Efficient Transformer Language Models

Jaedon Van Schalkwyk
Department of Computer Science
University of Cape Town, South Africa
VSCJAE001@myuct.ac.za

Abstract

The recent surge of research into language modelling has led to vast improvements in various natural language processing (NLP) tasks. In line with this is the overall improvement of the transformer architecture, and the attention mechanism which it utilises, which is now the most popular implementation of neural networks for NLP. Through new research, different variations of the attention mechanism have been proposed, most notably the block sparse implementation. This paper aims to evaluate the speed and accuracy of transformer models using sparse attention (BigBird and Longformer) when compared to the original full attention (GPT-2) on low resource South African languages. Byte-pair encoding is utilized to appropriately account for the agglutinative nature of these languages, ensuring the best overall performance regardless of the model. We hope that this paper will encourage research into language modelling for low resource African languages.

1 Introduction

Natural Language Processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret, and manipulate natural language text or speech (Chowdhary 2020). NLP applications use language models that focus on statistical and probabilistic models to determine the sequence of words occurring in a sentence based on the previous words (Chaudhuri 2022). One primary area of use for such a model is natural language generation, which aims to produce plausible text in human language from an input (Li et al. 2021). In recent years the advancement of language models and NLP has been synonymous with the drastic improvement of the transformer architecture. Transformers are a neural network architecture, first proposed by Vaswani et al. 2017, that are used mainly for processing long sequences such as text passages. These models make use of an encoder-decoder (sequence-to-sequence) structure that takes words/sentences as input and produces words/sentences as output. They are especially useful in NLP as they are able to effectively learn from large text corpora and transfer over that knowledge to practical applications.

However, transformers can be computationally expensive, limiting the length of sequences they are able to process. This is due to a key architectural feature that they employ: **attention**. Attention is a mechanism that compares the words (tokens) in a sequence to other tokens in the sequence, generating relation and context between them. In the original transformer proposed by Vaswani et al. 2017, a full attention mechanism is used, where every token attends to every other token, resulting in a *quadratic* complexity.

This paper aims to compare the original transformer, to two newly proposed modifications, namely: BigBird (Zaheer et al. 2020) and Longformer (Beltagy et al. 2020). Both of these models use different versions of a modified attention mechanism, **sparse attention**, which performs and scales with linear complexity. These models improve on the inefficiencies of the original transformer, allowing longer sequences to be processed with a greater accuracy. In addition to this, this paper also sets out to evaluate the performance of transformer models on low resource languages, such as the ones from the South African Benue-Congo language family. In South Africa, the most prolific language families are those of the Nguni (isiZulu, isiXhosa, siSwati, and isiNdebele) and the Sotho-Tswana (Sesotho, and Sesotho sa Leboa) groups. Together these groups represent almost 70% of the population (Mesthrie et al. 2002). The datasets used in this paper were chosen to represent these groups appropriately, and as such, this research was conducted on isiZulu and seSotho language sets. These languages also provided some of the largest available text collectives.

The main focus of this paper will be on evaluating each language model's text generation ability on low resource languages, measured through accuracy and speed. In this work, we make the following contributions: perform extensive experimentation on different transformer architectures using varying datasets and comparing and evaluating the results.

2 Background

Language models predict the next word in a sequence based on the preceding context. They are able to define a probability for every possible next word and produce a probabilistic distribution over an entire vocabulary (Jurafsky et al. 2022). These models assign probabilities $P(w_{1:n})$ to sequences of length n, such that $w_{1:n} = w_1, ..., w_n$. Using these conditional probabilities in combination with the chain rule yields the following equation:

$$P(w_{1:n}) = \prod_{i=1}^{n} P(w_i|w_{<i}) \quad (1)$$

This research focuses on autoregressive language models, also known as casual language models (CLMs). CLMs are feed-forward models that predict future values from past values. The sequence generation process for these models is modelled as a Markov chain, where the next token that is to be predicted depends only on previous tokens (Liao et al. 2020). The training objective is formulated as follows:

$$L_{clm}(X) = \sum_{i=1}^{n} logP(w_i|w_{<i};\theta) \quad (2)$$

where θ represents the model parameters in use.

2.1 Tokenization

Tokenization is a way of splitting a sentence into tokens. A token is an instance of a sequence of characters that are grouped together to form smaller, more useful semantic units for processing (Manning et al. 2019). Tokenization allows machine learning models to better understand words by themselves, as well as how they contribute to the overall meaning of a sentence. Tokens can either be words, characters, or subwords (n-gram characters).

Conventionally, language models tokenize input text into individual words, where words lying outside of the vocabulary set are replaced with *unknown tokens*. The problem with this approach is that the entire information of that word is lost. South African Benue-Congo[1] languages are highly agglutinative, meaning that whole-word tokenization is sub-optimal for language modelling (Mesham et al. 2021). Furthermore, character level tokenization rapidly increases the input and output sentence length and thus requires the model to learn from long sequences. As such, subword tokenization, specifically Byte Pair Encoding (Sennrich et al. 2016), will be used in this paper.

Byte Pair Encoding (BPE) is a word segmentation algorithm that keeps the most frequently used words as whole, while splitting the rare and unknown words into smaller token batches until a desired vocabulary size is reached (Provilkov et al. 2020). BPE effectively combats large vocabulary problems, and is ideal for agglutinative languages. To ensure fair and accurate evaluation across all models, BPE tokenizers were trained for each model using the training datasets. An example of what the tokenization looks like can be seen below in Figure 1.

Ngalolosuku , sathi :
ngalo ##lo ##suku , sa ##thi :

Bopang dikamano tse ntle le basebetsi ba bang
bopa ##ng dikamano tse ntle le basebetsi ba bang

Figure 1: BPE tokenization using the trained GPT2Tokenizer on isiZulu (top) and seSotho (bottom). The tokenizer was trained following the original tokenizer as per Section 2.3.1. BigBird and Longformer follow similar tokenization schemes

2.2 Evaluation

Evaluation of language models can either be done intrinsically or extrinsically. Intrinsic evaluation analyzes statistical measures of the model as a stand-alone system, while extrinsic evaluation analyzes the impact of the model on broader NLP applications (Mollá et al. 2003). The evaluation strategy chosen for this paper focuses on intrinsic evaluation metrics related to perplexity and bits per character (BPC).

Intuitively, evaluation of language models is done by analyzing how well they predict unseen text. Quantita-

[1]Language classification consisting of both Nguni-Tsonga and Sotho-Makua-Venda language families

tively they can be evaluated using **perplexity**. Perplexity is the inverse probability that the model θ assigns to the test set, normalised by the length of the test set (Jurafsky et al. 2022), defined below:

$$PP_\theta(w_{1:n}) = P_\theta(w_{1:n})^{-\frac{1}{n}} \quad (3)$$

Due to the inverse relationship of equation (3), the higher the conditional probability of the word sequence, the lower the perplexity. Thus, lower perplexities indicate increased accuracy of the model's estimate of the true distribution for the masked token.

Perplexