distribution of words in V for the next word w_t .

3. Finally, simultaneously learn the word feature vectors and the parameters of that

probability function with a composite function f, comprised of the two mappings C and g:

$$f(i, w_{t-1}, ..., w_{t-n+1}) = g(i, C(w_{t-1}), ..., C(w_{t-n+1})) \quad 2.8$$

In this model, each word in the vocabulary is associated with a distributed word feature

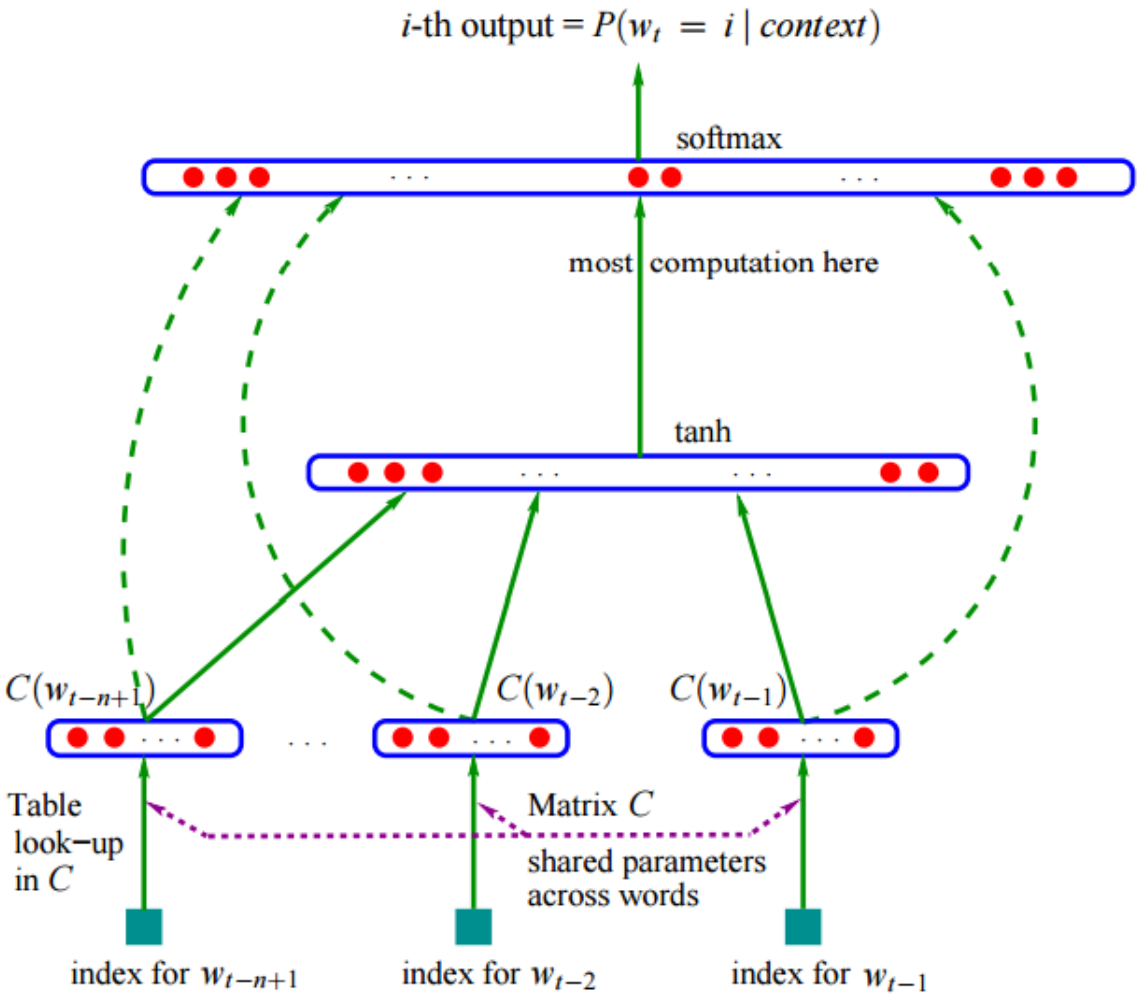


Figure 2.4: *An overview of the network architecture of neural probabilistic language model taken from [1]*

vector, and the joint probability function of words sequence is expressed by a function of the feature vectors of these words in the sequence. The model can learn the word feature vectors and the parameters of that probability function simultaneously. This neural network approach can solve the sparseness problem, and have also been shown to generalize well in comparison to the n-gram models in terms of perplexity. However, a major weakness of this approach is the very long training and testing times. Therefore, several other open questions for the future are addressed, mostly concerning speed-up

techniques, more compact probability representations (trees), and introducing a-priori knowledge (semantic information etc. to cluster the highly discrete word forms).

Two models[50, 51] that were concerned about training and testing speed of NLM were proposed. The basic idea in these papers is to cluster similar words before computing their probability, in order to only have to do one computation per word cluster at the output layer of the NN.

A hierarchical probabilistic NLM [51] is proposed to speed-up training and prediction. A binary hierarchical tree of words in the vocabulary was built using expert knowledge. The binary tree is to form a hierarchical description of a word as a sequence of decisions. Instead of directly predicting each word probability, a hierarchical LM learn to take the hierarchical decisions. This model was two orders of magnitude faster than the non-hierarchical model it was based on. However, it performed considerably worse than its non-hierarchical counterpart.

Another hierarchical LM is the hierarchical log-bilinear (HLBL) model [52], which uses a data-driven method to construct a binary tree of words rather than expert knowledge. The authors first trained a model using a random tree over corpus, then extracted the word representations from the trained model, and performed hierarchical clustering on the extracted representations. In this model, the probability of the next word w is the probability of making the sequences of binary decisions specified by the word's encoding, given its context. Since what only matters for generating a probability at each node is the predicted feature vector, determined by the context, the probability of the current word can be expressed as a product of probabilities of the binary decisions:

$$P(w_n = w|w_{1:n-1}) = \prod_i P(d_i|q_i, w_{1:n-1}) \tag{2.9}$$

where d_i is the i-th encoding for word w_i, and q_i is the feature vector for the i-th node in the path to the corresponding word encoding. The above probability definition can be extended to multiple encodings per word and a summation over all encodings, which allows better prediction of words with multiple senses in multiple contexts. The best HLBL model reported in [52] reduces perplexity by 11.1% compared to a baseline Kneser-Ney smoothed 5-gram LM, at only 32 minutes training time per epoch.

These continuous models share some common characteristics, in that they are mainly based on feedforward neural network and word feature vectors. This approach doesn't suffer from the data sparsity problem, since it can be seen as automatically applying smoothing. After introducing hierarchical tree of words, the models can be trained and tested more quickly, and can outperform non-hierarchical neural models as well as

the best n-gram model.

2.4.2 Recurrent Neural Network Based Models

We have known that feed-forward neural network based LM use fixed length context. However, recurrent neural network do not use limited size of context. By using recurrent connections, information can cycle inside these networks for an arbitrary long time. The recurrent neural network based language model (RNNLM) [53] provides further generalization: instead of considering just several preceding words, neurons with input from recurrent connections assumed to represent short term memory. Specifically, the network architecture is given in figure 2.5, in this architecture:

1. Input layer w and output layer y have the same dimensionality as the vocabulary (10K — 200K).
2. Hidden layer s is orders of magnitude smaller (50–1000 neurons).
3. U is the matrix of weights between input and hidden layer, V is the matrix of weights between hidden and output layer.
4. Without the recurrent weights W, this model would be a bigram neural network LM.

A variant [54] of RNNLM was presented to further improve the original RNNLM by decreasing its computational complexity, which was implemented by factorization of the output layer. In this work, simple factorization of the output layer using classes have been implemented. Words are assigned to class proportionally, while respecting their frequencies. The modified RNN model can thus be smaller and faster, both during training and testing, while being more accurate than the basic one.

We have introduced the two main neural language models. Next, we provide a short overview of the main differences between FNN-based LMs and RNN-based LMs:

1. When using a FNN, one is restricted to use a fixed context size that has to be determined in advance. RNNs in principle use the whole context, although practical applications indicate that the context size that is effectively used is rather limited. However, RNNs at least have the advantage of not having to make decisions on the context size, a parameter for which a suitable value is very difficult to determine.
2. Because RNNs are dynamic systems, some issues which cannot arise in FNNs can be encountered. For example, it may happen that the influence of a given input

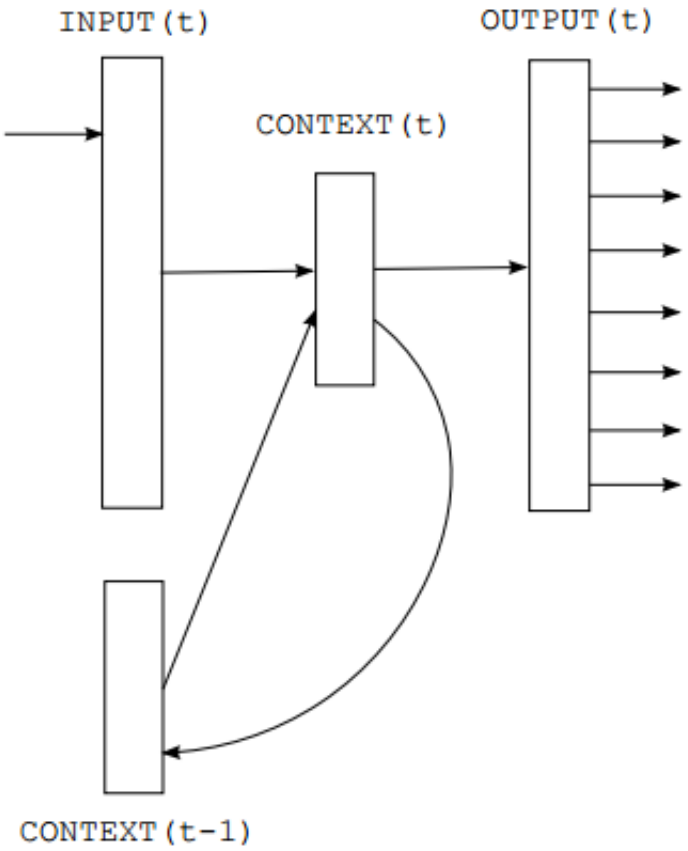


Figure 2.5: *Recurrent NNLM Architecture taken from [2]*

on the network output blows up exponentially as subsequent training examples are presented, a highly undesired artifact.

3. Comparisons between RNNs and FNNs in applications on statistical LM typically favor RNNs. The reason will become clear in later advanced models.

2.4.3 Advanced Models

We noted that NLM are mostly word-level language models up to now. They do not understand subword information (e.g. morphemes). For example, they do not know, a priori, that 'eventful', 'eventfully', 'uneventful' and 'uneventfully' should have structurally related embeddings in the vector space. More recently, people have started to focus on subword-level LM and a character-wise NLM [55] was proposed. This NLM relies on character-level inputs through a character-level convolutional neural network, whose output is used as an input to a recurrent NLM. The experiments demonstrate that the model outperforms word-level LSTM baselines with fewer parameters on language with

rich morphology (Arabic, Czech, French, German, Spanish, Russian).

At the same time, a gated word-character recurrent LM[48] is presented to address the same issue that information about morphemes such as prefix, root, and suffix is lost, and rare word problems using word-level LM. Unlike the character-wise NLM which only dependent on character-level inputs, this gated word-character RNN LM utilizes both word-level and character-level inputs. In particular, the model has a gate that determines which of the two ways to use to represent each word, that is, whether to derive the word into character-level or the word-level itself. This gate is trained to make this decision based on the input word. The major contribution of this model with this kind of threshold mechanism is that it effectively uses the character-level inputs to better represent rare and out-of-vocabulary words.

Actually, the recurrent LM captures the contextual information (i.e. previous words) implicitly across all preceding words within the same sentence using recurrent neural networks. Besides, the range of context that a vanilla RNN can model is limited, due to the vanishing gradient problem. To achieve larger context, a novel method to incorporate corpus-level discourse information into LM is proposed, which is called larger-context LM [48]. The conventional recurrent LM estimate the sentence-level probability, assuming that all sentences in a document are independent from each other.

However, this assumption of mutual independence of sentences in a corpus is not necessary for the larger context LM. It models the influence of context by defining a conditional probability in term of words from the same sentence, but the context is also composed of a number of previous sentences of arbitrary length. That is to say, the context information is modeled explicitly by context representation of a sequence of preceding sentences. Specifically, authors build a bag-of-words context from the previous sentence, and then integrate it into the Long Short-Term Memory (LSTM). Thus, this model explores another aspect of context-dependent recurrent LM. The larger-context LM improve perplexity for sentences, significantly reducing per-word perplexity compared to the LM without context information. Similarly, in order to incorporate document-level contextual information, a document-context LM[48] is presented. In this work, local and global information is combined into the multi-level recurrent architectures in LM.

2.4.4 Summary

In this section, we summarized the current work in LM. Based on count-based(statistical) LM, the NLM can solve the problem of data sparseness, and they are able to capture the contextual information in a range from subword-level to corpus-level. However, a very

long training time and large amounts of labeled-training data are the main limitations. Sub-word modelling and large-context LM are the frontier of LM.

2.5 Transformers for Natural Language Processing

Due to their ability to process sequential information, recurrent neural networks have been used in many NLP applications; unlike FFNNs, they are capable of encoding different weights (and giving different output) for identical items based on their surroundings in a sequence — that is to say, a RNN system that parsed one word at a time could still associate a black dog (a type of food)with fuzzy paws, a corn dog with ketchup, and a sun dog with refraction. Moreover, since the retention of information from previous sequence items can be performed recursively, RNN systems can be designed that recall items arbitrarily far back in a sequence: for example, being able to continue the sequences Tom looked at the black dog, Tom looked at the corn dog, and Tom looked at the sun dog with fondly, hungrily, and indirectly, respectively [56] [57].

While capable of impressive solutions, many-layered FFNNs and RNNs both proved vulnerable to the vanishing gradient problem: since gradients (encoded as finite-precision numbers) are required to back propagate across all layers of a model, they can vanish to zero (or explode to infinity) over a sufficiently large number of layers. The long short-term memory network (LSTM), first proposed by [15] [41] sought to resolve this issue by introducing a novel architecture consisting of multiple distinct cells with input, output and forget gates. In 2009, an LSTM-based model submitted by Alex Graves' team won the ICDAR competition for handwriting recognition; [58] another was the most accurate model in the competition and a third was the fastest [59].

Another issue RNNs and LSTMs encounter is that they can only take into account the context of previous sequence items. [56, 60] This can create issues when parsing sentences like Tom rode his bike to the store, put out the kickstand, and turned off the engine, in which the necessary context of the bike being a motorcycle is revealed only at the end. One method of solving problems like this is the bidirectional LSTM, which proceeds in both directions simultaneously, giving access to both past and future input features.[56] Conditional random fields use tags to connect inputs directly to outputs.[56] There exist combinations of the above approaches, like the LSTM-CRF network and the BI-LSTM-CRF network.[56] Other improvements on the RNN model include neural Turing machines, adaptive computation time, neural programmers, and attention mechanisms, the latter of which form the basis for the development of a transformer called Generative Pre-Trained version-2 (GPT-2) and related technologies [61].

GPT-2 is a decoder transformer model [62], which has achieved state-of-the-art performance in text prediction [**?**]] due to its ability to observe all sequence elements simultaneously with its self-attention block.

2.5.1 The Transformer Algorithm

In order to predict a sequence of response tokens M={m_t},t=0,1,...,N conditioned on initial word typed in by a user {w_t}, t=0,1,..,T we need to estimate the following conditional probability distribution 2.10

$$P(m_0, m_1, ..., m_N | w_0, ..., w_T) = \prod_{i}^{N} P(m_i | w_0, w_1, ..., w_T, m_0, ..., m_{i-1} \quad (2.10)$$

With the auto-regressive approach, the objective is to maximize the following log-likelihood(2.11)

$$L(M) = \sum_{i} logP(m_i | w_0, ..., w_T, m_{i-k}, m_{i-k+1}, ..., m_{i-1}; \partial \quad (2.11)$$

where k is the length of predicted word sequence, and the conditional probability P modeled using a neural network with parameters ∂. These parameters are learned via stochastic gradient descent(SGD) optimization procedure.

GPT-2 applies a multi-headed self-attention operation over the input context tokens (2.11) followed by position-wise feed-forward layers (2.12) to produce an output distribution over target tokens (2.13)

$$h_0 = w_e.C + W_p \quad (2.12)$$

$$h_1 = transformer - block(h_{l-1}), \forall l = 1, 2, ..., n \quad (2.13)$$

$$P(m_t) = y_t = softmax(h_n.WT_e), t = 0, ..., N \quad (2.14)$$

where C= c-1, c-k+1, ...c-1 is the context vector of tokens, n is the number of layers, $W_e \epsilon R^{|V_x| x d_x}$ tokens embedding matrix, and $W_p \epsilon R^{N_c t x x d_x}$ is the position embedding matrix, which encodes relative positions of tokens in a sequence.$N_c tx$ is the length of the sequence attended to (context length), |V| is vocabulary size, and $d_x s$ the embedding dimension.

We are reusing the input token embedding matrix as the output classification matrix [27], which allows to remove the large fully connected layer reducing the number of parameters by 25%.

More specifically, we introduce a projection matrix $A = (a)_i j \epsilon R^d model x d x$ initialized according to a random uniform distribution

Given an encoded word context and a hidden state at the last temporal step $h_n(T) \epsilon R^d model$ the predicted token embedding vector by multiplying the two together as (2.15)

$$W_e^p red = (W^p red)_j \epsilon R^d x \tag{2.15}$$

$$W_j^p red = \sum_i h_n i(T_j) a_i j \tag{2.16}$$

Subsequently, the logits are obtained as (2.17):

$$y_k = \sum_j W k j W_j^p red + b_k \tag{2.17}$$

where $b_k, k = 0...|V|_1$ is the bias vector, and number of hidden units per transformer block. During inference, beam-search decoding algorithm is applied to iteratively extract best token sequences according to a negative log-likelihood optimization objective.

In the following sections we will examine what features and mechanisms made transformers more powerful than other NNLMs like RNN, LSTMs and FFNNs.

2.5.2 Selective Focusing

By the early 2010s, the best performance in neural machine translation was achieved with the encoder–decoder model, in which a RNN or LSTM encoder network encoded source sentences into vectors, and a decoder network of similar architecture processed these vectors into translated output. [63] saw the introduction of significantly more complex attention mechanisms, which vastly augmented these models' performance. Attention mechanisms gave these models the ability to adaptively focus their decoder networks' attention on specific aspects of the source text, rather than forcing them to parse the entire text as one vector. [64] then saw the introduction of transformer models, which went a step further by using attention mechanisms to replace the RNN/LSTM architecture entirely [16] [65].

2.5.3 Attention Mechanisms

One constraint of encoder–decoder models like BERT was the difficulty of compressing the encoding of larger sentences into fixed-length vectors; performance often deteriorated on larger inputs. In 2014, Bahdanau et al. [66] introduced an extension to the encoder–decoder model that could align and translate jointly. [64]

For each word of the source sentence that was translated, the Bahdanau model's encoder (a bidirectional RNN with 1000 hidden units in each direction) searched the entire rest of that sentence for the positions of relevant information. Rather than giving the decoder a fixed-length vector encoding of the entire input sequence (like previous models), it produced context vectors, associated with those positions as well as previously generated target words. [66] The decoder (which also had 1000 hidden units) then used these context vectors to decide where to focus its attention [66] [64][61].

While attention mechanisms were effective in improving performance when used to augment existing convolutional and recurrent neural network architectures, it was soon discovered that performant models could be built using attention mechanisms on their own, without anything else underlying them [3].

In June 2017, the Generative Pre-Trained transformer architecture was first introduced, in a paper released by Google's deepMind. [3] Transformers are a type of model based solely on attention mechanisms, discarding convolution and recurrence altogether. Unlike previous RNN-based models, transformers can process sequential input without needing to perform computation on each item in sequence; this means they can be massively parallelized. [3] On the WMT'14 French–English task, specifically trained French–English translation model using the transformer architecture was able to establish a new single-model benchmark of 41.8 BLEU [3]. Since their introduction, transformers have seen use in many NLP applications [60].

2.6 Pre-Training Transformers

On June 11, 2018, OpenAI released a paper entitled Improving Language Understanding by Generative Pre-Training, in which they introduced the Generative Pre-Trained Transformer (GPT) [62] At this point, the best-performing neural NLP models primarily employed supervised learning from large amounts of manually labeled data. This reliance on supervised learning limited their use on datasets that were not well-annotated, in addition to making it prohibitively expensive and time- consuming to train extremely large models;

[62, 67] many languages (such as Swahili or Haitian creole) are difficult to translate and interpret using such models due to a lack of available text for corpus- building.[62] In contrast, GPT's semi- supervised approach involved two stages: an unsupervised generative pre-training stage in which a language modeling objective was used to set initial parameters, and a supervised discriminative fine-tuning stage in which these parameters were adapted to a target task.[62]

The use of a transformer architecture, as opposed to previous techniques involving attention-augmented RNNs, provided GPT with a more structured memory than could be achieved through recurrent mechanisms; this resulted in robust transfer performance across diverse tasks [62] During transfer, we utilize task-specific input adaptations derived from traversal-style approaches, which process structured text input as a single contiguous sequence of tokens. [62]

2.6.1 Corpus

The transformer's unsupervised pre-training was performed using booksCorpus, [68] a dataset of over 7,000 unpublished fiction books from various genres; while other models this dataset was chosen in part because its long passages of continuous text conditioned the model to handle long- range information. Other available datasets, while larger, were rejected on the basis that they lacked this long-range structure (being shuffled at a sentence level). [62] The ftfy library was used to clean the BooksCorpus text (standardize punctuation and whitespace); it was tokenized using spaCy[62]

2.6.2 Architecture of GPT

GPT's architecture used in this work was a 24-layer decoder-only transformer, using 24 masked self-attention heads, with 128 dimensional states each for a total of 1536. Rather than simple stochastic gradient descent, the Adam optimization algorithm was used; the learning rate was increased linearly from zero over the first 2,000 updates, to a maximum of 2.510^{-}4, and annealed to 0 using a cosine schedule [62]. While GPT's fine-tuning was adapted to specific tasks, its pre-training was not; to perform the various tasks, minimal changes were performed to its underlying task-agnostic model architecture [62].

Despite this, GPT still improved on previous benchmarks in several language processing tasks, outperforming discriminatively-trained models with task- oriented archi-

tectures on a number of diverse tasks [62].

2.6.3 Performance

Models are mostly evaluated for their ability of classifying the given pairs of sentences from various datasets based on the relationship between them using natural language inference task also known as textual entailment. They classify as "entailment", "contradiction" or "neutral"[62]. Examples of such datasets include QNLI (wikipedia articles) and MultiNLI (transcribed speech, popular fiction and government reports, among other sources);[69] on these GPT achieved, respectively, a 5.8% and 1.5% improvement over previous best results [62] It similarly outperformed previous models on two tasks related to question answering and commonsense reasoning — by 5.7% on RACE,[69] a dataset of written question– answer pairs from middle and high school exams, and by 8.9% on the Story Cloze Test [70].

Another task, semantic similarity (or paraphrase detection), assesses whether a model can predict whether two sentences are paraphrases of one another; on the Quora Question Pairs (QQP) dataset, GPT improved on previous best-performing models by 4.2%. [62] In a text classification task using the Corpus of Linguistic Acceptability (CoLA), GPT achieved a score of 45.4, versus a previous best of 35.0. Finally, on GLUE, a multi-task test, [71] GPT achieved an overall score of 72.8 (compared to a previous record of 68.9).

2.6.4 Training

Since the transformer architecture enabled massive parallelization, GPT-series models could be trained on larger corpora than previous NLP models. While the initial GPT model demonstrated that the approach was viable, GPT-2 would further explore the emergent properties of networks trained on extremely large corpora. CommonCrawl, a large corpus produced by web crawling and previously used in training NLP systems,[71] was considered due to its large size, but was rejected after further review revealed large amounts of unintelligible content [9, 71] Instead, OpenAI developed a new corpus, known as WebText; rather than scraping content indiscriminately from the World Wide Web, WebText was generated by scraping only pages linked to by Reddit posts that had received at least three upvotes prior to December 2017. The corpus was subsequently cleaned; HTML documents were parsed into plain text, duplicate pages were eliminated, and Wikipedia pages were removed (since their presence in many other datasets could have induced overfitting)[9].

While the cost of training GPT-2 is known to have been $256 per hour, [72] the amount of hours it took to complete training is unknown; therefore, the overall training cost cannot be estimated accurately.[73] However, comparable large language models using transformer architectures have had their costs documented in more detail; the training processes for BERT and XLNet consumed, respectively, $6,912 and $245,000 of resources.

2.6.5 Limitations

GPT-2 can generate contextually-appropriate text for a range of scenarios, even odd ones like a CNN article about Donald Trump giving a speech praising the anime character Asuka Lang; ley; Sory; u. Here, the tendency to generate nonsensical and repetitive text with increasing output length, even in the full 1.5B model, can be seen; in the second paragraph, grammar begins to deteriorate, and the output eventually becomes one incoherent sentence repeated over and over.

While GPT-2's ability to generate reasonable passages of natural language text were generally remarked on positively, its shortcomings were noted as well, especially when generating texts longer than a couple paragraphs; Vox said "the prose is pretty rough, there's the occasional non-sequitur, and the articles get less coherent the longer they get".[74] The Verge similarly noted that longer samples of GPT-2 writing tended to "stray off topic" and lack overall coherence;[12] The Register opined that "a human reading it should, after a short while, realize something's up", and noted that "GPT-2 doesn't answer questions as well as other systems that rely on algorithms to extract and retrieve information"[72].

GPT-2 deployment is resource-intensive; the full version of the model is larger than five gigabytes, making it difficult to embed locally into applications, and consumes large amounts of RAM. In addition, performing a single prediction "can occupy a CPU at 100% utilization for several minutes", and even with GPU processing, "a single prediction can take seconds"[3].

2.6.6 Applications and Subsequent Research

Possible applications of GPT-2 described by journalists included aiding humans in writing text like news articles. [65] Even before the release of the full version, GPT-2 was used for a variety of applications and services, as well as for entertainment. In

June 2019, a subreddit named SubSimulatorGPT2 was created in which a variety of GPT-2 instances trained on different subreddits made posts and replied to each other's comments, creating a situation where one could observe an AI personification of Bitcoin argue with the machine learning-derived spirit of ShittyFoodPorn";[12] by July of that year, a GPT-2-based software program released to auto-complete lines of code in a variety of programming language was described as "game-changer"[12].

In 2019, AI Dungeon was launched, which used GPT-2 to generate dynamic text adventures based on user input. [75] While AI Dungeon now offers access to the GPT-3 API as an optional paid upgrade, the free version of the site continues to use GPT-2.[76] Latitude, the company formed around AI Dungeon, raised $3.3 million in seed funding in 2021 [77].

In February 2021, a crisis center for troubled teens announced that they would begin using a GPT-2-derived chatbot to help train counselors by allowing them to have conversations with simulated teens (this use was purely for internal purposes, and did not involve having GPT-2 communicate with the teens themselves) [78].

2.7 Cross-Lingual Transfer Learning Approach

Transfer learning refers to training a model in a resource-rich language and applying it in a resource-poor language in zero-shot or one-shot learning. Zero-shot learning refers to training a model in one domain and assuming it generalizes more or less out-of- the-box in a low-resource domain. One-shot learning is a similar approach that uses a very limited number of dataset from a low-resource domain to adapt the model trained in rich-resource domain. This approach is particularly popular in machine translation where the weights collected for a rich-resource language pair are transferred to low- resource pairs. An example of such an approach is a model by [31]. A parent model is trained in a high-resource language pair (French to English) and some of the trained weights are reused as the initialization for a child model which is further trained on a specific low-resource language pair (Hansa, Turkish and Uzbek into English). Similar approach was explored by [79] where the parent language pair is also low-resource but it was related to the child language pair.Language models and transfer learning have become one of the cornerstones of NLP recently.

The central idea underlying the transfer learning approach is that there are certain commonalities between languages that could be exploited to build, for example, a language model for one language from another model. The process of cross-lingual transfer learning

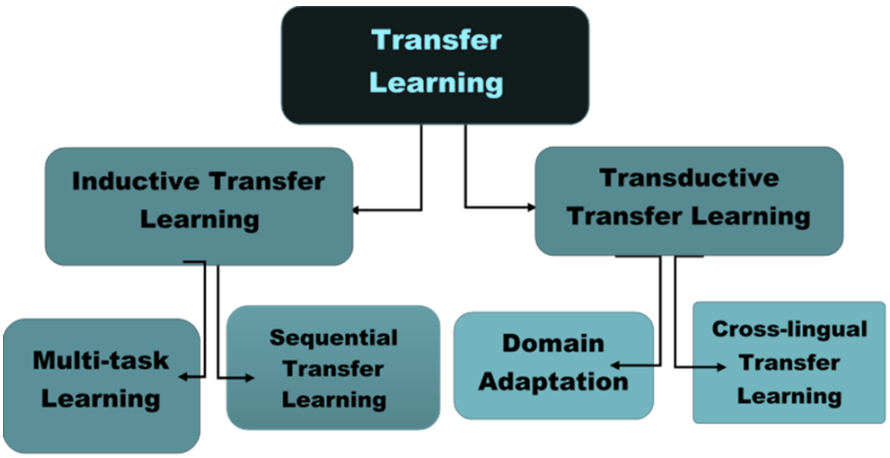


Figure 2.6: *Cross-lingual Transfer Learning Types taken from* [101]

refers to transfer of resources and models from resource-rich source to resource-poor target languages on several levels such as:

2.8 Multilingual VS Monolingual Models for Cross-Lingual Transfer

Multilingual or Polyglot Learning converts data in all languages to a shared representation (e.g., phones or multilingual word vectors) and trains a single model on a mix of data sets in all languages, to enable parameter sharing where possible. This approach is closely related to recent efforts to train a cross-lingual Transformer language model trained on 100 most popular languages and cross-lingual sentence embedding [80]. The later approach learns joint multilingual sentence representations for 93 languages, belonging to more than 30 different language families and written in 28 scripts. With the help of a single BiLSTM encoder with a shared BPE vocabulary for all languages, which is coupled with an auxiliary decoder and trained on parallel corpora, the approach allows learning a classifier on top of the resulting sentence embedding using English annotated data only, and transfer it to any of the 93 languages without any modification.

There is a repeated criticism against multilingual models is that they obtain very less performance than their monolingual counterparts [81]. A comparison of XLM-R and RoBERTa, to evaluate this claim on the XNL in task bench-mark was experimented by [81]. They extended the comparison between multilingual XLM models and monolingual BERT models on 7 languages and compared the performance as in Table below. They trained 14 monolin- gual BERT models on Wikipedia and Common Crawl (capped at 60 GiB), and

two XLM-7 models. They also increased the vocabulary size of the multilingual model, however they show that for cross lingual transfer, monolingual baselines outperform XLM-7 for both Wikipedia and CC by 1.6(%) and 1.3(%) average accuracy.

2.9 Evaluating Language Models

There are two ways of evaluating a language model's performance, extrinsically and intrinsically. Extrinsic evaluation refers to embedding the language model in an application, letting users test the updated system, and telling how much they think the application improves the quality of their end-user experience. For example, an extrinsic evaluation could be performed by embedding a language model to a smartphone's auto-completion tool. With this embedded auto-completion tool, the users could score how useful the auto-completion is and how much time it saves from them or how often they use it. Intrinsic evaluation metrics make it possible to measure the quality of a model detached from any particular application, which means that it makes it easier to compare with other models [38]. In language modeling, the most commonly used intrinsic evaluation metric is perplexity [36], [37]. For doing an intrinsic evaluation, there needs to be a corpus of data that acts as a test set. This corpus of test data is separate from the training data that is used to train the language model. The test set also ensures that the model will not be over trained with the particular training data, which is called over-fitting a model with training [38].

These two evaluation strategies complement each other. Hence, this thesis uses both evaluation methods. Though, intrinsic (perplexity) testing is sufficient and there is need for extrinsic evaluation to make it clear in evaluating language models.

2.9.1 Perplexity

A language model's perplexity (PPL) on a given test set is the inverse probability of the test set, normalized by the number of words [38]. When the test set is W = wi, w2..., wn: It is possible to use the chain rule for expanding the probability of W: As this equation 2.15 shows, the PPL is high when the probability of the words in a particular sequence is low. However, we want to maximize the conditional probability of word sequences and minimize the PPL of a test set. In practice, this means that we are trying to create a language model that can mimic the word sequences of the test set so that it is able to predict a subsequent word when given the previous words. When assuming that the test

set is a perfect representation of the language, it means that the smaller the PPL is, the better the language model performs in creating the actual utterances used in the language [38].

$$PPL(W) = P(w1, w2, \ldots.., wn)^{-1/n} = \sqrt[n]{\frac{1}{P(w1, w2, \ldots, wn)}} \quad 2.18$$

It is impossible to use chain rule for expanding the probability of W (5.3):

$$PPL(W) = \sqrt[n]{\Pi_{i=1}^{n} \frac{1}{P(wi|w1, w2, \ldots, wn)}} \quad 2.19$$

As an example, for bigram (or 2-gram) language model this would be computed as follows (5.6):

$$PPL(W) = \sqrt[n]{\Pi_{i=1}^{n} \frac{1}{P(wi|w_i - 1)}} \quad 2.20$$

Perplexity has useful properties. For instance, PPL can be easily computed for a set of test data. Computing PPL with any language model with a single training and test set makes it ideal for comparing different language models' performances. This fact makes it one of the fastest overall quality metrics when comparing language models [38],[36], [37].

Perplexities also have downsides. First of all, an improvement in PPL (intrinsic) does not necessarily mean an extrinsic performance improvement in a language processing application such as machine translation or speech recognition. Hence, LM's improvement should always be tested also extrinsically in the application before concluding the evaluation of the model. However, the PPL often correlates with the extrinsic improvement of the language model; consequently, it is commonly used as an indicator of the quality of the language model. It is also important to note that the perplexities of two different language models are comparable only if they have the same vocabularies [38].

2.10 Lexical unit selection for NNLM

When it comes to the training data format, Neural network language models can be built in a few different ways. The traditional way of training language models has been to use word-based training data. However, wordbased models have some drawbacks, and hence, other techniques have been developed. Other models built to overcome the challenges that wordbased models face are sub-word based models, character-based models, and combinations of these two models. This section introduces these different ways of building NNLMs.

2.10.1 Word-based models

Word-based language models have the benefit of being the most intelligible language models. It is easy for a human being to understand how a wordlevel based language model works when it estimates the probability of a sequence of words. Word-based embeddings are also well suited for capturing the distributional similarity between words. However, there are also disadvantages of using only word-based language models [38].

One of the disadvantages related to these word-based language models arises when the vocabulary of these models grow. To calculate the probability distribution with a vast vocabulary becomes computationally heavy and slows down the system. The reason for this is the amount of calculated inner products that are the vocabulary size (V) times the word vector (w) length (len(P) * len(w)) which in turn radically slows down the updates on the gradient descent [82]. This problem arises with some languages that simply have too vast a lexicon to represent every word as an embedding.

Fortunately, there are methods that seek to address this challenge. Some examples of these are Hierarchical Softmax [82] Importance Sampling [83], class-based models [5], Noise Contrastive Estimation [84, **?**], and self normalizing partition functions [6].

Another drawback is that even languages or applications with manageable lexicons will encounter unknown words due to spelling mistakes and new and borrowed words from other languages [38]. These word-based language models can only model the words that they know. They are limited only to the words that have been included in the model initiation phase when the initial word vectors were created. If the language model sees words that it does not know, it will replace that word with an <unk> (unknown) token. Replacing words with <unk> tokens decreases the accuracy and performance of the language models due to the loss of the structure and sense of a sentence.

Having an upper limit for vocabulary is a major problem with agglutinative languages because even some common words might not be included in the "known" corpus. The Kafi-Nono language is one such problematic language because it is possible to create words by concatenating morphemes. Hence, it has a vast amount of infrequent words that are some morphological variants, which creates an extensive vocabulary, making word-based models impractical[85].

2.10.2 Sub-word based models

Results based on larger sub-word units have proven to be able to deal with new words and offer reasonable accuracy and training speed [37]. Sub-word approaches also have some drawbacks, such as the specification of the sub-word unit creation, which often differs from language to language. Also, the fact that a word can have multiple different segmentations into sub-word units depends on the context [86]. For instance, in the Kafi-Nono language, the word "keemo" might mean the number three or prophecy. Depending on the context, the ideal sub-word units would be either "keem" (mo) or "keemo" (three or prophecy).

2.10.3 Character-based models

Character-based language models solve problems in capturing the similarity of words like "drink", "drinks", and "drinking", unlike the word-based model (if not using pre-trained word embedding like glove from [80]). Also, character-based models do not need to decide how to split words as the vocabulary is just all the alphabets. However, they have their drawbacks. For example, to successfully model long-term dependencies, we need large hidden representation, which means higher computational costs, that might become unreasonable in practice . One successful approach to overcome some problems of both word-based and character-based language models is a model that uses both of them as an input [87].

2.11 About Kafi-Nono Language

Kafi-Nono is a language which is spoken by around 3 million people in south western part of Ethiopia. It belongs to Afro-Asiatic language super family of the North-Omotic, Southern Gonga sub-group. Kafi-Nono uses Latin script for typing. The language has 22 consonants. Out of these, six of them are both long and short consonants. Five of the 22 consonants, are borrowed from English and Amharic languages. Also, it has five long and short vowels. The long vowels and consonants can be obtained by doubling the corresponding short vowels and consonants, respectively. The difference in length of both vowels and consonants induce difference in meaning. For example, the Kafi-Nono word baro means corn while baroo means forehead. In Kafi-Nono, tone has a semantic and grammatical function. For example, kemo (with high tones) can mean buy and the same

word (with low tones) can also mean sell Kafi-Nono is one of a 'low resource' languages broadly spoken in South western part of Ethiopia. It is spoken by about 3 million people in the Kafa zone and the surrounding.

Emphasis was not given to the language before 1987 by the previous governments. The language started as official working language of the Kafa zone since 1987 and offered as independent course both at primary and secondary school level in Kafa zone [63, 88]. Kafi-Nono uses Latin Script for writing purpose and has 22 consonant phonemes and 5 vowel phonemes. Out of these, six of them are both long and short consonants. Among the 22 consonants, five of them are borrowed from English and Amharic languages. In Kafi-Nono, a sentence is a set of words that contain subject and a predicate. Kafi-Nono Language follows Subject->Object ->Verb (SOV) grammatical rule [63].

2.11.1 Linguistic Characteristics of Kafi-Nono

Fleming (1976 b) Showed that germination is very common in KN language, but germination does not appear at the beginning of the word. Regarding the vowels, he states that diphthongs are rare in the language. Except before /y/ and /w/ or noun suffixes /o/, /e/, or /i/, vowels do not combine as diphthongs. [63] deals with the tonology of Kafi-Nono. He identifies that the language has rising and falling tones in addition to the two basic tones, high and low. He states that absorption and bridging are the two most pervasive phonologically accountable pitch phenomena in Kafi-Nono tonology. It means in Kafi-Nono, when a bimoraic syllable with LH contour occurs followed by H, the sequence LHH undergoes tone absorption resulting in LH. He classifies the nominal and verbs in to different classes (i.e. class I, II, and III) since the Kafi-Nono tonal alternations are associated with morphological processes. This means there are different vowels referred to as theme vowels that pop up when some suffixes are attached to root forms. Consider the following examples.

According to Tadesse, consonant germination, vowel length, and tone are phonemic. Examples in A and B below show the phonemic feature of vowel length and tone respectively. According to Tadesse, consonant germination is common. However not every consonant germinate: consonants such as /z/ /r/ and /w/ resist germination and others t, sh, d, P, l, germinate most of the time. Still others f, sh, h, y is realized as either a stop or an afflict when they germinate.

word	meaning
keemo	'three'
keemo	'Prophesy'
kasho	'farm tool'
garoo	'bur'
gotisho	'swamp'
kasho	'ripe'
garoo	'pure'

Table 2.1: *The Phonemic feature of Vowel length and tone*

Short Front	i , e, a
Short Back Round	o, u
Long front up round	ii, ee, aa
Long back round	oo, uu

Table 2.2: *Kafi-Nono Vowels*

2.11.2 Kafi-Nono Dialect

Dialects are Variations of the same language specific to geographical regions or social groups. Although many similarities are shared, there are usually large variations at several linguistic levels; some of these are: phonological, grammatical, orthographic (e.g., "color" vs. "colour" in english) and always different vocabularies. As a result, Neural language models trained or fine-tuned for one certain dialect achieve poorly when tested on a different dialect of the same language [18]. Additionally, systems trained on many dialects simultaneously, fail to generalize well for each individual dialect separately [12]. Inevitably, multi-dialect languages pose a challenge to neural LM systems. If sufficient data is available for each dialect, a good practice is to treat each dialect individually. Otherwise, in cases where dialects are resource-scarce, these models are boosted with data from other dialects.

According to Theil (2007) Kafi-Nono is usually divided into six dialects which he calls, Gimbo, Dechi, Xallo, Manjiyo, Chani, and Geshi.

2.11.3 Kafi-Nono Vowels

Most scholars agree that Kafi-Nono has five long vowels and five short vowels if foreign sounds are excluded, vowel phonemes of Kafi-Nono. Six long and five short vowel phonemes. For the sake of ease let us see by contrasting the vowel phonemes of the three scholars in 2.2.

2.11.4 Kafi-Nono Part of Speeches

In Kafi-Nono Words can be divided into two broad categories: closed class types and open class types [5]. Closed types are those that have moderately fixed morpheme members while open classes are those that continually changed even can be borrowed from other languages. Since Kafi-Nono is under-resourced language, to till, there were no defined tag sets and/or tagged corpus available for research and development. Thus, in consultation with linguists,[5] identified a total of 34 part of speeches for the language. The part of speeches is defined according to the hierarchies of word classes and sub-classes of nouns, verbs, adjectives, pronouns, adverbs, prepositions. In addition to these, conjunction, interjections, numerals and punctuation are also included as basic classes of Kafi-Nono language.

2.12 Related Works

2.12.1 Foreign languages

English Word Prediction

Antal van den Bosch [58] proposed classification-based word prediction model based on IGTREE. A decision-tree induction algorithm has been favorable scaling abilities. Token prediction accuracy, token prediction speed, number of nodes and discrete perplexity are evaluation metrics used for this work. Through a first series of experiments they demonstrate that the system exhibits log-linear increases in prediction accuracy and decreases in discrete perplexity, a new evaluation metric, with increasing numbers of training examples. The induced trees grow linearly with the amount of training examples. Trained on 30 million words of newswire text, prediction accuracies reach 42.2% on the same type of text. In a second series of experiments we show that this generic approach to word prediction can be specialized to confusible prediction, yielding high accuracies on nine example confusible sets in all genres of text. The confusible-specific approach outperforms the generic word-prediction approach, but with more data the difference decreases.

Agarwal and Arora [59] proposed a Context Based Word Prediction system for SMS messaging in which context is used to predict the most appropriate word for a given code.

The growth of wireless technology has provided alternative ways of communication such as Short Message service (SMS) and with tremendous increase in mobile Text Messaging, there is a need for an efficient text input system. With limited keys on the mobile phone, multiple letters are mapped to same number (8 keys, 2 to 9, for 26 alphabets). The many to one mapping of alphabets to numbers gives us same numeric code for multiple words.

Trnka[60] conducted a research on topic Adaptive Language Modeling for Word. AAC devices are highly specialized keyboards with speech synthesis, typically providing single- button input for common words or phrases, but requiring a user to type letter-by-letter for other words, called fringe vocabulary. Word prediction helps speed AAC communication rate. The previous research conducted by different scholars using ngram models. At best, modern devices utilize a trigram model and very basic recency promotion. However, one of the lamented weaknesses of ngram models is their sensitivity to the training data. The objective of this work is to develop and integrate style adaptations from the experience of topic models to dynamically adapt to both topically and stylistically. They address the problem of balancing training size and similarity by dynamically adapting the language model to the most topically relevant portions of then training data. They present the results of experimenting with different topic segmentations and relevance scores in order to tune existing methods to topic modeling. The inclusion of all the training data as well as the usage of frequencies addresses the problem of sparse data in an adaptive model. They have demonstrated that topic modeling can significantly increase keystroke savings for traditional testing as well as testing on text from other domains. They have also addressed the problem of annotated topics through fine-grained modeling and found that it is also a significant improvement over a baseline ngram model.

Word Prediction for Persian Language

Masood Ghayoomi and Seyyed Mostafa Assi[61]studied word prediction for Persian language. In this study, they designed and developed based a system on a Statistical Language Modeling. The corpus contained approximately 8 million tokens. The corpus is divided in to three sections: one was the training corpus that contained 6,258,000 tokens, and 72,494 tokens; the other section was used as the developing corpus which contained 872,450 tokens, and the last section was used as the test corpus which contained 11,960 tokens. The user enters each letters of the required word; the system displays a list of the most probable words that could appear in that position. Three standard performance metrics were used to evaluate the system including keystroke saving, the most important one. The system achieved 57.57% saving in keystrokes. Using such a system saved a great number of keystrokes; and it led to reduction of users effort.

Ghayoomi and Daroodi [26] studied word prediction for Persian language in three approaches. Persian is a member of the Indo-European language family and has many features in common with them in terms of morphology, syntax, phonology, and lexicon. This work is based on bi-gram, tri-gram, 4-gram models and it utilized around 10 million tokens in the collected corpus. Using Keystroke Saving (KSS) as the most important metrics to evaluate systems performance, the primary word-based statistical system achieved 37% KSS, and the second system that used only the main syntactic categories with word-statistics achieved 38.95% KSS. Their last system which used all of the available information to the words get the best result by 42.45% KSS.

Word Prediction for Russian Language

Hunnicutt et al. [56] performed a research on Russian word prediction with morphological support as a co-operative project between two research groups in Tbilisi and Stockholm. This work is an extension of a word predictor developed by Swedish partner for other languages in order to make it suitable for Russian language. Inclusion of morphological component is found necessary since Russian language is much richer in morphological forms. In order to develop Russian language database, an extensive text corpora containing 2.3 million tokens is collected. It provides inflectional categories and resulting inflections for verbs, nouns and adjectives. With this, the correct word forms can be presented in a consistent manner, which allows a user to easily choose the desired word form. The researchers introduced special operations for constructing word forms from a word„s morphological components. Verbs are the most complex word class and algorithm for expanding root form of verbs to which their inflectional form is done. This system suggests successful completion of verbs with the remaining inflect able words.

Word Prediction for Hebrew Language

Netzer et al. [57] are probably the first to present results of experiments in word prediction for Hebrew. They developed a NLP-based system for Augmentative and Alternative Communication (AAC) in Hebrew. They used three general kinds of methods: (1) Statistical methods: based on word frequencies and repetition of previous words in the text. These methods can be implemented by using language models (LMs) such as the Markov model, and unigram/bigram/trigram prediction, (2) Syntactic knowledge: part of speech tags (e.g. nouns, adjectives, verbs, and adverbs) and phrase structures. Syntactic knowledge can be statistical-based or can be based on hand-coded rules and (3) Semantic knowledge: assigning categories to words and finding a set of rules that constrain the

possible candidates for the next word. They used 3 corpuses of varying length (1M words, 10M words, 27M words) to train their system. The best results have been achieved while training a language model (a hidden Markov model) on the 27M corpus. They applied their model on various genres including personal writing in blogs and in open forums in the Internet. Contrary to what they expected, the use of morpho-syntactic information such as part of speech tags didn't improve the results. Furthermore, it decreases the prediction results. The best results were obtained using statistical data on the Hebrew language with rich morphology. They report on keystroke saving up to 29% with nine word proposals and 34% for seven proposals, 54% for a single proposal.

2.12.2 African Languages

Melinda Loubser and Martin J. Putt kammer [89] has conducted the possibility of neural networks on low-resource south African languages. In their thesis, the viability of neural network implementations of core technologies (the focus of the paper was on text technologies) for 10 resource-scarce South African languages is evaluated. Accordingly, Neural architectures that performed well on similar tasks in other settings(high-resource) were implemented for each task and the performance was assessed in comparison with currently used machine learning implementations of each technology. The neural network models evaluated perform better than the baselines for compound analysis, are viable and comparable to the baseline on most languages for POS tagging and NER, and are viable, but not on par with the baseline, for Afrikaans lemmatization. The research gap in [89] is that Neural architectures that performed well on similar tasks in high resource settings were directly implemented for each task and the performance was assessed in comparison with currently used machine learning implementations of each technology. Large-scale transformer-based language models such as ELMo, BERT [90], GPT2 [62], Grover, and CTRL [91] have produced state-of-the-art results on many NLP tasks and proven to be capable of generating highly fluent text passages. For example, GPT-2 has been used to generate patent claims and to power task-oriented conversational dialogues [92].

2.12.3 Local languages

Locally the most widely used type of language models are the corpus-based probabilistic(statistical) ones. These models provide an estimate of the probability of a word sequence based on training data. Therefore, large amounts of training data are required in order to ensure statistical significance. But even if the training data are very large, it is impossible to avoid the problems of data sparseness and out-of-vocabulary (OOV) words.

These problems are particularly serious for languages with a rich morphology, which are characterized with high vocabulary growth rate and a correspondingly high perplexity of their language models. Since the vocabulary size directly affects system complexity, a promising direction is towards the use of Neural network approach in language modeling) [93]. So far there are few works are done on word prediction on local languages and all of them are based on statistical methods: Tigist Tensou [94] conducted a research on word sequence prediction for Amharic. In her work, statistical methods and linguistic rules are used. Statistical models are constructed for root/stem, and morphological properties of words like aspect, voice, tense, and affixes. Word sequence prediction using a hybrid of bigram and tri-gram model offers better keystroke savings in all scenarios for her experiment. For instance, when using test data disjoint from the training corpus, 20.5%,17.4% and 13.1% keystroke savings are obtained in hybrid, tri-gram and bi-gram models respectively. Evaluation of the model is performed using developed prototype and keystroke savings (KSS) as a metrics. According to their experiment, prediction result using a hybrid of bigram and tri-gram model has higher KSS and it is better compared to bi-gram and trigram models separately. Nesredin Suleiman [95] performed a research on word prediction model for Amharic online hand writing recognition. In his work, he used a corpus of 131,399 Amharic words and extracted statistical information to determine the value of N for the N-gram model, where the value two (2) is considered as a result of the analyses. His Experiment shows a prediction accuracy of 81.39% The analyses used to get information like the average word-length of Amharic language; the most frequently used Amharic word-length and the like have been used to decide which for N-gram model to use best, where N is the number of characters after which the prediction process starts. Ashenafi Bekele Delbeto [96], researched on Afan oromo next word predictor using bi-gram and tri-gram statistical methods, the same approach as [95] and [94]. Additionally, as [96] stated, he has employed stemming and POS tagging for capturing the morphological features of Afan oromo language. According to the evaluation, the primary word-based statistical system achieved 20.5% KSS, and the second system that used syntactic categories with word-statistics achieved 22.5 that, statistical and linguistic rules have good potential on word sequence prediction for Afaan Oromo.

2.13 Suitability Assessment of Reviewed Methods

In this section the use of previously reviewed word prediction methods suitability for the target languages is discussed. So, the key question is: Are the word prediction methods that we have previously discussed useful for inflected and low resource languages? As we mentioned, in non-inflected languages it is feasible to include in the dictionary all the

forms derived from each lemma, taking into account that the number of variations is quite small. For instance, in English friends is the only variation (without creating composed words) of friend, and the verbs have a few variations too. In Kafi-Nono, the word nuuchoo (with the same meaning than friend) may vary in gender and number, giving the words: nuuchee, nuucheena'o and nuuchittino. Here the variations that the word nuuchoo may have in Kafi-Nono which has more than 90 forms, making it impossible to store all of them in the dictionary of the system. This is one of the changes to be taken into account for the design of a predictor for this type of languages.

In inflected languages, the complexity in making the changes is very high, because of the number of possibilities. One possibility is to group the suffixes depending on their syntactic function to make it possible to have an easy automation. In addition, we shouldn't forget that suffixes may be recursively concatenated that means suffixes themselves have suffixes. In the previously presented prediction methods, the ones using probabilistic information mainly work with the words as isolated entities. That is, they work seeing each word in the dictionary as a whole to be guessed, without taking into account the morph-syntactical information inherent to the languages. So, a word that is not at the lexicon cannot be guessed. The impossibility to store all the combinations of a word, make these methods not very suitable for inflected languages. Therefore, it would be very interesting to treat the entire sentence. Then, the statistical syntactic approach is not very useful, because it only takes into account the previous word. And the syntactic with grammars is very hard to implement, because of the number of variations a word may have. Maybe a great number of rules have to be defined to cope with all the variations, but in this way the probabilities to guess the rule which is being used are very small, because of their variety. The same thing happens with the semantic approach, which has, as it has been said before, the same procedural characteristics as the grammar based syntactic one. So, the complexity needed to create a correct word, including all the suffixes it needs, in inflected languages may make it necessary to search for other prediction methods, apart from each that are shown statistical methods.

When we come to the traditional NNMs, such as FFNN, LSTM and RNN the cost of training a new Neural model is also very high for most languages in terms of Computing resources(power) and Language resources. Their performance is very related to the size of data and they need a huge amount of data to for showing their highest performance. Hence, in this work, we have explored a cost-effective approach-cross-lingual transfer learning method to adopt a strong source language pre-trained transformer model, trained from a large monolingual corpus to a low-resource language of our interest, Kafi-Nono.We found the transformer being the most suitable in overcoming the shortcomings of all statical and traditional NNM specially for the language we are studying.Transformer overcomes the data need gap by leveraging the cross-lingual transfer technique and the morphologi-

cal behaviour through its architectural arrangement. The transformer model selected for this work is Generative Pre Trained(GPT2) version 2. All the shortcomings of traditional NNM and advantages of transformer are mentioned in section 2.5 and 2.6 above.

2.13.1 Summary

In this section, we have discussed works related to word sequence prediction for different languages. We understand that languages have their own linguistic characteristics requiring specific approaches to word prediction. Hence, the research conducted on one language cannot be directly applied to other languages. Therefore, the aim of this study to design and develop word sequence prediction model for Kafi-Nono by taking the unique features and resources of the language into consideration.

2.14 Proposed Approach

Based on related works reviewed in this Section , different researchers used different approaches to develop language models. Researchers have their own point of view with evidence to apply a particular approach in such domain. In this Section we need to highlight about the approaches used in this research work and the reasons behind it before we are going to the system architecture. Kafi-Nono is a language that can be characterized as both Morphologically rich(inflected) and Digitally low-resource. For handling these two problems, neural deep learning method (for handling inflection) and pre-trained transformer model (for handling low-resource problem) system is proposed. The proposed model is biasing on the deep learning approach which involves learning words and word embedding using The first step in the deep learning phase was to somehow convert these mentions into a feature vector. After feature extraction, a position embedding vector is created and added to the word embedding vector. Then we could feed these features to a variety of neural network layers and see how they perform. We started by trying to build our own word encoding using Byte-Pair encoding (BPE) and train it using a hugging-face library.

CHAPTER THREE : Research Methodology

Research methodology is a systematic way to solve a problem. It is a science of studying how research is to be carried out. Essentially, the procedures by which researchers go about their work of describing, explaining and predicting phenomena are called research methodology. It is also defined as the study of methods by which knowledge is gained. Its aim is to give the work plan of research.

3.1 Research Design

In a experimental design, the researcher tests his or her intervention in an educational setting such as a Laboratory, makes modifications depending on the data collected, and conducts the intervention until it produces good results. The data collected is in any form of work that will show that the student has learned what is expected. In the design experiment, modifications are made over time.The design experiment adapts the intervention to suit the setting through iteration.Experimental design methods allows the researcher to understand better and evaluate the factors that influence system by means of quantitative and/or qualitative approaches. Such approaches combine theoretical knowledge of experimental designs and a working knowledge of the particular factors to be studied. Experimental research is a scientific technique to study that focuses on creating research with a high level of internal and external validity. The correctness of assertions about cause and effect relationships is referred to as causal validity. This type of research will yield results that can be validated by experiment or observation. This thesis is based on an experimental study design, which includes data preparation, preprocessing, evaluation metrics, and analysis as basic procedures.

3.2 Literature Review

The research study started with a critical literature review in order to establish an in-depth understanding of the landscape of the Word Prediction. A number of related works and resources are reviewed. This consists of thesis, conference and journal articles, white papers and word prediction systems developed for other languages. The large portions of reviewed materials are conference and journal articles. The nature, background history

and operational function of word prediction systems are studied. In addition, a discussion is made with Kafi-Nono Linguistic experts regarding the linguistic nature of the language like the grammatical structure and morphology of Kafi-Nono.

3.3 Data collection and preparation

This study conducted to design a WSP language model and implement for Kafi-Nono Language. Unfortunately, Kaf-Nono is a low resourced language.Kafi-Nono does not have a publicly available text corpora resource for such studies. For this reason,we have prepared our own corpora for this thesis work implementation. The corpora preparation started from collecting web based digital books(holly bible),Text books in digital form(PDF), randomly selected word, sentence, and paragraph level Kaf-Nono text files from social media(Facebook).We have used a web scrapping tools such as HTTrack to collect data from websites. The collected data is needed to be preprocessed before it is ready to be fitted to train the model.Language models have their own data preparation technique slightly different in terms of embedding dimension and feeding batch size. For Our pretrained language model we have made the preprocessing includes normalization, segmentation and tokenization.The details are given in section 4.2.

3.4 Design and Implementation Approach

Transfer Learning in Language modeling involves fine-tuning hyper-parameters and re-training a selected pretrained model. In this thesis we have selected a transformer based pretrained English Model to make it learn the Kaf–nono language pattern and use it for word prediction in kafi-nono language. We have selected the English Version of pretrained model for it resembles in character and grammatical structure with Kafi-Nono. The architecture, module, and components of the proposed system were defined. Performance metrics, training approach and parameter tuning were specified. Required tools, algorithms, programming languages were identified, chosen, and studied. To develop the system we have used different software and hardware tools.We have used python programming language for development. Benefits that make Python the best fit for our thesis work include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language.

3.5 Evaluation

After the system is developed in order to check whether our study works in accordance with the ideas and theories of word sequence prediction,It is evaluated for it performance. There are common evaluation techniques for WSP models, such as accuracy,perplexity, bits-per-character, and cross entropy. The most widely used evaluation metric for WP language model is Perplexity. We have implemented evaluation using two methods: automatic (perplexity) and human evaluation for accuracy. We have used two evaluation methods in order to be sure that the automatic and manual evaluation results are consistent on our data and our systems performance.More Can be found on section 2.9 and 5.10.

4.1 Overview

In this chapter we discuss about the proposed models of our work along with the design constraints and implementation issues. The main components of word prediction model along with sub components and the interaction between the cooperative components will be presented. As described in literature and background, Kafi-Noonoo is a language with scarce data in text form. This scarcity of those resources makes the language not to be incorporated in the field of the Natural Language Processing (NLP) and information extraction (IE). By taking this in consideration and reading different supportive articles, journals and books, we selected a cross-lingual transfer method being most suitable to model kafinoonoo language and predict next word. We used books, reports and cultural documents of the language corpus as our data source. As discussed in chapter two various approaches have been proposed to predict next word of different languages. The models depend on the characteristics of the language and the available amount of data. As a result, it is difficult to apply the models proposed for other languages to Kafi-Noonoo directly. Here, we have proposed a new fine-tuned model for Kafi-Noonoo called KNGPT2. This task requires us to build a system which can predict next words with corresponding referent words.

The main goal of this work is to present a simple Transformer-based NNL model called KNGPT2. It is built by re-training and hyperparameter optimization of pretrained transformer model called GPT2 that can be used for word prediction in autocompletion tools, scoring the "rightness" of sentences, or generating text. The model has been developed using a Python library called Pytorch, which allows developers and researchers to modify neural networks and tune the training process effortlessly. This effortless tuning is essential for both research and industry since the models have to be trained with multiple different configurations to figure out how to develop the best performing language models. One part of this LM development process includes so-called hyper-parameter optimization or fine-tuning, where the different default parameters are set for the model and observed which parameters perform the best [22].

In addition to the flexibility, Pytorch is optimized to utilize multiple GPU and CPU cores to speed up and parallelize the massive numerical computation. Computational optimization is critical to keep the research and product development iteration cycle as

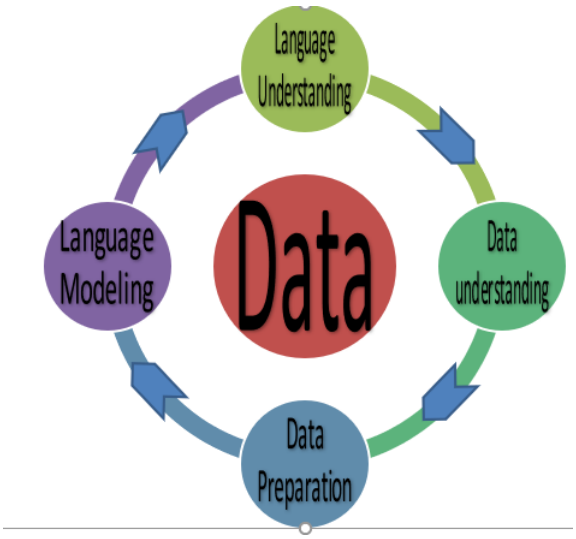


Figure 4.1: *The General Method of Language Modeling*

fast and productive as possible.

GPU utilization is essential because the training times of neural networks can be 10 to 100 times faster on GPUs than CPUs [29] Concerning parallel computation, even with the most efficient GPUs, the training time of complex and computationally heavy models can take multiple days [31]. Hence, it is crucial to parallelize the computation to multiple GPUs. Then, it is still possible to decrease the computation time significantly even if there are no possibilities to improve the computation time by optimizing the code or hardware.

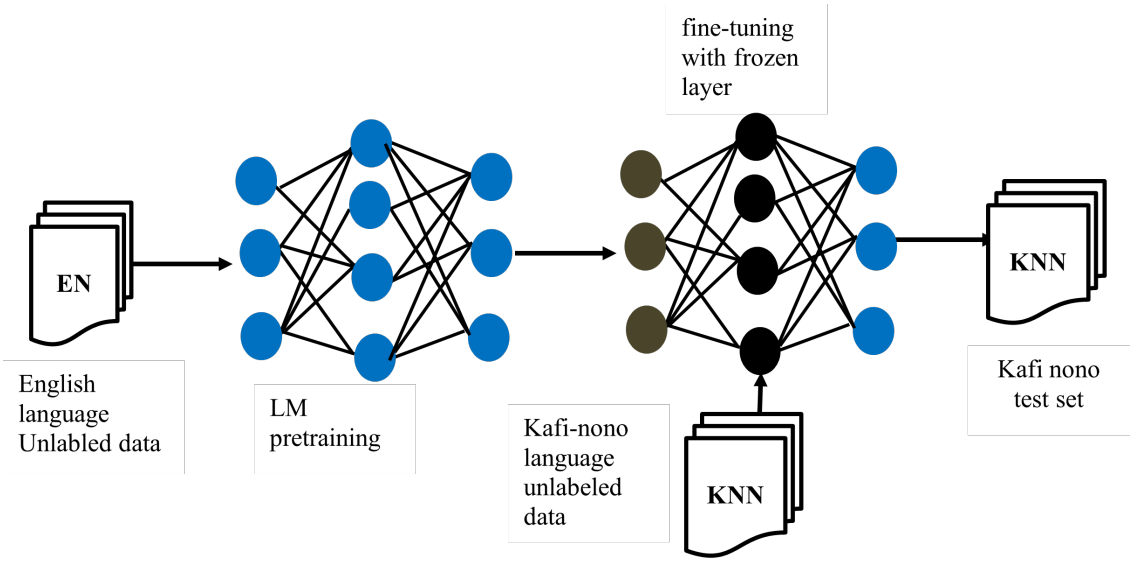


Figure 4.2: *The Fine-Tuning Process- LM was first trained with huge English language Unlabled data (LM pretraining) and then fine-tuning and training on top of the frozen embeddings with Kafi-nono language unlabeled data*

Data Source	No. of Sentences	No.of Words	No. of Tokens
New Testament Text	5100	14,500	34,300
Educational text Books	3800	12,700	25,800
Social Media	1800	7,200	14,900
Total	10,700	214400	75,000

Table 4.1: *Data set Statistics*

4.2 Data Preparation

Data is the fuel of the machine learning age. There are no existing Kafi-Noonoo corpora for any machine learning tasks and hence language modeling tasks. Thus, it is necessary for us to create a corpus for this study. Different books are chosen as sources of data.

One of the quality of data is its representativeness. Books are good sources of data since it represents structured content well, however books are very domain-specific depending on the plots and settings. The educational books contents are written for ease of understanding. So, we have collected data from three main sources these are the new testament holy bible in Kafi-Noonoo language, Educational Text books and social dialogues (this data was extracted from an interview made by an international social researcher in some part of Kafa zone). Hence we have considered data representativeness.

Many of the books we have collected were full of duplicate contents. These duplicate data were removed to insure quality of data.

The need for large amount of data by deep neural networks is reduced by different techniques and tools in order to reach low-resourced languages. One of the techniques is using cross-lingual transfer learning, where a pre-trained language model in another language (e.g. English) is fine-tuned in to a new-language. In this approach, without requiring gigabits of data we can get the advantage of neural networks in to resource scarce languages

. We have used different tools and techniques to extract Kafi-Noonoo data. One of such tools is the HTTRACK, which is used to dumb websites to a local storage. And text collector tools to collect text data from the offline repository, downloaded earlier. Theses tool was used to extract a non-downloadable text corpus of the new testament bible in Kafi-Noonoo language. The statistics of the dataset is given in Table 4.1 The dataset also covers a variety of topics in the books of the New Testament Holly bible in Kafi-Noonoo. We have formatted (segmented) the data so that the beginning and end of the text is used for training and evaluating our models. To give our AI model the most uncluttered and

High quality learning data possible, we have used a series of simple markers to give GPT-2 the shape of a writing prompt response. we added <|sos|> for indicating start of sentence and <|eos |> to denote end of sentence tokens to match the writing prompt dataset, while this was a huge, time-consuming effort, but it made the output infinitely more interesting. We have saved the whole dataset as a .TXT file.

4.2.1 Data Processing

Collected Kafi-Noonoo corpus needs different type of preprocessing before we made ready for word prediction system, as which Kafi-Noonoo has different language specific features that should be normalized. The document preprocessing component such as, tokenization, sentence segmenting, character normalization, stop word removal etc. are applied on the dataset to handle language specific issues.

4.2.2 Sentence Segmenting

When we process a batch of sentences, they aren't always the same length. This is a problem because IDs, the input to the model, need to have a uniform shape. segmenting is a strategy for ensuring IDs are rectangular by adding a special padding token to sentences with fewer tokens. This is done by setting the padding parameter to True to pad the shorter sequences in the batch to match the longest sequence. We need segmenting because we wanted to give the AI the most uncluttered and high-quality learning data possible, so we used a series of simple markers to teach GPT-2 the shape of a writing prompt response.

We use pad_sequences for padding. pad_sequences uses arguments such as sequences, padding, maxlen, truncating, value and dtype.

Sequences: list of sequences that we created from the data corpus.

Padding: 'pre' or 'post (default pre). By using pre, we pad (add 0) before each sequence and post pad after each sequence.

maxlen: maximum length of all sequences. If not provided, by default it will use the maximum length of the longest sentence.

Truncating: 'pre' or 'post' (default 'pre'). If a sequence length is larger than the provided maxlen value then, these values are truncated to the maxlen. 'pre' option will truncate at the beginning whereas 'post' truncate at the end of the sequences.

Value: padding value (default is 0)

dtype: output sequence type (default is int32)

4.2.3 Tokenization

This is the task of breaking texts in to piece of meaningful tokens. Sometimes it can be defined as a arrangement of characters or a defined text unit. Tokenization is the job of slicing sentences(text) up into pieces, possibly at the same removing certain characters such as punctuation. A token is an example of a sequence of characters in some text document that are arranged together as a meaningful semantic unit for processing. The tokenizer then tokenizes all the text segments which have space between each other as independent token. We used Byte-Pair Encoding (BPE) word tokenizer for this task.

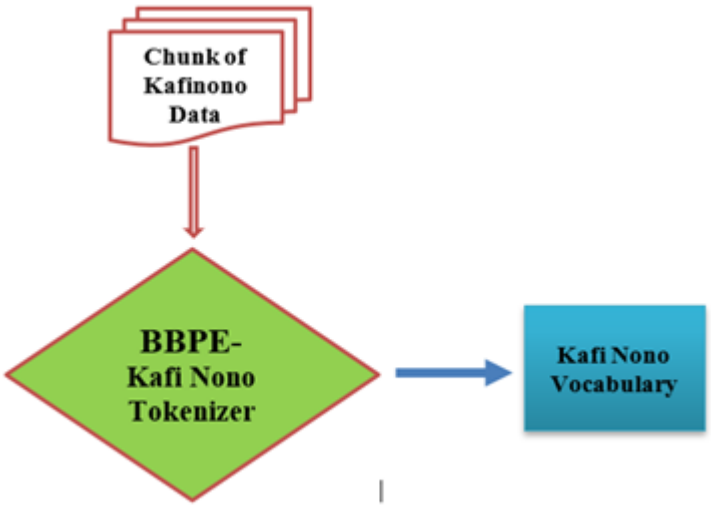


Figure 4.3: *Kafi-Nono tokenizer*

4.3 The Pre-Trained Transformer Model

We use a Transformer GPT-2 based architecture for our prediction LMs. The model largely follows the details of the OpenAI GPT model [32]. GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages. GPT-2 is trained with a objective to predict the next word, given all of the previous words within some text. The diversity of the dataset causes this simple goal to contain

naturally occurring demonstrations of many tasks across diverse domains. GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data. Zero-Shot Transfer: The pre-training task for GPT-2 is solely language modeling. All the downstream language tasks are framed as predicting conditional probabilities and there is no task-specific fine-tuning. The Pre-Trained Transformer Model GPT-2 architecture implements a deep neural network, in which input is processed by multiple layers of neurons whose weights determine the activation patterns of subsequent layers, and the final layer of neurons constitutes the output of the network. The Internal Configuration of the model is:

"activation_function": *"gelu_new",*
"architectures": [
"GPT2LMHeadModel"
],
"attn_pdrop": 0.1,
"sos_token_id": 50256,
"embd_pdrop": 0.1,
"eos_token_id": 50256,
"initializer_range": 0.02,
"layer_norm_epsilon": 1e-05,
"model_type": "gpt2",
"n_ctx": 768,
"n_embd": 768,
"n_head": 12,
"n_layer": 12,
"n_positions": 768,
"n_special": 0,
"predict_special_tokens": true,
"resid_pdrop": 0.1,
"summary_activation": null,
"summary_first_dropout": 0.1,
"summary_proj_to_labels": true,
"summary_type": "cls_index",
"summary_use_proj": true,
"task_specific_params":
"text-generation":
"do_sample": true,
"max_length": 50
"vocab_size": 50257

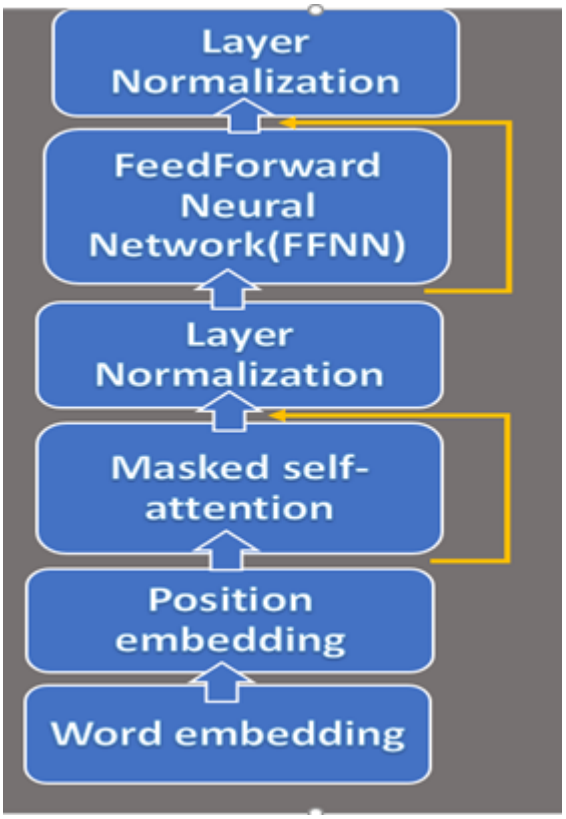


Figure 4.4: *The pre-trained model (GPT2) Structure*

4.4 Proposed System Architecture and Components

In general, the following are some of the general components of Kafinono Next word prediction system.

4.4.1 Data Preprocessing Component

This component allows the system to identify language specific aspects and to normalize the document in order to save the CPU cost, space, and to enhance the system performance.

4.4.2 The Transformer Component

The transformer is the main component where the neural network operations are done to model the patterns of the language. It contains both the Embedding Components and the neural components. Embedding Components extract both word-wise and position-wise features. The word and position embedding vectors are added up in this component and passed to the next component.

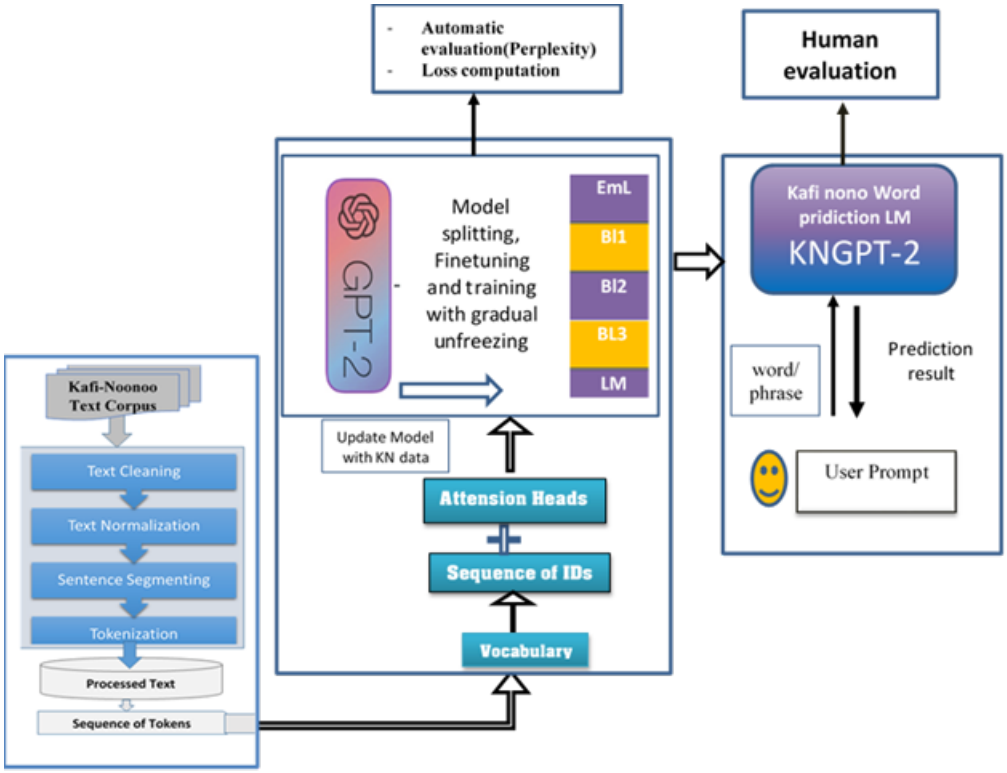


Figure 4.5: *The proposed system architecture*

The proposed system architecture of our next word prediction system is adopted from the general architecture of transformer based natural language generation and prediction system. Our own system specific components are incorporated along with the general architecture. Figure 4.5 renders our proposed system architecture.

4.4.3 Text Cleaning

Cleaning data from undesired characters and spaces is necessary for quality of data. We have used Beautiful Soup for cleaning our text and find all paragraphs objects that we don't want. From this point, we can already use regex to return the string obtained

by replacing the leftmost non-overlapping occurrences of the pattern (“</?p[ŝ]*>”)in the string by the replacement (“”). In this way, we clean the text from undesirable characters.

4.4.4 Text Normalization

Normalization is a set of operations applied to a raw data to make it less random or cleaner. Common operations include stripping white space, removing accented characters or lower casing all text. Unicode normalization, is also a very common normalization operation applied in most tokenizers.Each normalization operation is represented in the Tokenizers library by a function called normalizer, and we can combine several of those by using a the Sequence function. Here we used a normalizer applying unigram normalization(Characters are decomposed by canonical equivalence, and multiple combining characters are arranged in a specific order) and removing unicode characters which don’t belong to Kafi-Nono language.

ALGORITHM 4.1: Text Normalization Algorithm with

Require: Input: set of strings D, target vocab size k
Ensure: procedure UNIGRAMLM(D; k)
$V \leftarrow$ all sub-strings occurring more than once in D (not crossing words)
while $|V| > k$ **do**
▷ Prune tokens
Fit unigram LM ∂toD
if $t \in V$ **then**
▷ Estimate token ‘loss’
$L_t \leftarrow p_{\partial}(D) - p_{\partial^t(D)}$
where ∂_t is the LM without token t
end
Remove $min(|V| - k; [a|V|])$of the tokens t with
highest L_t from V , where $a \in [0, 1]$ is a hyper parameter
end
Fit final unigram LM ∂ to D
return V, ∂
EndProcedure

This uni-gram LM text normalization method begins with a super-set of the final vocabulary, pruning it to the desired size see Algorithm 4.1.It takes the vocabulary V and unigram LM parameters ∂ and performs Viterbi inference to decode the segmentation with maximum likelihood under ∂. This method is similar to Morfessor's unsupervised segmentation without its informed prior over token length.

4.4.5 Tokenization and Input Encoding

The word vectors used for the first layer of GPT-2 are not simple one-hot tokenization. We have used Byte pair encodings (BPE). Byte-Pair Encoding (BPE) tokenization is unsupervised tokenization, in which the most frequently occurring pair of Unicode characters is recursively replaced with a character that does not occur in the vocabulary – these approach is adopted by various contextual language models in NLP.

BPE effectively bridges between word level inputs for frequent symbol sequences and character level inputs for rare symbol sequences.The byte pair encoding scheme compresses an arbitrarily large tokenized word list into a set vocabulary size by recursively keying the most common word components to unique values. Before token/word is fed to the model, it will be embedded in the model's embedding component. BPE is Subword tokenization algorithms consisting of two components: a vocabulary construction procedure, which takes a corpus of text and returns a vocabulary with the desired size, and a tokenization procedure, which takes the built vocabulary and applies it to new text, returning a sequence of tokens. In theory, these two steps can be independent, although for the algorithms we examine the tokenization procedure is tightly coupled to the vocabulary construction procedure. A BPE vocabulary is constructed as in Algorithm (4.2):

BPE tokenization takes the vocabulary V containing ordered merges and applies them to new text in the same order as they occurred during vocabulary construction. The WordPiece algorithm, closely resembles BPE. However, instead of merging the most frequent token bigram, each potential merge is scored based on the likelihood of the language model to be trained on a version of the corpus incorporating that merge.

ALGORITHM 4.2: BPE Kafi-Nono Tokenization Algorithm

Require: Input: set of strings D, target vocab size k
Ensure: **Procedure** BPE(D; k)
$V \leftarrow D$ all unique characters in D
(about 10,700 in Kafi-Nono corpus)
while $|V| < k$ **do**
// Merge tokens
$t_L, t_R \leftarrow$ most frequent bi-gram in D
$t_{NEW} \leftarrow t_L + t_R$ /Make new token
$V \leftarrow V + [t_{NEW}]$
Replace each occurrence of t_L, t_R in D with t_{NEW}
end
return V
EndProcedure

5.1 Overview

This chapter presents the experiment performed with the proposed methodology and the findings.

In the first part of this section, the dataset,the experiments environment specification and the experimental parameters will be discussed. Then experimental results are presented and discussed. In the result section, the performance of the proposed algorithm is evaluated against different methods.

5.2 Dataset and Data Processing

Our training data set contains 10,700 sentence segments of the Kafi-Nono text. All of these are the first and new data sets, which has never been seen by any machine learning models before. There are three steps in our data pipeline. The first step is fresh data collection. The second step is to split long text into segment lengths. The third step is to encode the text segments in the required GPT-2 format to digest it. The first step is described is Chapter 3 above. The latter two steps, second and the third, are presented in more implementation details as below. At the second step, we implemented the segment breakdown manually. Our heuristic-based implementation is based on the observation that line segments are often headed by punctuation. The punctuation mark is either full stop or question mark or exclamation mark most of the time. It is conventional to separate a Kafi-Nono text in this way. The punctuation marks alone are not enough to split a text into segments, however. After identifying text segments, we follow Woolf 's code [62] to add "<|sos|>" to the beginning of a sentence and "<|eos|>" to the end. In this way our training dataset is prepared in a specific format. Later at the inference stage, we can also identify the beginning and the end of a sentence in all generated text. The third stage of our data pipeline is straightforward. We use the encode function in the GPT-2 code and transform text data into compressed NumPy format (*.npz). The format is ready for training iterations and saving time. We shared our training data in both numpy format and plain text format. Future researches can reuse our formatted text file for GPT-2. we added start of text <|sos|> and end of text <|eos|> tokens to match the

writing prompt dataset, but we also made sure each response had the original writing prompt marked. While this was a huge, time-consuming effort, but it made our output infinitely more interesting. we saved the whole dataset as a .TXT file. Here's what the sample dataset looks like the following We also reserved some text for user testing on the models performance. The words for user testing are just randomly parted from the main dataset.

<|sos|>Yeeri taan bi ceeggitoomon ittoshicha getee taach immati shuunoomon, ceeni bi qaaroon ta gettemmoyich kittinnee guuphi maddachon tunahoye. </eos|> <|sos|> Hini qaaro wonnee yemeenoochee tiijiqqi shiijareena'o bullich aacheeti maacee mooy- one; tunaballi and ebi ara maacee mooyo Yeerich yaafeetina'oyich gaxxeetone</eos|>
<|sos|>Boonoshichoo Yeeri ebi aacheeti maacee mooyo am shaahoon woriganikkii beeto gaata bi tunoon aagateena'ochi daggooch arichiyoyich shunniye</eos|> ebi aacheeti maacee mooyo Yeerichi oogee yiiroon danee gibano tunati Kiristoosi ittoshi mullee daggooch beemone</eos|>
<|sos|>Asho bulli bi ikki ikkoo Kiristoosina biich beeti ikkittinoona bedditi ashoon bi tunatee Yeer waan giddiyoyich ariyoo bullina asho bullin kaaraa ciiccaabee arichii beeto noone</eos|>

5.2.1 Data Partitioning

To evaluate our model, we have divided the dataset in to training,and evaluation parts each with a dataset of 80% and 20% respectively. We have evaluated our fine-tuned model-KNGPT2 against the evaluation and testing dataset. Test Dataset is the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets). The test set is generally what is used to evaluate competing models. Many a times the validation set is used as the test set, but it is not good practice. The test set is generally well curated. Our test set contains 51 randomly selected words/phrases carefully sampled data that spans the various classes that the model would face, when used in the real world.Our test data is not included in the training and evaluation dataset; hence is prepared from new(unseen) data. The prototype was tested with these test data along with human evaluation process. Since we are using the prototype to generate next words for human evaluation, we don't need separate testing for user rating.

5.3 Experimental Setup

One of the shortcomings of deep neural networks is their greedy behavior both in terms of computing resources and data. They are not effective for ordinary computing resources like laptops or desktops we use in our regular times. For this reason, we used Google Collaboration [88] for GPU. Google Collaboration is a Jupyter based notebook environment that entirely runs in the cloud. Though Google Colab is freely available, with limited use of only 12 continuous hours in a session. For our experiments it is sufficient. For training tasks that require more than 12 hours, we can save the training checkpoints in training to our google drive and restore them for continuous training in next session. Manual effort is needed to initialize a fresh session on Colab, in which the previously saved check points are loaded in to the Colab and setting the paths to the environment. Though it is time consuming, such a free accessible platform may make it easier for scholars to try many different experiments.

Huggigng Face and Fastai(facebook) companies made a way to easily access the huge models and complex data structures which we use in this work. Both of them made it convenient to easily download and acess GPT-2 models and tokenizers as easily as calling a library. Python programming language and Pythorch libraries are used to implement the proposed system.

Google Colab offers free GPUs and TPUs for researchers. Since we will be training a large neural network it is the best opportunity to take advantage of this since we have access to a GPU, otherwise training will take a very long time. A GPU can be added by going to the menu and selecting: Edit > Notebook Settings > Hardware accelerator > (GPU) The GPU available on Colab is NVIDIA Tesla T4 equipped with roughly 15GB RAM memory. The memory size is sufficient for finetuning all layers of the small model (124M)but it is not sufficient for the medium (345M) and large model (747M). It is familiar that TPU (Tensor Processing Unit) which is more powerful than GPU, is also accessible on Colab. With the data-parallel implementation, pure computation time T_batch mini-batch step remains constant in the number of worker GPUs. The amount of data processed during one mini-batch step increases linearly with the number of engaged workers N. Synchronization between workers performed by means of a tree-like allreduce, would yield logarithmic complexity $T_synkalogN$ Thus, the number of mini-batches would decrease linearly with N giving a following scaling model equation 5.3

$$T_epoch = \frac{1}{N}.(T_batchxT_sync) = \frac{1}{N}.(A + B.log(N)) = O(\frac{log(N)}{N} \quad (5.1)$$

Overall, the model architecture, tokenization, and training procedure produce a large number of hyper-parameters that must be tuned to maximize predictive performance. These hyper-parameters include numerical values such as the learning rate and number of transformer layers, dimension of embedding space, but also abstract categorical variables such as the precise model architecture or the parameters in the GPT-2 transformer model scales near-linearly as a function of number of transformer blocks, and quadratically with the number of hidden units per block as 5.5.

$$d_x.(|V| + N_ctx) + A.n.d_m^2odel \tag{5.2}$$

The constant A here is defined by the parameters of the MLP part of the transformer. The chosen monolingual GPT-2 models have 12 layers, scaled dot-product attention with 12 heads, and are trained with BPE vocabulary size of 50,000, The rest of the model architecture parameters is summarized in Table5.2.

Currently, the Hugging Face library seems to be the most widely accepted and powerful pytorch interface for working with GPT-2. In addition to supporting a variety of different pre-trained transformer models, the library also includes pre-built modifications of these models suited to our specific task. The library also includes task-specific classes for token classification, question answering, next sentence prediciton, etc. Using these pre-built classes simplifies the process of modifying GPT-2 for our purposes. Next, we install the transformers package from Hugging Face which will give us a pytorch interface for working with GPT-2. This library contains interfaces for pretrained language models of our focus, GPT-2. We've selected the pytorch interface because it give a good balance between the high-level APIs which are easy to use but don't provide insight into how they work and tensorflow code which contains lots of details but which led us into lessons about tensorflow, when the purpose here is GPT-2.

5.4 Training and Test Results

The experiments conducted during this thesis work utilize two different Language models: LSTM constructed from scratch and transfer learning of Pretrained model GPT2. Pretrained one is the one that this thesis is presenting. These two models are compared to each other for the purpose of bench-marking the performances of each models on our target language and available data.

5.4.1 Long-Short-Term Memory-LSTM Model

Long-term short-term memory networks (LSTMs) (Fig. 5.1) are a modified version of periodic neural networks that make it easier to remember past data in memory. Here, the problem of the vanishing RNN gradient is solved. LSTM is well suited for classifying, processing, and forecasting time series based on time lags of unknown duration and trains the model with back-propagation. There are three gates in the LSTM network:

1. Entrance gate: learn what value from the input should be used to modify the memory. The sigmoid function decides which values to pass through 0.1 and the tanh function provides a weighting to the values transmitted, determining the level of their importance in the range from -1 to 1:

$$i_t = a(W_i * [h_{t-1}, x_t] + b_i)C_t = tanh(W_c * [h_{t-1}, x_t] + b_i) \quad 5.3$$

2. Forget gates: learn what parts should be thrown out of the block. The sigmoid function determines this. Consider the previous stateh_{t_1} and the input content (x) and output a number from 0 (skip it) to 1 (save it) for each number in the state of cell $C_{t^{\backsim}{}_1}$:

$$f_t = a(W_f * [h_{t-1}, x_t] + b_f) \quad 5.4$$

3. Output gate: the input and memory of the unit are used to solve the output. The sigmoid function decides which values to pass through 0.1, and the tanh function provides a weighting to the transmitted values, determining their level of importance in the range from-1 to 1 and multiplying by the sigmoid output:

$$o_t = a(W_o * [h_{t-1}, x_t] + b_o)h_t = o_t * tanh(C_t) \quad 5.5$$

The model has 5 layers, the embedding layer with 1x10 dimension, first and second layers with 1x1000 neurons, two dense layers with 1000 neurons and 32536 neurons respectively. Fig. 5.2. IN this LSTM, there was no inherent knowledge of the English language, and thus is trained only with our Kafi-Nono data.

5.4.2 LSTM Model Training

The figure 5.2 demonstrates the structure of the used LSTM .The model consists of the embedding layer, LSTM, and neural layers. The embedding layer is highly important

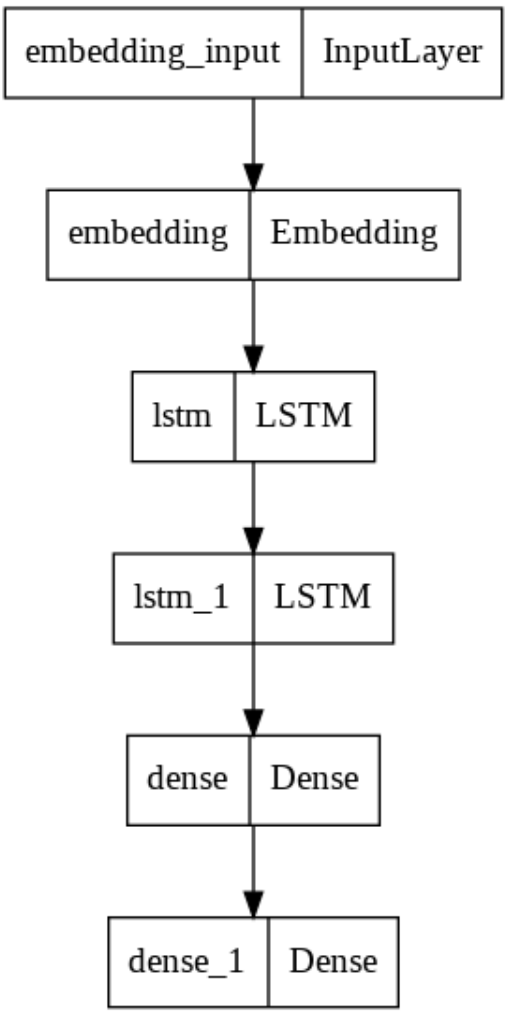


Figure 5.1: *LSTM Model Structure*

for NLP tasks as it helps to achieve better results focusing on keywords. We trained the model with 150 epochs, batch_size of 64,loss function categorical_crossentropy ,Adam optimizer and learning rate of 0.001.The training on the dataset took 12 hours to complete.The training was carried on our kafi-nono data only. Results of model evaluation on

Loss	Accuracy
8.750	0.1455

Table 5.1: *LSTM Model Evaluation result*

the validated data set are displayed in Table 5.1.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding (Embedding)        (None, 1, 10)             325360

 lstm (LSTM)                  (None, 1, 1000)           4044000

 lstm_1 (LSTM)                (None, 1000)              8004000

 dense (Dense)                (None, 1000)              1001000

 dense_1 (Dense)              (None, 32536)             32568536

=================================================================
Total params: 45,942,896
Trainable params: 45,942,896
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.2: *LSTM Model Details*

5.5 Pretrained Model

Currently, there are three versions of GPT-2 transformer model released for public access. The main difference among them is their size: GPT-2 Small, GPT-2 Medium and GPT-2 Large.

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Figure 5.3: *Currently GPT2 publicly available model sizes*

We choose the small sized model for our work because of two reasons: One is the smaller model is less resource intensive than larger models, secondly the largest model is very expensive in terms of hardware resources(memory, GPU, etc) and time required for training it. Moreover, the larger models requires very large memory which we cannot afford for this work. So to get the best out of it, we chose the GPT-2 small for this work. The structural parameters of small-sized GPT-2 English model are: 12 layers of NNs,768 hidden units, 12 Attention heads and 124 million parameters.

5.5.1 Fine-tuning Hyper-parameters

Hyper-parameters top-p, top-k and temperature are known as Fine-Tuning hyper parameters. They are called so because they can be adjusted at any time, before training or after training is completed or during generation time. We have used GPT2-small as the pre-trained language model for model training; In the generation phase, we use the top-p decoding strategy with p = 0.95 to generate 768 tokens at maximum.

Temperature A method to make the distribution P(ww1:t1) sharper, that is, maximizing the chance of high probability words and lessening the chance of low probability, is done by using a parameter called temperature of the SoftMax words. We set the temperature value to different values between 0 and 1. There are less strange words and the output is a bit more coherent now! While applying temperature can make a distribution less random, in its limit, when setting temperature to 0, temperature scaled sampling becomes equal to greedy decoding and will suffer from the same problems as before.

Top-K Sampling a simple, but very powerful sampling scheme, called Top-K sampling was introduced by [97]. In this technique, the K most probable next words are collected and the probability mass is reallocated only among those K next words. this sampling scheme is adopted by GPT2 which was one of the reasons for its success story in text generation. In our experiment, we extended the limit of words used for both sampling steps from 3 words to 10 words to better illustrate Top-K sampling.

Top-p sampling Top-p also known as nucleus filtering sorts words in the vocabulary by descending logit, apply softmax to turn logits into probabilities and keep the top N tokens so that the sum of their probabilities is less than or equal top-p. If we set top-p=0.9 and the network believes the first most likely token has a probability of occurring of 95%, then that will be the only token retained (e.g. N equals 1, as 95% > 90%). On the contrary, if the LM is less sure about its predictions, the top words will likely be associated with similar (low) probs, hence more than one need to be kept to reach a cumulative probability of 90%. The idea here is that if the model is super confident about a specific token to be next, then we just pick it. If instead, the confidence level is low, it makes little sense to select the most probable word, as the second or third tokens will show very similar likelihoods to the top one. If top-p<=0 this filter is not applied Note that the above two filters make sense only if temperature is not set to 0. In this case (temperature=0), as already mentioned above, the script performs greedy sampling, e.g. it always selects the top word by probability Therefore, profiteering a list of tokens has no effect on the final result.

The training Hyper-parameters are used to control the learning process of the model

on how to handle the data and aline with the network architecture accordingly. We have set the learning rate to 6.2510^5, cumulative batch size of 128, learning rate decay of 0.98 per epoch, and categorical cross-entropy loss, epoch of 3 (every epoch has 2500 iteration steps) and optimization algorithm to Adam and an activation function of SoftMax.

5.6 Training Procedure for Pretrained Model

5.6.1 Tokenization and Input Formating

One important part of a language model is the tokenization, which essentially converts the input sequence into tokens that the embedding layer of the model understands and embeds in a meaningful fashion. For this, we implement a ByteLevelBPE tokenizer and train it on our Kafi-Nono corpus.

For building up a vocabulary from our data, the byte pair encoding in language models which tends to work in the opposite direction; it starts out with a set of characters in that language, and through passes on the data, builds up subwords by finding the pairs present in the dataset, and then merging to find larger pairs, and so on. In this way, the tokenizer learns a vocabulary directly from the dataset itself and not from any manual input from an external source.

We are following steps in order to get a ByteLevelBPE tokenizer with a vocab in Kafi-Nono. Get the pre-trained GPT-2 Tokenizer Model pre-trained with an English corpus from the Hugging Face Transformers library. This will give us the tokenizer structure we need and the pre-trained model weights to start training our GPT-2 model in Kafi-Nono from weights already trained in another language. And we Train a Byte-level BPE (BBPE) Tokenizer on the Kafi-Nono corpus by using the Hugging Face Tokenizers library this will give us the vocabulary files in Kafi-Nono of our ByteLevelBPE tokenizer. Then we import the tokenized Kafi-Nonon config files (vocab.json, merges.txt) into the pre-trained BPE, this will give us a ByteLevelBPE structure with the vocab in Kafi-Nono figure **??**.

Update the Embedding Matrix of the PreTrained Model

In the next step we need two things: a fastai tokenizer for data loading, and updating the embedding matrix with Kafi-Nono data which requires converting Hugging Face tokens to fastai tokenizer.The reason we use fastai tokenizer on top of hugging face tokenizer is

"Ġgallatemmonoyichiye":20739,"Ġciftirachi":20740,"Ġcokeeheete":20741,"Ġcobcobbimii":20742,"Ġcoqqireeto":20743,"Ġmarigidi":20744,"Ġmaridino":20745,"Ġgeepphiroona":20746,"Ġgeechenaniye":20747,"Ġgeediyoonichon":20748,"Ġiritichiiniye":20749,"Ġishiichiqqa":20750,"Ġishcichiye":20751,"Ġshigiiyaniya":20752,"Ġxiraakoosee":20753,"Ġbarixemoosi":20754,"Ġsikenduusina":20755,"Ġsikaari":20756,"Ġsixookinon":20757,"woheexonaa":20758,"Ġyihudichoniye":20759,"Ġshooddeeyemmone":20760,"Ġshoolliiyemmine":20761,"Ġshootteyoon":20762,"Ġhemogeeneesina":20763,"Ġarixoobuloosichi":20764,"Ġarifaagoosi":20765,"Ġkiceyeechonon":20766,"Ġgeeciyonalli":20767,"Ġawugisixoosi":20768,"Ġroofoonich":20769,"Ġroobenichi":20770,"Ġsaliimee":20771,"Ġsamootiraaqi":20772,"Ġsapphiiri":20773,"Ġshofiro":20774,"Ġshombaqoon":20775,"Ġceechemmoniye":20776,"Ġmadditonoyichiye":20777,"Ġmaddemmonoyichalli":20778,"Ġaxaaxeeguuchittinoocheenoo":20779,"Ġsaameeliichee":20780,"Ġgiishacaha":20781,"Ġtaggeetochi":20782,"Ġtopphaziyooni":20783,"Ġdooshirati":20784,"Ġshapaalloonoo":20785,"Ġikkotataachesho":20786,"Ġfilippisiyoosi":20787,"Ġmaskootoona":20788,"Ġzemaasinaa":20789,"Ġbogoomone":20790,"Ġgaacaalleene":20791,"Ġdiireto":20792,"Ġbuuroomonaa":20793,"Ġcaammahan":20794,"Ġdaabiloosichoo":20795,"Ġgoobiibe":20796,"Ġgooqqoocee":20797,"Ġshaggahoobolla":20798,"Ġanqeexoosinaa":20799,"Ġantiphaaxiriisi":20800,"Ġmuccahotinalli":20801,"Ġniqolaawiyoosin":20802,"Ġniqaaroonin":20803,"Ġsodoominaa":20804,"Ġsophaaxiroosiyoo":20805,"Ġtegeggabee":20806,"Ġteraahichi":20807,"Ġnuuqqoonoocheenoo":20808,"Ġgogginonooch":20809,"Ġgoggiyoonoyichiye":20810,"Ġshemmataachotinaabiyaa":20811,"hiriggoo":20812,"Ġmalkoosine":20813,"Ġupeeyaniye":20814,"Ġqollehotinalli":20815,"Ġyuliyaasinoyich":20816,"Ġquyeechonalli":20817,"Ġqolaasiyaas":20818,"Ġkiristoosiyooyeerichone":20819,"Ġxiroofininaa":20820,"Ġsaridiyoonina":20821,"Ġalipaanich":20822,"Ġfolookisi":20823,"Ġxirifoosinoyich":20824,"gaawichireeshiye":20825,"Ġbaarinegeesi":20826,"Ġfagiloosinaa":20827,"Ġfaliixoosina":20828,"Ġsusaaninaa":20829,"Ġsuraakuusi":20830,"akalidami":20831,"Ġshemmataachotinaabiyaateeri":20832}

Figure 5.4: *Sample of BPE KN Tokens*

that fastai is more convenient for loading data to the model. Secondly, we will update the embedding matrix of the English GPT2 model with the Kafi-Nono vocabulary on top of English vocabulary. This will set to the embedding weight for further fine tuning during the training phase Figure 5.5.

Creating a Fastai Dataloader

We have tokenized data and the data needs to get loaded to the model. So we need a fastai dataloader. This also re-uses the Kafi-Nono tokenizer we made earlier. The fastai v2 library expects the data to be assembled in a DataLoaders object that has a training and validation data devider. We can do this done by using the dataloaders method in the fastai library. We just have to specify a batch size and a sequence length here

5.6.2 Splitting the Model and Gradual Unfreezing

Transfer learning involves accessing some parts of the transformer model for partial training and leaving others in each training epoch. In this stage we split the pretrained model to train modules separately. Training some part of the model at a time and leaving

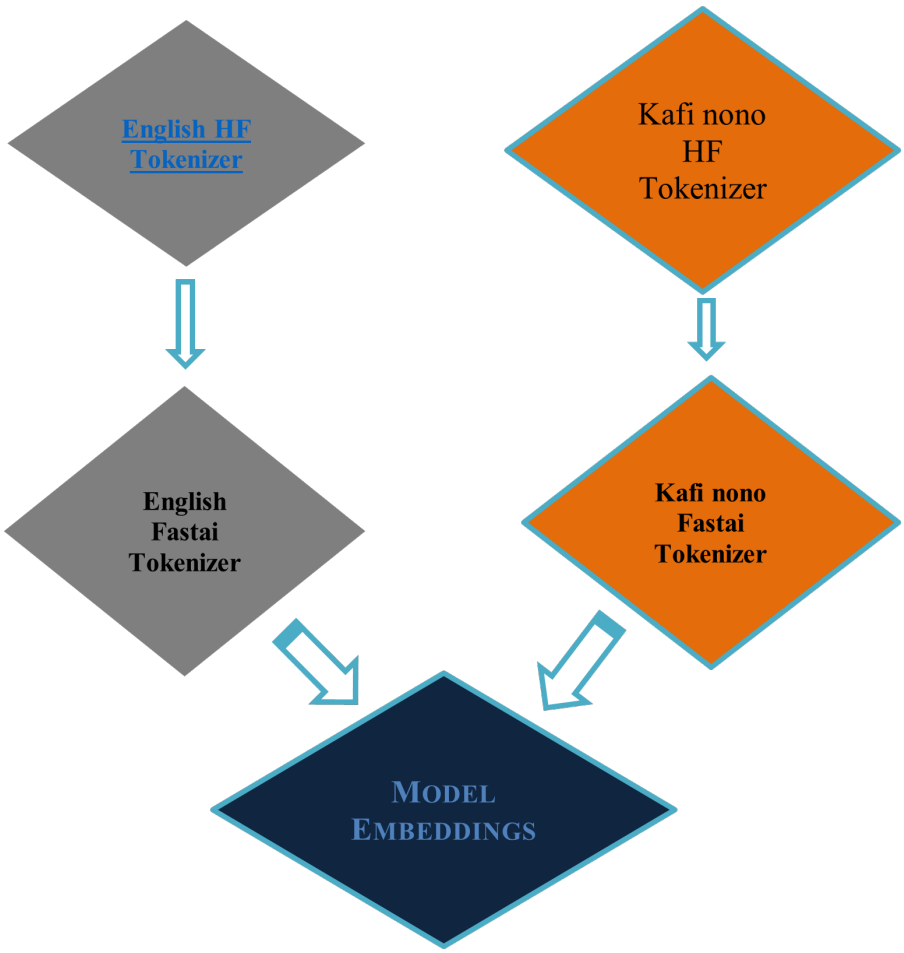
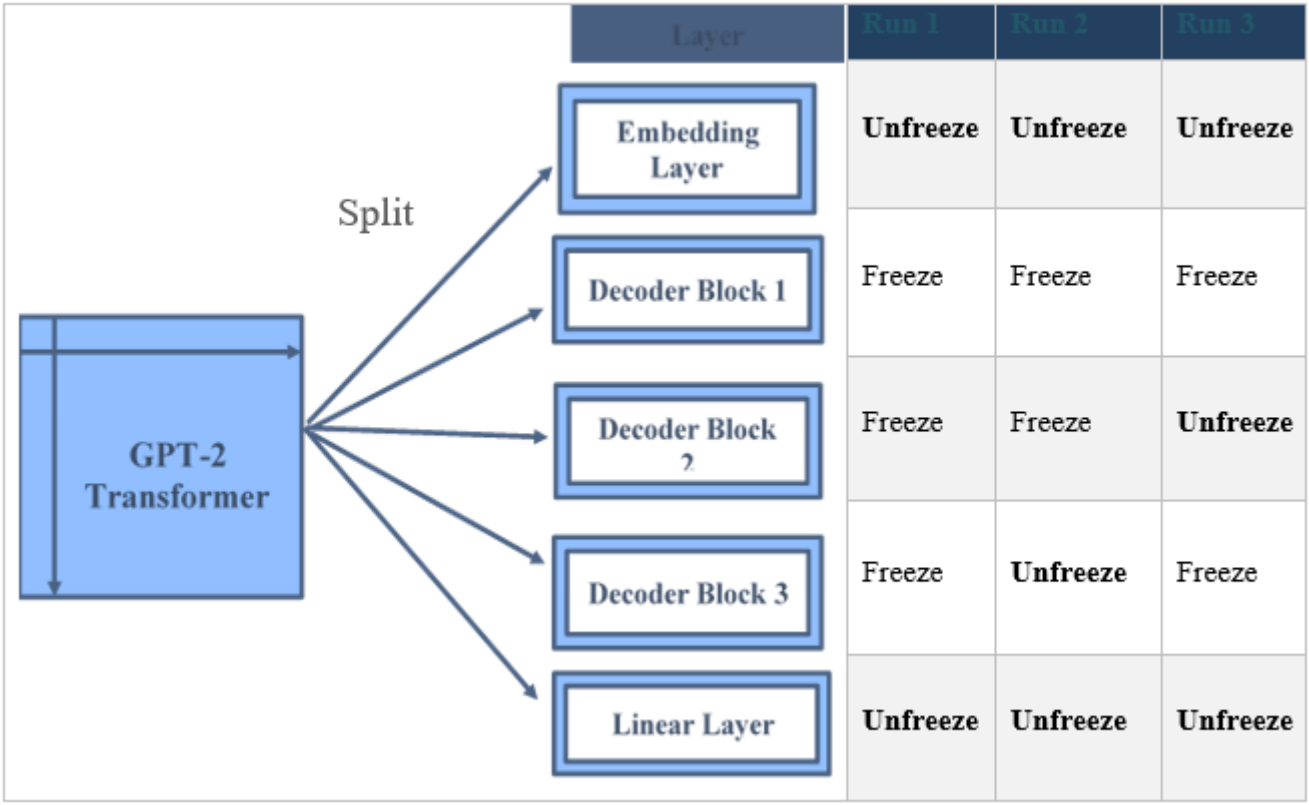


Figure 5.5: *Updating Model Embedding*

the other part left is called gradual unfreezing. The GPT-2 model parts are the embedding layer, 3 decoder blocks(each containing 4 NN layers) and the linear layer(LM-head layer). Generally, the model has 2 main parts (or parameters groups): transformer and lm_head. As we can read in [141], the lm_head is a copy of the vocab embedding matrix (wte) in order to get after the softmax probability of each token in the vocab. Therefore, we need to split all the transformer layers group to get all layers. As such, we finetune the following layers per run:

- Finetuning step 1: Unfreeze(train) The LM head + embedding and freeze others
- Finetuning step 2: Unfreeze(train) LM head + embedding + decoder block 3 and freeze others
- Finetuning step 3: Unfreeze(train) LM head + embedding + decoder block 2 and freeze others

as shown in figure 5.6



Layer	Run 1	Run 2	Run 3
Embedding Layer	**Unfreeze**	**Unfreeze**	**Unfreeze**
Decoder Block 1	Freeze	Freeze	Freeze
Decoder Block 2	Freeze	Freeze	**Unfreeze**
Decoder Block 3	Freeze	**Unfreeze**	Freeze
Linear Layer	**Unfreeze**	**Unfreeze**	**Unfreeze**

Figure 5.6: *Splitting and gradual unfreezing*

Now, we can create our layers groups that will allow us to use all the fastai v2 fine-tuning techniques. Moreover, we decided to follow the fine-tuning method showed for text classification training in the [142] by creating 5 parts: 3 decoder blocks each 4 layers groups , one embedding layer and the linear layer. The underlying idea of this, though difficult to theoretically or experimentally prove, is that the earlier decoder blocks would focus more on semantics and structure, while the later layers focus more on the actual dialect (English, Kafi-Nono, etc.) This split is provided to the final Learner along with the dataloader created earlier, the base model to start from and a loss function.

5.6.3 Pretrained Model Training

Now, we import the pretrained GPT-2 model, as well as the tokenizer. Also, like we mentionned earlier, GPT-2 is Huge. It is likely that if we try to use it on ou computer, we will be getting a bunch of CUDA Out of Memory errors. In such cases, an alternative that can be used is to accumulate the gradients function. The idea is simply that before calling for optimization to perform a step of gradient descent, it will sum the gradients of several operations. Then, it will divide that total by the number of accumulated steps, in order to

get an average loss over the training sample. That means much fewer calculations.

Now, we are ready to create our Learner, which is an object for grouping data as training and validation, model and loss function and handles model training and inference. Since we are in a language model setting, we pass accuracy and perplexity as metrics, and we need to use the callback we just defined. Lastly, we use mixed precision to save every bit of memory we can since we have a modern GPU, it will also make training faster. And now, finally, we create the training function that will use all our data to fine-tune GPT-2 so that it can predict quality Kafi-Nono text in the future.

Now for the this part we gradually unfreeze the aforementioned groups one by one, and each time fit a single cycle , one run of the entire dataloader. After every 500 steps, we saved the checkpoint so that we don't loose our work if something happens in the middle of the training. We did really much hyperparameter tuning on the learning rate, thanks to google for its free GPUs. In the end, we save the model for further experiment. Figure 5.7 Optimizing transformer neural networks is a computationally intensive problem

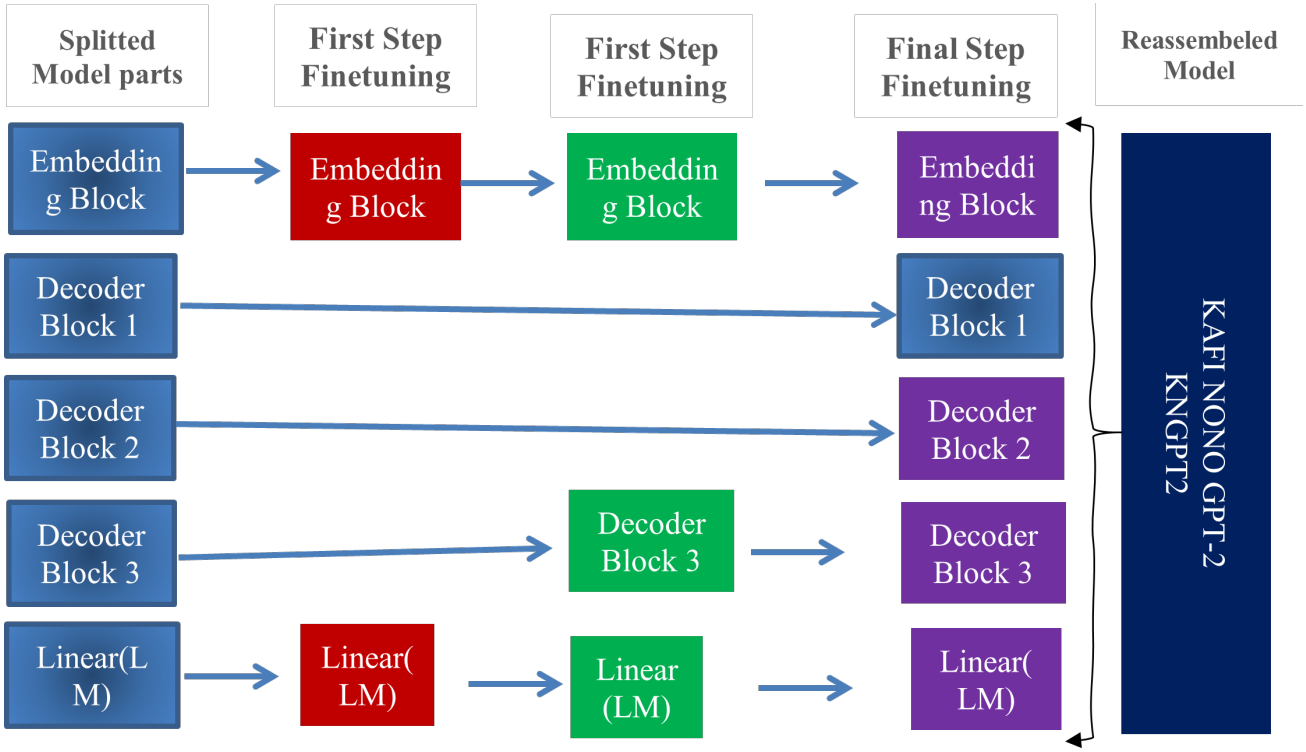


Figure 5.7: *updating Embedding layer with new dataset*

which requires the engagement of high-performance computing (HPC) clusters in order to improve time to solution.

Selection of well-performing hyperparameters requires searching a high-dimensional space. To evaluate a neural architecture or a set of hyperparameters entails running full model training and inference.

Hyperparameter	Explanation	Best value
d_model	Hidden units per layer	768
N_ctx (block size)	Code context length	768
d_x	Embedding dimension	768
N_head	Attention heads	12
Dropout	Dropout keep probability	0.9
Batch-size		128
Epoch		3
Learning rate		6.25x10-5
Optimization Algoritm		Adams
Loss function		Cross-entropy
top-k		60
Temprature		0.9

Table 5.2: *Well-performing values of model architecture hyperparameters*

We scale up the training using synchronous data-parallel distributed training algorithm with local gradient accumulation. The learning rate controlling the magnitude of the weight update during gradient optimization is lowered upon completion of each epoch according to the cosine decay. In a distributed regime, we increase the learning rate during the first few epochs (“warm-up” period) to facilitate reliable model convergence. The online training is implemented as a Python library integrating PyTorch and HuggingFace with Adams algorithm for gradient summation.

The GPU available on Colab is NVIDIA Tesla T4 equipped with roughly 15GB RAM memory. The memory size is sufficient for finetuning all layers of the small model (124M)but it is not sufficient for the medium (345M) and large model (747M). It is familiar that TPU (Tensor Processing Unit) which is more powerful than GPU, is also accessible on Colab.

5.7 Sequence Decoding

Each inference call to the model yields a probability distribution vector over subtokens in the vocabulary. This can be conceptualized as an N_ary tree of subtokens rooted in the last subtoken of the word context typed in by a user. The depth of the tree is defined as a length of the desired completion sequence. Each word sequence suggestion is

effectively a path on the tree, from the root node to a terminal node. The beam search algorithm is employed to explore and rank those paths, improving recommendation relevance of word sequences. At every step, the results are aggregated and the top k results are selected, where k is the beam width. Decoding continues for a preset number of subtokens or until a break token is reached. The set of break tokens includes the <EOS> (end-of-sentence) token as well as other language-specific tokens that often precede end-of-line under common word style patterns.

A naive beam search implementation would iterate over the top k candidates at every step to produce the output vector. However, for a sequence of length L this would require L $\times$ k inference calls to the model, significantly increasing the inference time and degrading the overall real-time user experience. Instead, we aggregate top k candidates and perform batched inference calls at every decoding step, which reduces the number of inference calls to L provides a comparison of the inference speeds for scenarios with different beam widths and sequence lengths, quoting speed-ups gained through parallelization.

Given sequential nature of the beam search decoding, we cache the attention keys and values of the transformer blocks as computed by the model for previous step (token), passing it as input to the model at the current step instead of recalculating from scratch. This further speeds up inference by 10%. The speed improvement with parallel and cached search is most apparent for large L.

5.7.1 Suggestion Processing

As mentioned in section 3, we introduce some new tokens that are not present literally in the input data. As we decode the output sequences, the model would generate those new tokens at the appropriate locations. In order to incorporate those tokens fluently into the completion sequences, we need to post-process them on the user side into printable characters using the following rules: (1) <SOS> and <EOS> tokens are ignored, as they almost never seen in the suggestion sequence and do not provide any additional information relevant to the completion sequence. (2) <EOS> serves as a break token for beam search decoder.We truncate the com at this token as it indicates the end of the line.

5.8 Model Comparison and Selection

Next we will compare the performance of the two selected models and select the better to accomplish our objective in this thesis.

While trained with the LSTM model because of our Kafi-Nono data was small and the dimension of the dictionary was only 61 words, an accuracy of 14.55% was achieved , which is bad result for the start of the research. With increasing training data, greater accuracy can be achieved.In the next, for the LSTM the prediction on validated text is given. Variants of one and several word prediction were tested. As it can be seen (Table 5.3) all the predicted words are not correct and none sense.

Word	**Prediction**
Asho (1 word)	baldiilll
No showee (2 words)	Hallo builsssn
Amo biich hamo (3 words)	qaasism bifst

Table 5.3: *The LSTM predicted results.*

From the current demonstration in table 5.3, it is clear that the prediction of the next word for Kafi-Nono is incorrect. The outputs of the LSTM model were random and lacked a coherent Kafi-Nono word structure. Additionally, given our focus on mimicking

Model	**Training Time**	**Execution Time**
LSTM	12 hours	15 seconds
Pretrained	3 hours	11 seconds

Table 5.4: *Comparison by time.*

Model	**Accuracy**
LSTM	0.1455 (14.55%)
Fine-tuned Model	0.89 (89%)

Table 5.5: *Comparing Evaluation on the base model(LSTM) and Pretrained model*

the content and style of specific languages, we chose a transformer-based model to allow our model to “focus” on learning our target languages’s structure given a specific context rather than simply learning how to complete a sentence as was the case with LSTM. Additionally, a disadvantage of recurrent neural networks is that the less recent context is eventually “forgotten” by the model when generating the next word. With transformer models, a sentence’s topic is never forgotten, allowing for robust word generation. Thus, we decided to move forward with the pretrained model for using transfer learning.

Initial Word(phrase)	Output generated by	Prefix with Generated Next Words
"ebich ittoshi mullooch",	LSTM	ebich ittoshi mullooch gicccchhg
	KNGPT2	ebich ittoshi mullooch giyaachemmina'one iye
"gaawe asho",	LSTM	gaawe asho omboool,..
	KNGPT2	gaawe asho koniyolla bi tuna gaata, ta gabiti xaa
"no showee beeti hinnoo",	LSTM	no showee beeti mkkpjihhiu. kl
	KNGPT2	no showee beeti hinnoo chuuqqeebe! iye
"shemmee gaacoon",	LSTM	shemmee gaacoon utyt;";'.,iuhu
	KNGPT2	shemmee gaacoon digenoona gaacho hakkiyo) qelli maac
"jimma university",	LSTM	jimma universiti oiuyfyvbbn ,../.m,
	KNGPT2	jimma universiti mooyo biich beeti emiroo ceennitone|
"kafi showoochee",	LSTM	kafi showoochee oiui,.;;'; kmlkju,./
	KNGPT2	kafi showoochee qitooch hammiye
"doyee shimbo",	LSTM	doyee shimbo ujjjjimm kllj
	KNGPT2	doyee shimbo kooreeti shimboon ciinnimmi kooroo
...		

Table 5.6: *Sample Output comparison from both models*

5.9 Experiments

During our experiments, we aimed to investigate four goals using the pretrained model. The first goal is to understand how fast the transformer model, adapts to Kafi-Nono language. We did this goal by generating and observing the generated text in step by step course of early fine-tuning process. Our second goal is to observe the loss values converging trend by recording the loss values during fine-tuning. Our third goal is to analyze the overall quality of the generated words/ sentences by unconditional random sampling. We used a random sampling approach in pursuit of higher text quality. Our final goal is to compare the generated words based on different text inputs for conditional random sampling. We further saved the fine-tuned model (KNGPT2) for future researchers to query the our fine-tuned model and review its Kafi-Nono text generation.

5.9.1 How Fast Transformer Model Adapts to KN Lang. Pattern

We measured how fast the GPT-2 adapts to KN language by observing the occurrences of our special tags ("<|sos|>" and "<|eos|>") in generated text. It is a reasonable expectation that GPT-2 can generate more KN like text if the number of fine-tuning step increases. Fig. 5.8

```
====== GENERATION 0 ======
kafi-noonooch gaacciyeete<|eos|>. <|sos|> hini shawee toommooch too'iyoona wotta bi kishoona gaawe
gaacciyeete<|eos|>. <|sos|> hini shawee toommooch too'iyoona wotta bi kishoona gaawe gaacciyeete<|eos|>. <|
sos|> hini sh

====== GENERATION 1 ======
kafi-noonooch, biishoonaa biishoona biich biisho bi tunatoyich, biishoona biisho bi tunatoyich, biishoona
biisho bi tunatoyich, biishoona biisho bi tunatoyich, biishoona biisho bi tunatoyich, biishoona biisho bi
tunatoyich, biishoona biisho bi tunatoyich, bi

====== GENERATION 2 ======
kafi-noonooch, qaaroona qaaroona giidoona aabichiqqi bi giidoona qaaroona woyee bekkiiheete<|eos|>. <|sos|> ebi
yee gub, aroochee yeerichi qaaroona giidoona qaaroona woyee bekkiiheete<|eos|>. <|s

====== GENERATION 3 ======
kafi-noonoochaa bi tunatoyich, yeeri bi qaaroon biich arii beeto<|eos|>. <|sos|> tunaballi yeeri biin biin
qaaroon biich arii beeto<|eos|>. <|sos|> yeeri biin qaaroon biich arii beeto<|eos|>. <|sos|> biin qaaroon bi

====== GENERATION 4 ======
kafi-noonoochaa ittoshi toommooch beeti shawee toommooch bi shuunati kashoona ittoshin shuunoonaa ittoshin
qolloona ittoshi shuunoonon ittoshi kishoona ittoshi kishoonon ittoshine<|eos|>. <|sos|> ebich ittoshi aafooch
ittoshine; aafooch
```

Figure 5.8: *The positions of special tags "<|sos|" and "<|eos|>" in the generated text*

It is to our surprise how few steps are required to generate the first KN-like text. As early as at the 200th step, generated the following KN text in figure 5.9 If there were more dataset, it would be still astonishing to see the effectiveness of transfer learning at every stage and, the effectiveness would be unreasonable. We leave this doubt to be cleared in the future by using a large corpus. Even a simple neural network with a single layer could be difficult to comprehend [9]. interpreting black box models has been a topic of research in Deep Learning for a long time. When Compared with other neural network models, there is a chance that the attention mechanism in Transformer models may provide better interpretability. For example, Vig [23] presented an open-source tool for visualizing multi-head self-attention in Transformer-based language models. The tool is capable of visualizing attention at three levels of courses: attention head level,model level and neuron level. It is noted that not all of the generated texsts make good sense. The sampling is a simple and intuitive way to understand the learning is in progress inside the transformer.

5.9.2 Training Loss In Fine-Tuning

In this trial we used most commonly used hyper parameters only changing the learning rate from 10-4 to 6.25×105. And we expected a lower training loss. The convergence of the training losses is shown in Fig. 5.11. In general, a lower learning rate leads to a slower convergence.Based on the graph, it is reasonable to us that the training loss is likely to decrease after more training steps more and more. At which step it will become

```
[190 | 309.07] loss=2.31 avg=2.63
[200 | 325.42] loss=2.35 avg=2.61
======== SAMPLE 1 ========
eeoo, bireena'on miixataano tuniyoo, woddee miixataano tuniyoo, tate ami asheena'o tunemmoyich immi
ibaroona xalloo miixoo am shiijit imoyich dojjoo amoyich, miixoo amoommo amoyich hakkiimm miixatonaa
amona mihoona miixatonaa amoneebeet miixatona amoommo amoyich hakkimm miixataani miixo amoyich
hakkiimm miixet miixemmona miixo amoyich hakkiimm miixo amoyich hakkiimm miixemm miixemmo miixamoone
miixemmona miixet miixemmo miixemmo miixemmona miixemmo miixemmo miixemmo miixemmo miixemmo miixemm
miixemm miixemmona miixemmona miixemmo miixemm miixemm miixemmo miixemmona miixemmo miixemm
miixemmona miixemmo miixemmona miixemm miixemmo miixemm miixemm miixemmo miixemm miixemm miixemmo
miixemm miixemm miixemmo miixemmona miixemmo miixemmona miixemmona miixemm mihoona miixemmone
miixemm miixemmone miixemm miixemm miixemmo miixemm miixemm miixemmona miixemmo miixemm miixemmo
miixemm nihoo miixemmo miixemmona mihoona mooyone miixemmona miixemmo miixemm miixemmona
mihoonamoyich dojjoo maac miiibeenaa mihoon mihoon miihoon mihoommon miixemmo miixemm biroon
miixoommona mihoon miihoon mihoon mihoon doyiibeeto miixemmona miixemmona mihoommone dojjoo mihoon
muumiyoona mihoon amoommone miixemm giddet miixemmo dojjoo mihoon mihoommona mihoommona mihoon
mihoon shuuneet miixemm miixemmi miixemmo mihoon mihoon doyee bana dojjoo dojjiibee mihoon kicee
mihooommone dojjjibee mihoon woddeemmo dojjjibee mihoon kiceemmo dojjjihoo shemmee mihoon kiceemmo
dojjjoo shemmee mihoommone dojjjeemmo mihoon kiceemmona mihoon mihoon mihoommoon mihoon mihoommona
mihoommona mihoon mihoon mihoommona mihoon mihoon mihoommon mihoommo mihoon mihoon mihoommo mihoon
mihoommo mihoo mihoo uumiyoo boono iiroommon qoppeemm yawoona oogee minnakannoon shiichi dojjoo
mihoo mihoon kiceemmona mihoon mihoon mihoon mihoon mihoommona mihoommo mihoo mihoon mihoommon mihoo
mihoon mihoommon mihoommo mihoommo mihoon mihoon mihoommo mihoommon mihoommo mihoon mihoommo
mihoommo mihoon mihoommo mihoon mihoommo mihoommo mihoommo mihoommo mihoommo mihoommo mihoommo
mihoon mihoommon mihoommo mihoommo mihoommo mihoon mihoommo mihoommo mihoommo mihoommo oogee
minnakannoon shichii bekkiiheete maa beetee mihoommo mihoo mihoommo mihoon mihoommo mihoon mih

[210 | 367.46] loss=2.57 avg=2.61
[220 | 383.95] loss=2.66 avg=2.61
```

Figure 5.9: *Sample automatically generated at 200th step- We observe that there are repeated words showing the model's early leaning stage.*

```
[400 | 682.20] loss=2.00 avg=2.31
======== SAMPLE 2 ========
 wa'oon danataani qaaroon boonoshi gaachoona boonoshina tookka ikka woyee dambe mihe maacoochena aal
lachi bekkach mihe maacooche miho hakkaani kechechi noonoonaa noonoon oogooch qaare gaboo hakkaani
mihe maacoochee moorfe ikkoo dojjecho  baareena'on gabaaneena'on shichoon shichee noonoonoo mihe maa
cooch beet eena'on hicheena'on barii biishoona biishoona boono shacoch bekkiibe! biishoon mihe
maacoochena mihe maacooch beet ara mihe maacooch bekkiibee boono shacocha xiishoo gabii gaboon
hakkiyaa boono shaci hinnoon shacoch mihe maacooch beet ashi bushoo waan baachiye eb moorfe doyee
moorfeena'o aalle mihe maacooch hinnoona beeto taane bi qellich boono shacooch beet qaaroon mihe
maacooch giddeebeeto tateena'on gaachoona xu'iyeete  doyechina'o tuno hakkimm kiineena'o beet boono
shacceemm malloon halliibee mihe maacooch beet mihe maacoochaa eb hinno halliibee oogooch bi qellich
mihe maacooch beet giddiyee hakkoona boono qelloona boono mihe moorfeena'on giddiyoo hakkimmina'onaa
boono halligoona boono hakkeemmon tuniyo doyechina'o tuno hakkimm mihe maacoochena mihe maacoochee
moorfeena'on halliibee moorfeena'on mihe maaceexi qaaroon biisho bi tunemmona mihe maacooch
giddiyooch qaare waan daneheete eb mihe aamechina'o mihe maacoochena biisho biishoon doyiibeet tuus
kimooch moorfeena'on gaachoona moorfeena'o mihe maaceexi ucheena'one eb hinnoona doyee mihe maacooch
qannit mihe maacooch waanee biisho biisho  wochi doyechina'o ikkochaa maaco micciyeete eb mihe
maacooch woddoo mihe maacooch beet moorfeena'on shichee gaboo mihe qeppeena'on halliibee doyechina'o
mihe maacoochena biishoona boono qellich giddiyeehan ceenniyaa mihe maacooch beet boono shacceemmi
kiceeteete eb doyechina'o shichee toommo biich qeppeena'on gaacho biich qellich mihe maacooch
giddiyoo taane, biich qeppeena'on giddiyee, qeppeena'on, biich qeppeena'on, biich qeppeena'ona boono
halliyeemm shabaaggibeet  biriyooch, mihe iiqqoon qeppeena'on aaf dojjechina'onaa kaarahe toommo
biich qeppo biich qeppeena'ona toommo biishoon dojjoona mihe aame doyee bi qelloona biishe
gaacciibeet qeppeena'on bi shachemmon halliibee birii giddiyee hinnoona mihe maacoochee moorfeena'on
biisho moorfeena'o biriyee daqqoone eb mihe maacooch qeppeena'on dojjee daqqoone eb bekkiitaa eb qep
peena'on boonoshi imo qaawiihe hinneena'one eb moorfeena'och sheqqeet wochoon imo toommo biishoo imm
one eb moorfeena'ochi qeppeena'on dojjechina'o mihe maacooch b

[410 | 720.72] loss=1.97 avg=2.30
```

Figure 5.10: *Sample text automatically generated at 400th step-We observe that the model already removed repetition which observed in the previous 200th step and is putting the words in context well improvement is clearly shown*

flat is unknown, though. The use of a different dataset to validate and prevent over-fitting is also unknown. We leave this kind of topics to the future after having more computing resources and data.

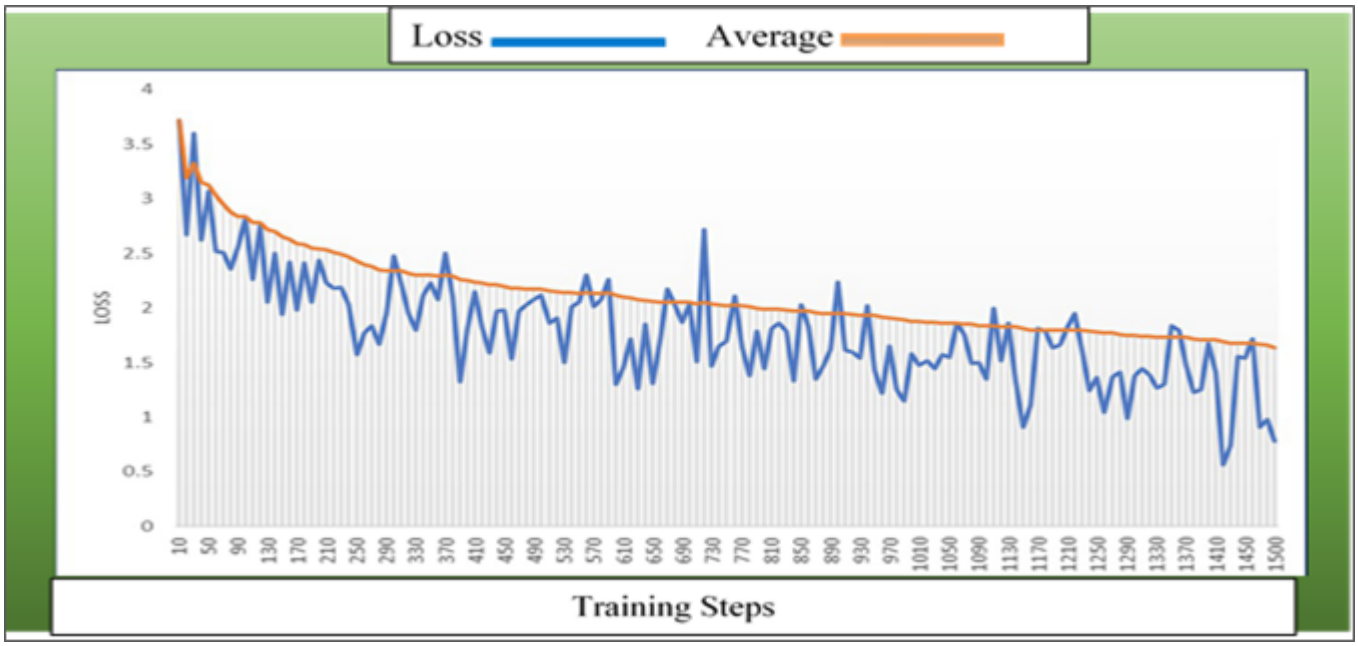


Figure 5.11: *Training loss*

5.10 Evaluation

We use two evaluation mechanisms named, an automatic metric called perplexity, and Human evaluation to evaluate the quality of text generated by our model.

5.10.1 Perplexity

Perplexity (PPL) is one of the most common metrics for evaluating language models. Before diving in, we should note that the metric applies specifically to language models sometimes called auto-regressive or causal language models Perplexity is defined as the exponential average log-likelihood of a sequence. If we have a tokenized sequence X=(x0,x1,. . . ,xt), then the perplexity of X is, The perplexity values were calculated for each of the generated samples, as equation 5.6

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, .., w_N)}} \tag{5.6}$$

The perplexity was computed for each of the produced samples, as well as for the test set of actual dataset using once again a Google Colab notebook. First, the pertained model and tokenizer were loaded. Then, the input data was tokenized in blocks of length 768 using this tokenizer. The loss of each block was calculated using the model for each input text file, and their average was taken to obtain the perplexity value of that file. The loss corresponds to the cross-entropy of the input, so the perplexity of a block was obtained by taking two to the power of the loss value. To evaluate our model using the perplexity, we have divided the dataset in to training,and evaluation parts each with a dataset 80% and 20% respectively. We have evaluated our fine-tuned model-KNGPT2 against the evaluation and testing dataset. The evaluation was carried on during training and after each 30% of the training elapse, an evaluation is computed. This is very important to examine how much the perplexity is improved while advancing the training. The lower the perplexity is the better the accuracy. fig 5.12 The perplexity in fig 5.12 shows that it decreased

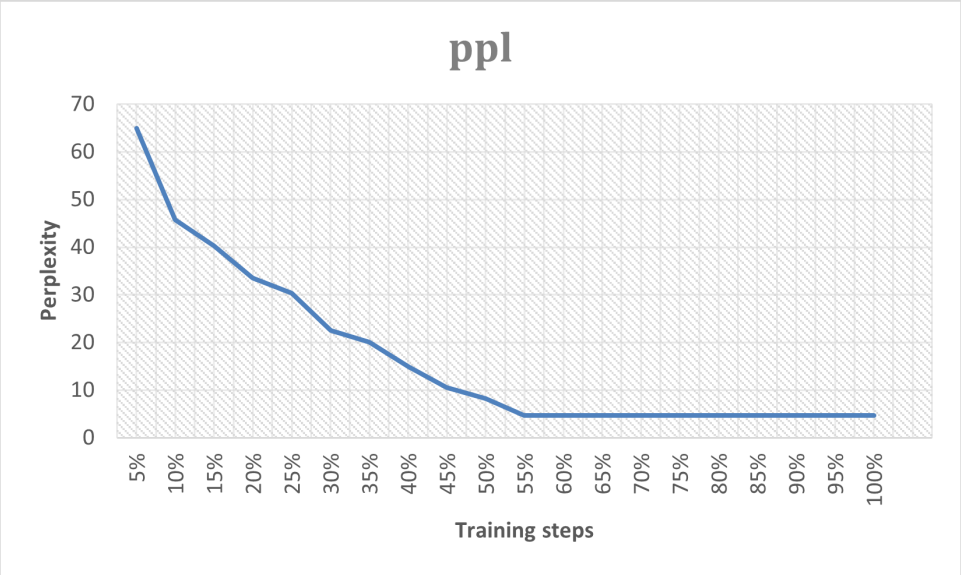


Figure 5.12: *Perplexity*

continuously until the middle of the training step at 55% and stopped decreasing. The value where it was stuck was 4.71. This shows that the model quickly understood the Kafi-Nono language and reached its lower point of perplexity around half of the training time.

5.10.2 Human Evaluation

Due to the lack of parallel data in transfer area, automatic metrics are insufficient to evaluate the quality of the transferred sentence. Therefore, we also conduct human language expert evaluation experiments. We randomly select 6 prefix sentences (4 sen-

Initial Word(phrase)	Output generated by	Prefix with Generated Next Words
"ebich ittoshi mullooch",	LSTM	ebich ittoshi mullooch giccccchhg
	KNGPT2	ebich ittoshi mullooch giyaachemmina'one iye
"gaawe asho",	LSTM	gaawe asho omboool,..
	KNGPT2	gaawe asho koniyolla bi tuna gaata, ta gabiti xaa
"no showee beeti hinnoo",	LSTM	no showee beeti mkkpjihhiu. kl
	KNGPT2	no showee beeti hinnoo chuuqqeebe! iye
"shemmee gaacoon",	LSTM	shemmee gaacoon utyt;";'.,iuhu
	KNGPT2	shemmee gaacoon digenoona gaacho hakkiyo) qelli maac
"jimma university",	LSTM	jimma universiti oiuyfyvbbn ,../.m,
	KNGPT2	jimma universiti mooyo biich beeti emiroo ceennitone|
"kafi showoochee",	LSTM	kafi showoochee oiui,.;;'; kmlkju,./
	KNGPT2	kafi showoochee qitooch hammiye
"doyee shimbo",	LSTM	doyee shimbo uijjjimm kllj
	KNGPT2	doyee shimbo kooreeti shimboon ciinnimmi kooroo
...		

Table 5.7: *Sentence Samples, generated by LSTM and KNGPT2 conditioned on prefixes for human evaluation*

tences generated for each) from each test set for human evaluation. For each review, one prefix input and four generated samples are shown to a reviewer. 10 reviewers are asked to choose the best sentence for fluency. Which sentence is the most fluent one. When we consider human linguistic experts' annotation on fluency of the generated samples, experts are requested to evaluate the fluency of each individual sample on a scale of 0-5, with 1 being "not fluent at all" and 5 being "very fluent," as done in [113] We obtain annotations from external professional linguistic experts. For fluency, we use average of the ten annotations. The method of generation is completely hidden and the order of samples in testing is randomized. Human evaluations show the average of scores (0 to 5) and the ratio of words/phrases evaluated between 4 and 5 . All results for human evaluation are on 51 randomly selected words/phrases. Our approach produces a large number of high-quality words/phrases with 89% accuracy.

5.10.3 Evaluation Results

Our model KNGPT2, is built on top of the GPT-2 – a multi-layer generative pretrained transformer model for language modeling, which is a variant of the GPT trained from scratch on huge English corpus data. Our final model yields a perplexity of 4.71 and a human evaluation of 89% for Kafi-Nono language.

5.11 The Prototype

In order to, evaluate the model and make the necessary experiment on developed word sequence prediction models a prototype developed. The prototype has been developed using Python programming language in the colab notebook. Figure 5.13 shows a user interface of the prototype.

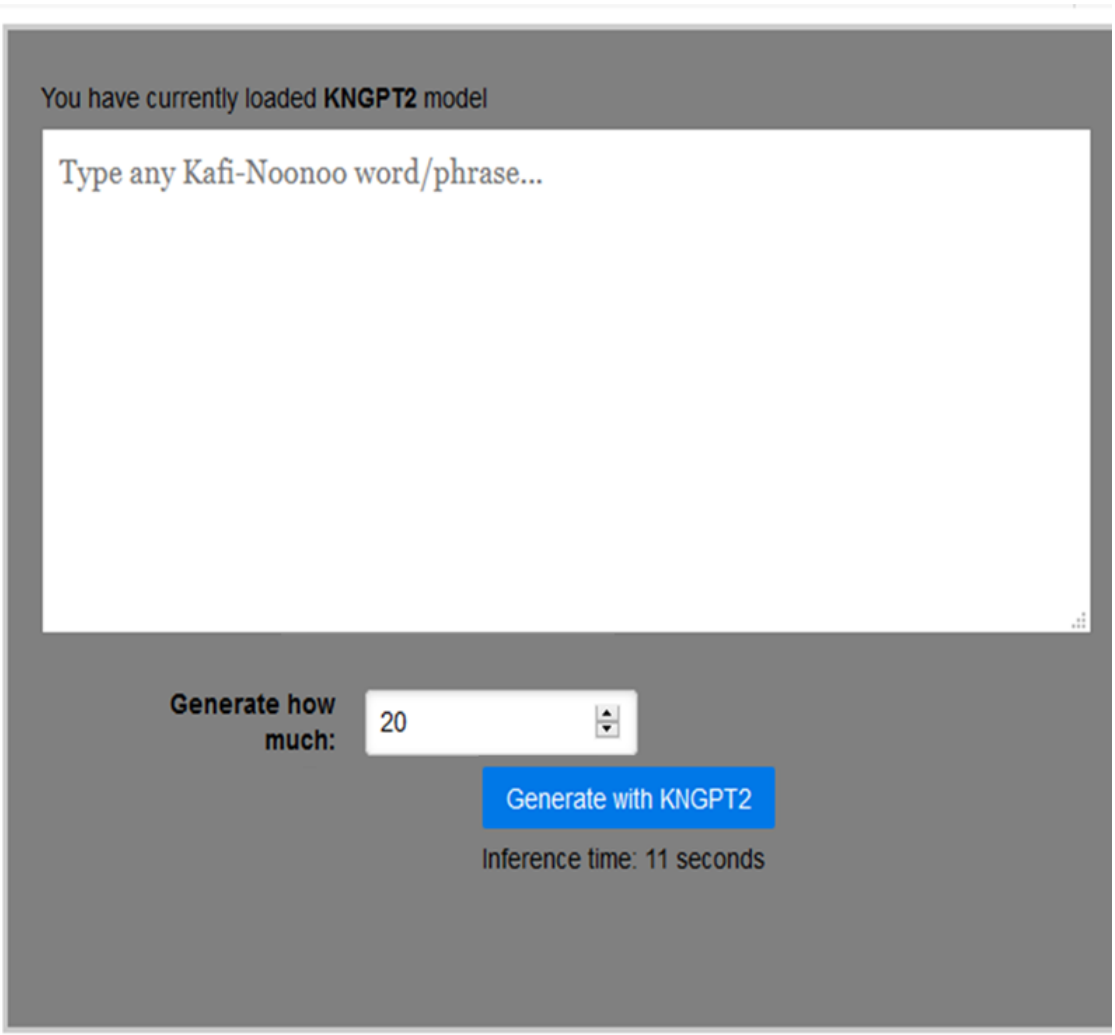


Figure 5.13: *User Interface for the system*

The prediction engine starts prediction task after users type a word or phrase in the text area and presses the generate button. Then the engine proposes the most probable k words for the number of words entered and displays in a text box next to the given word/s.

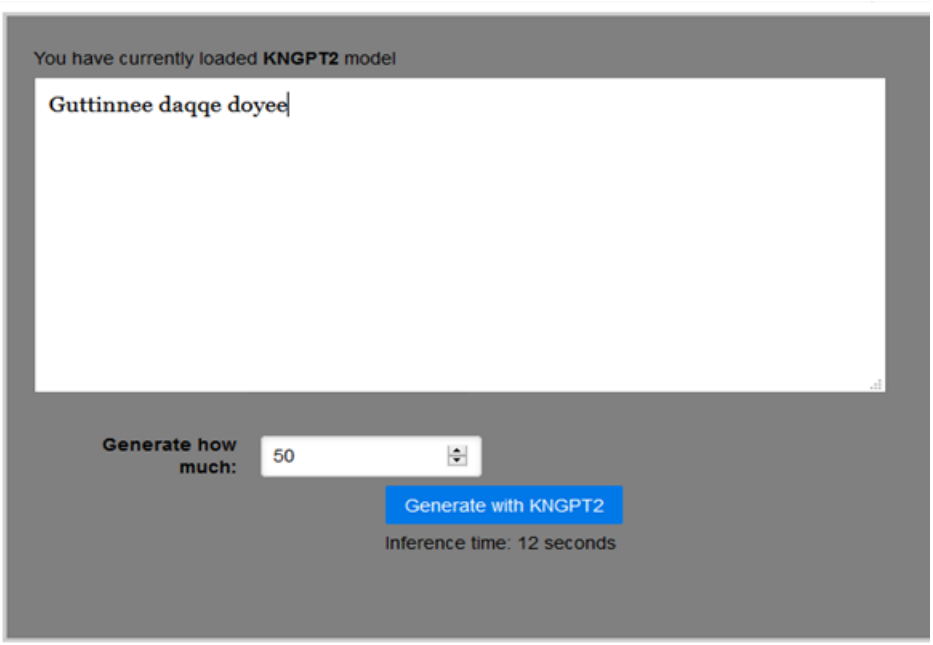


Figure 5.14: *Initial word Entered*

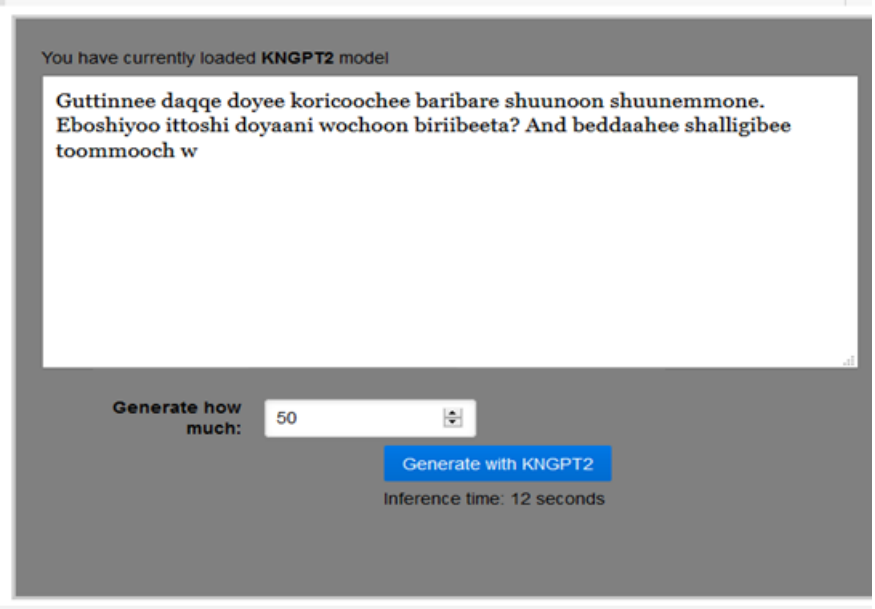


Figure 5.15: *Generated Kafi-Nono words*

5.12 Discussion

The purpose of this research work is to contribute for the development of next word perdition language model system for Kafi-Nono language. Language modeling is one of

the building block in natural language understanding because the model serves as a basement stepping stone for all other NLP tasks like NLI, Summarization, pos tagging, question answering and dialogue generation etc.

For training of the model we used the corpus collected from different sources. For the sake of this study only we collected a new Kafi-Nono data from different sources: the new testament holy bible, educational and cultural books and social media. The data is processed off as discussed in section 4.2. Specifically, hence the language was examined as under resources as discussed in section 4.2. Many researchers have also tried to use statistical methods such as N-gram to develop word prediction system for other local languages, but the most powerful techniques of the neural network models were not approached before for word prediction in local languages.This may be because of the data requirement of NNLMs. However, in this study, we have used a data efficient technique, transfer learning to leverage the shortage of data and incorporation of powerful NNLM to our local language, Kafi-Nono. As the experiment shows that in most cases, the fine-tuned pretrained model, with seem to work the best word prediction for Kafi-Nono.

From The result, we can see that Transfer learning saved our effort, time and resources very much. We are able to implement our work without the need for building a model from scratch, use only small amount of data, and very few training time. All this were the advantages of using transfer learning techniques without affecting the effectiveness of language models.

The qualitative analysis in the result show that the pretrained models learn the pattern of Kafi-Nono language faster than any other models ever. The transfer leaning approach is the most recent and very effective for low resource languages

In the result, our observation is that the quality of generated text depends on the distribution of reasonable words in actual cases. If there are many reasonable words, the number k might be too low. If there are only a few reasonable words to sample from, the number k might be too high. From the samples, we observed that the possible next words of each of the given prefixes were generated respectively with acceptable quality except the," Jimma University" which is completely non Kafi-Nono noun phrase. In such cases, we can see here that the noun system of Kafi-Nono mostly ended with -o or -oo is totally different from English. It can be seen that the length of the initial input text is short. It can also be noted that the quality of text generation depends on the input text. For example, when the input text is longer generally, it is harder for all of the generated text to be relevant to the input text. When the input text is shorter and looks like a Kafi-Nono beginning sentence, the quality of generated text is usually better. We leave experimental study on quantitative analysis for future and provide the below examples to show some reasonable quality of text generation:

\textbf{Initial Word(phrase)}	Generated Next Words
"ebich ittoshi mullooch",	ebich ittoshi mullooch giyaache
	ebich ittoshi mullooch giyaachemmina'one iye
	ebich ittoshi mullooch giyaachemmina'one
	ebich ittoshi mullooch giyaacha? mullee too'iyoon qajiichii meechi
"gaawe asho",	gaawe asho gumboon yechiye
	gaawe asho koniyolla bi tuna gaata, ta gabiti xaa
	gaawe asho taate biishoona ta bekkii beetoyich,
	gaawe asho qaawiihe
"no showee beeti hinnoo",	no showee beeti hinnoochaa no ciinnemmona yeerichi aafoo
	no showee beeti hinnoo chuuqqeebe! iye
	no showee beeti hinnoochaa wutte cinnachi shabaattooch daqq
	no showee beeti hinnoochaa wutte cinnachi shabaattooch daqq
	no showee beeti hinnoochaa showee shaahooch guupheebeeti ka
"shemmee gaacoon",	shemmee gaacoon digenoona, boono gaacoona degee imo
	shemmee gaacoon digenoona gaacho hakkiyo) qelli maac
	shemmee gaacoon digenooch hakkiimm hinneena'on k
	shemmee gaacoon digenooyich iritiyeemmi mooyo beega
"jimma universiti",	jimma universiti kashee qaaro hini yawoon gaacho ta sh
	jimma universiti mooyo biich beeti emiroo ceennitone l
	jimma universiti asheena'oochee qanaatoona uuche she
	jimma universiti asheena'ochi mooyon kaataa sha'
"kafi showoochee",	kafi showoochee daneheete aachoonaa birawoona, ebosh
	kafi showoochee qitooch hammiye
	kafi showoochee kechiti mooyon danii kichiqqi qolla
	kafi showoochee wotta maariyami, shiichiqqi shiij
"doyee shimbo",	doyee shimbo kooreeti shimboonaa kooroona yesheti
	doyee shimbo kooreeti shimboon ciinnimmi kooroo
	doyee shimbo kaacheti biroon tachii kotiichii sh
	doyee shimboochee tiijjit shalligoon kobere yawo t

Table 5.8: *Output examples from four systems of the Pretrained model*

5.12.1 Tones

A fundamental challenge arises because of Kafi-Nono's tonal system. KN has rich tonal morphology. Syntactic interactions between words impact the surface form tone a syllable takes. Thus, semantically alike words may take different surface tones than is present in the relevant lexical entry, resulting in mismatches with the lexicon. Removing tones might yield a higher hit rate of word prediction and allows tone divergences between surface forms and lexical entries to be overcome. This advantage is increased in exchange for higher polysemy (lexical ambiguity). Although this condition of polysemy is what the method of [98] is designed to address, it means the language model fails to model tones and doesn't significantly help LM-pre-training in any case. Future work have to investigate morph phonological processing for KN, since there is regularity behind these tonal changes which could mitigate these issues if addressed.

5.12.2 Kafi-Nono Dialects

KafNoonoo has three mainly known dialects, the Manjo, Chena dialect and the Gimbo dialect. If we train our model separately for each dialect, our data will be not enough since data is scarce in Kafi-Nono language. For handling the dialect, a separate dialect handling mechanism (adaptation methods [24] are normally used to incorporate dialect-specific information into the system, but this is out of the scope of this thesis.

5.12.3 Domain

Another difference that may contribute to the results is that the domain of the text collected is significantly different. The KN corpus is more of books collection. We used educational books and the bible books.

chapter Six : Conclusion, Contribution and Recommendation

In this chapter we try to address the brief summary of the research work including the main contribution and future works that could be extended from this work.

6.1 Conclusions

The study deals with Kafi-Nono word sequence prediction. As discussed in the previous chapters, in this research, transfer learning approach was used which is important for under resourced languages in general. To this end, we have forwarded the conclusion and recommendation as presented in the following sections.

The overall focus of this research is to investigate Word prediction which addresses the problem of low typing speed and poor spelling and completion. A word prediction guesses what word a user is typing, based on previously written words and prefix letters of the current word. The predictor can then either insert the word with the highest frequency into the text or let the user choose an alternative from some kind of list interface which is then completed. Ideally, this can speed up and ease the user's typing of words. The method used here for developing the word prediction is transfer learning on huge pretrained english model.

Deep learning and machine learning are progressing with leaps and bounds these days because of the availability of greater amount of computing power. Now language models are making so much progress with the help of the transformer models trained on terabytes of data. Two models LSTM and transformer model have been investigated for a particular lan- guage modeling problem as described. Both models, LSTM are an advanced variants of Recurrent Neural Networks.

Experiments were performed with the combination of different parameters in these models to check out which one is giving us better performance for our particular problem. For the quality of predictions, perplexity was used as a performance metric and softmax as an activation function. There are many factors like accuracy, training time, response time which needed to be considered before selecting the better model. If the model takes a much longer time while training it will impact other processes of the system.

It can be seen that both model results are very far to each other except for one

parameter which is training time. From the results, it can be seen that LSTM is taking much longer time than the transformer . It has been observed clearly that the transformer models are giving good results and with more larger models will become more efficient and with higher accuracy.

For under resourced Ethiopian language like Kafinono transfer learning which is unsupervised approach is recommended. Since there is no annotated corpus (even difficult to obtain electronic materials for such language), unsupervised approach plays a great role to predict simply without considering its contexts. The unsupervised approach which is not relies on hand- constructed rules that are acquired from language specialists rather than automatically trained from data.

In the beginning of the work, there we discussed existing solutions for word prediction task all of which the focus on n-gram language models. Consequently, theoretical background needed for the word prediction was introduced. That included methods for frequency estimates, in order to cope with sparse training data of the language models and approaches to combine more models together

In this research work, we proposed a novel NLP Language model called KNGPT2 for Kafi-Noonoo next word prediction that aims to model the language behavior and generate the next word/phrase or even sentence of the language. It is shown that the approach for solving this problem has been transfer learning followed by transformer model fine-tuning. In this work, we have built a next word prediction system for Kafi-Noonoo language. We have used a neural approach. The work presents a deep learning approach to next word prediction in Kafi-Noonoo language relying on the given data. In this work we developed a generic model for KN next word prediction system using deferent sub components, which contains major components such as preprocessing, tokenization and word encoding and transformer network, GPT-2. Each component is comprised of their own subcomponents and algorithms. We used python Programming language in the google colab platform as a developmental enviroment and other publicly available Libraries. When evaluating the system, special emphasis is placed on accuracy via automatic(perplexity) measures and human expert evaluations on the Training and testing dataset. We found out that our model best worked on segmented texts with a final evaluation perplexity score of 4.7 and a human evaluation of 89% accuracy. In this work the unsupervised machine learning technique: transfer learning achieved an accuracy of 89%, 14.55% on pretrained and LSTM models. We found that better results were achieved using unsupervised machine transfer learning features from the other type of machine learning like LSTM commonly used for word prediction. The results obtained were encouraging as there is lack of resource of the language because of shortage of corpus.

As we study the technical procedures to solve the main problem of the Kafi-Nono

language Word prediction the Appropriate technical application or implementations tools for the problem, For the implementation tools its better if using the python based machine learning frame works such as pytorch and libraries such as huggingface and fastai; and from machine learning transfer learning is best for the Under resource language like Kafi-Nono Transfer learning Word Prediction methods are based on prerained models and small corpora. They do not rely on labeled training text . Transfer learning used on small training data and suitable for under resourced languages and also this approach depends on the available and more reliable computing environment.

Previously nothing is done on Kafi-Nono language Word prediction and Auto completion so One of the existing problems with Kafi-Nono language is lack of word prediction and auto word completion services. There is no application that helps user of Languages as they type speedily and write correct spelling. Being not having word prediction, creates multiple problems with Kafi-Nono texts literary works such as journals, Books, fictions and some newspapers. In addition to this the speed of typing of many secretaries when they write Kafi-Nono texts is very low and misspelled word that create miscommunication.

The main purpose of developing the word prediction and Auto completion prototype for Kafi-Nono language is to overcome the problems that happens in the language user and new user of the language, The user of this language due to absence of the word prediction when they write Kafi-Nono texts their speed of typing is very low and misspelled word that create miscommunication. Therefore Auto completion of word prediction for Kafi-Nono will Increase or improve in terms of typing Speed, correcting the misspelled word and also make the language itself technology.

There is different way of measuring the performance of the system. In our case the most intuitive performance measure is the perplexity and human evaluation of word prediction.It gives an idea about how many succeeding words can be predicted correctly knowing the previous words. Our contributions include:

1. A new Kafi-Nono language dataset

2. being the first to apply transformer model on word sequence generation for local languages,

3. providing various experiment results for qualitative analysis and future research,

4. publicly availing the fine-tuned transformer model (KNGPT-2) for future researchers to further explore and

5. showing that cross-lingual language models can be implemented for local low resource languages modeling easily.

6.1.1 Future Work Recommendation

During our thesis work on Kafi-Nono language modeling, we have observed different challenges that arise both from the language characteristics and the technology access. For better results, we recommend the following key tasks.

1. Use separate dataset for each of the KN dialects- since dialects of KN affect the wording of the language it is very much better to try a separate dataset for each dialect.
2. implement bigger size pretrained models
3. Consider for handling tonal words before training either by removing or other mechanism
4. Increase the training epoch with more dataset for better results
5. Include human evaluation in addition to automatic measurement of accuracy.

Reference

[1] I. Goodfellow, Y. Bengio, and A. Courville, "6.5 back-propagation and other differentiation algorithms. deep learning," 2016.

[2] T. Mikolov, *Recurrent neural network based language model*. Eleventh annual conference of the international speech communication association, 2010.

[3] I. Polosukhin, L. Kaiser, A. N. Gomez, L. Jones, J. Uszkoreit, N. Parmar, N. Shazeer, and A. .-.-. Vaswani, "Attention is all you need," vol. 3762.

[4] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005*, 2013.

[5] F. T. Bekele, ""morphology based spell checker for kafi-noonoo language" a msc thesis submitted to department of computer science, addis ababa university, ethiopia, october, unpublished," 2018.

[6] A. De Brebisson and P. Vincent, "An exploration of softmax alternatives belonging to the spherical loss family," *arXiv preprint arXiv:1511.05042*, 2015.

[7] C. Hegde and S. . J. Patil, "Unsupervised paraphrase generation using pre-trained language models," vol. 5477, 2020.

[8] C. . J. Kaiser, "Towards data science " archived from the original on 15 february 2020 retrieved 27 february 2021," 2020.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[10] L. Booth and C. Morris, "wean ricketts, and alan newell. using a syntactic word predictor with language impaired young people. we h," in *J.* California, USA California State University, Northridge: Los Angeles, 1992.

[11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[12] M. Olazaran, "A sociological study of the official history of the perceptrons controversy," *Social Studies of Science*, vol. 26, p. 3, 1996.

[13] B. Wilson, "The machine learning dictionary," *University of New South Wales*, 2012.

[14] G. Lesher, B. Moulton, and D. Higginbotham, ""effects of n-gram order and training text size on word prediction," in proceedings of (resna99) annual conference, arlington, va," *p*, pp. 52–54, 1999.

[15] A. At, ""predicting the next search keyword using deep learning"," vol. 2019. [Online]. Available: https://towardsdatascience.com/never-leave-the-search-result-page-19b654791c27.html

[16] M. Ghayoomi and S. Momtazi", "An overview on the existing language models for prediction systems as writing assistant tools" department of computational linguistics saarland university," *Saarbrucken, Germany*, vol. 2009.

[17] A. Fazly and G. Hirst, *" Testing the efficacy of part-of-speech information in word completion"*. In Text Entry '03: Proceedings of the 2003 EACL Workshop on Language Modelling for Text Entry Methods, pages 9–16, Budapest, Hungary Association for Computational Linguistics, 2003.

[18] A. Yadav, ""artificial intelligence for low-resource communities: Influence maximization in an uncertain world"," Ph.D. dissertation, A Dissertation Presented to the faculty of the graduate school university of southern California in Partial Fulfilment of the Requirements for the Degree DOCTOR OF PHILOSOPHY (Computer Science), August 2018.

[19] S. Golovanov *et al.*, "Large-scale transfer learning for natural language generation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[20] K. Johnson, "Openai releases curtailed version of gpt-2 language model," *Venture Beat, Aug*, vol. 20, 2019.

[21] S. Enarvi, P. Smit, S. Virpioja, and M. Kurimo, "Automatic speech recognition with very large conversational finnish and estonian vocabularies," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 11, pp. 2085–2097, 2017.

[22] S. Falkner, A. Klein, and F. Hutter, ""bohb: Robust and efficient hyperparameter optimization at scale"," 2018, in: arXiv preprint.

[23] Y. Fan *et al.*, "(2016). "video-based emotion recognition using cnn-rnn and c3d hybrid networks"," in *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. pp. 445–450.

[24] K. Fukushima, ""neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position"," *In: Biological cybernetics*, vol. 36, p. 4, 1980.

[25] I. Sutskever, O. Vinyals, and Q. V. Le, ""sequence to sequence learning with neural networks",*" In: Advances in neural information processing systems, pp*, vol. 3104, 2014.

[26] D. So, Q. Le, and C. Liang, "The evolved transformer," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5877–5886.

[27] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," *arXiv preprint arXiv:1611.01462*, 2016.

[28] J. Jackson. Autocompletion with deep learning. [Online]. Available: https://tabnine.com/blog/deep

[29] X. Chen *et al.*, "(2014). "efficient gpu-based training of recurrent neural network language models using spliced sentence bunch",*" In: INTERSPEECH- pp*, vol. 641, 2014. [Online]. Available: https://www.isca-speech

[30] A. Colic, H. Kalva, and B. Furht, ""exploring nvidiacuda for video coding",*" in Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems*. MMSys '10. Phoenix, 2010.

[31] Y. You *et al.*, "(2019). "large batch optimization for deep learning: Training bert in 76 minutes",*" in International Conference on Learning Representations*, URL.

[32] R. Rosenfeld, ""two decades of statistical language modeling: Where do we go from here?" in: Proceedings of the ieee 88.8,*" p*, vol. 1270, 2000.

[33] A. Kannan *et al.*, "(2016). "smart reply: Automated response suggestion for email",*" in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp.

[34] R. Kneser and H. Ney, ""improved backing-off for mgram language modeling",*" in 1995 International Conference on Acoustics*. and Signal Processing. Vol. 1. IEEE, pp. 181: Speech, 1995.

[35] A.-L. Popkes, ""language modeling with recurrent neural networks - using transfer learning to perform radiological sentence completion". ma thesis,*" Rheinische Friedrich-Wilhelms-Univesity pp*, vol. 6645, 2018.

[36] M. Lankinen, ""modeling finnish language with character-word compositional language model",*" Master's thesis, Aalto University, 2016.*

[37] T. J. Goodman, ""a bit of progress in language modeling extended version",*" In: Machine Learning and Applied Statistics Group Microsoft Research. Technical Report, MSR-TR-2001-72. URL:, Tech. Rep., 2001.*

[38] D. Jurafsky, J. H. M. University, and x. . p. h. I. .-.-.-.-. . University of Colorado at Boulder) Pearson Prentice Hall, 2009, "Speech and language processing," *Vol.*, vol. 3, 2014.

[39] H. Adel, N. T. Vu *et al.*, "(2013). "recurrent neural network language modeling for code switching conversational speech"," I. I. C. on Acoustics, Ed. Speech and, 2013.

[40] H. Adel, K. Kirchhoff *et al.*, "(2014). "comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding"," in *Fifteenth Annual Conference of the International Speech Communication Association*.

[41] S. Hochreiter and J. Schmidhuber, ""long short-term memory"," *n: Neural computation*, vol. 9, p. 8, 1997.

[42] D. R. Beukelman *et al.*, "Frequency of word occurbence in communication samples produced by adult communication aid users," *Journal of Speech and Hearing Disorders*, vol. 49, no. 4, pp. 360–367, 1984.

[43] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," vol. 2019.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back- propagating errors," 1986.

[45] N. Garay-Vitoria and J. Abascal, "Text prediction systems: a survey," *Univers. Access Inf*, vol. 4, p. 3, February 2006.

[46] J. A. Van Dyke, "Word prediction for disabled users: Applying natural language processing to enhance communication," Ph.D. dissertation, Honors BA Thesis, University of Delaware, 1991.

[47] R. Kneser and H. Ney, "Improved backing-off for n-gram language modeling," in *International Conference on Acoustics, Speech and Signal Processing*, pp. 181–184,.

[48] Y. Ji, T. Cohn, L. Kong, C. Dyer, and J. Eisenstein, "Document context language models," in *Proceedings of the International Conference on Learning Representations*, pp. 148–154,.

[49] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *International workshop on artificial intelligence and statistics*. PMLR, 2005, pp. 246–252.

[50] A. Mnih and G. Hinton, "A scalable hierarchical distributed language model," in *Advances in Neural Information Processing Systems 21*. MIT Press.

[51] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of Interspeech*, pp. 462–466,.

[52] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of ICASSP*, pp. 253–258,.

[53] Y. Kim, Y. Jernite, D. Sontag, and A. Rush, "Character-aware neural language models," in *Thirtieth AAAI Conference on Artificial Intelligence*, pp. 381–389,.

[54] Y. Miyamoto and K. Cho, "Gated word-character recurrent language model," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, pp. 1992–1997,.

[55] T. Wang and K. Cho, "Larger-context language modelling with recurrent neural network," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, pp. 1319– 1329,.

[56] A. . F. Bajpai, "Recurrent neural networks: Deep learning for nlp," *Towards Data Science Retrieved*, vol. 19, p. 2021, January 2019.

[57] J. M. Keith Y, "The effects of word prediction on communication rate for aac„" *Department of Computer and Information Sciences University of Delaware Newark, DE 6*, vol. 1971.

[58] A. Graves and J. Schmidhuber, ""offline handwriting recognition with multidimensional recurrent neural networks"," *In: Advances in neural information processing systems, pp*, vol. 545, 2009.

[59] A. Graves and N. Jaitly, ""towards end-to-end speech recognition with recurrent neural networks"," in *International conference on machine learning*. pp. 1764–1772, 2014.

[60] C. Buck, K. Heafield, and B. van Ooyen, "N-gram counts and language models from the common crawl," *Archived from the original on*, vol. 28, p. 2021, January 2021.

[61] C. Olah, *Understanding LSTM Networks*, URL, 2015. [Online]. Available: http//colah.github.io/posts/2015-08-Understanding-LSTMs/

[62] A. Radford *et al.*, "(2019). "language models are unsupervised multitask learners"," *In: OpenAI Blog*, vol. 1, p. 8. [Online]. Available: https://cdn.openai.com/betterlanguagemodels/language_models_are_unsupervised_

[63] S. Tekle, *Kafi Noonee Doyee Indee weiqqeena'o*. Kaffa: Bonga, 2016.

[64] M.-T. Luong, H. Pham, and C. D. . A. Manning, "Effective approaches to attention-based neural machine translation," vol. 4025, 2015.

[65] C. Olah and S. Carter, "Attention and augmented recurrent neural networks," *Distill*, vol. 1, no. 9, p. e1, 2016.

[66] D. Bahdanau, K. Cho, and Y. . S. Bengio, "Neural machine translation by jointly learning to align and translate," vol. 473, 2014.

[67] Y. . J. Tsvetkov, "Opportunities and challenges in working with low-resource languages," *(PDF) Carnegie Mellon University Archived (PDF) from the original on*, vol. 31, p. 2020, March 2017.

[68] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. . J. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," vol. 6724, 2015.

[69] G. Lai, Q. Xie, L. Hanxiao, Y. Yang, and E. . A. Hovy, "Race: Large-scale reading comprehension dataset from examinations," vol. 4683, 2017.

[70] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. . A. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," vol. 7461, 2018.

[71] T. H. Trinh and Q. V. . J. Le, "A simple method for commonsense reasoning," vol. 2847, 2018.

[72] K. . F. Quach, "Roses are red, this is sublime: We fed openai's latest chat bot a classic reg headline," *The Register Archived from the original on*, vol. 9, p. 2021, March 2019.

[73] K. . M. Wiggers, "Google open-sources framework that reduces ai training costs by up to 80," 2020.

[74] K. . M. Piper, "A poetry-writing ai has just been unveiled it's pretty good," 2019.

[75] M. . D. Olson, "Ai dungeon 2, the text adventure where you can do nearly anything, is now on mobile," *Archived from the original on*, vol. 20, p. 2020, September 2019.

[76] J. . A. Nelius, "This ai-powered choose-your-own- adventure text game is super fun and makes no sense," *Gizmodo Archived from the original on*, vol. 28, p. 2021, February 2020.

[77] A. . F. Ha, "Ai dungeon-maker latitude raises $3 3m to build games with 'infinite' story possibilities," *TechCrunch Archived from the original on*, vol. 21, p. 2021, February 2021.

[78] A. Ohlheiser and K. . F. Hao, "An ai is training counselors to deal with teens in crisis," *MIT Technology Review Archived from the original on*, vol. 27, p. 2021, February 2021.

[79] T. Q. Nguyen and D. Chiang, "Transfer learning across low-resource, related languages for neural machine translation," arXiv, preprint, 2017.

[80] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global vectors for word representation", 2014.

[81] G. Lample and A. Conneau, "Cross-lingual language model pretraining," arXiv, preprint, 2019.

[82] F. Morin and Y. Bengio, ""hierarchical probabilistic neural network language model." in: Aistats," *Vol.*, vol. 5, pp. Citeseer, pp. 246–252, 2005.

[83] Y. Bengio, J.-S. Senecal *et al.*, "(2003). "quick training of probabilistic neural nets by importance sampling." in: Aistats," *p*, vol. 1.

[84] M. Gutmann and A. Hyvarinen, *"Noise-contrastive estimation: A new estimation principle for nnormalized statistical models"*. n: Proceedings of the Thirteenth International Conferenc, 2010.

[85] M. Kurimo *et al.*, "(2006). "unlimited vocabulary speech recognition for agglutinative languages"," in *Proceedings of the main conference on Human Language Technology Conference of the North America*.

[86] T. Mikolov and S. Kombrink, "(2011). "extensions of recurrent neural network language model"," in *In:IEEE international conference on acoustics*. speech and signal processing (ICASSP, 2011.

[87] L. Verwimp and J. Pelemans, "Patrickwambacq, et al. (2017). "characterword lstm language models"," in: arXiv preprint.

[88] Z. Mekuria and Y. Assabie, "A hybrid approach to the development of part -of-speech tagger for kafi noonoo text," in *Proceedings of the 15th 65 International Conference on Intellegent Text Processing and Computational Linguistics(CweCLing 2014)*. Nepal, 2014.

[89] M. L. and. Martin J. Puttkammer., "Viability of neural networks for core technologies for resource-scarce languages."

[90] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv, preprint, 2018.

[91] V. Keselj, "Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6, $115.00," 2009.

[92] J.-S. Lee and J. Hsiang, "Patent claim generation by fine-tuning openai gpt-2," *World Patent Information*, vol. 62, 2020.

[93] I. Goodfellow, Y. Bengio, and A. Courville, *6 5 Back- Propagation and Other Differentiation Algorithms*. Deep Learning pp 200–220: MIT Press, 2016.

[94] T. Tensu, "Word sequence prediction for amharic language thesis submitted to the school of graduate studies of addis. ababa university in partial fulfillment of the requirements for msc in cs., unpublished."

[95] N. Suleiman, "Word prediction for amharic online handwriting recognition."

[96] A. B. Delbeto, ""word sequence prediction for afaan oromo",. unpublished masters thesis, department of computer science, addis ababa."

[97] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," arXiv, preprint, 2018.

[98] L. Duong, H. Kanayama, T. Ma, S. Bird, and T. Cohn, "Learning crosslingual word embeddings without bilingual corpora," *arXiv preprint arXiv:1606.09403*, 2016.

[99] Ghosh, Surjya, et al. "Evaluating effectiveness of smartphone typing as an indicator of user emotion." 2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII). IEEE, 2017.

[100] Aron, Jacob. "How innovative is Apple's new voice assistant, Siri?." (2011): 24.

[101] Niu, Shuteng, et al. "A decade survey of transfer learning (2010–2020)." IEEE Transactions on Artificial Intelligence 1.2 (2020): 151-166.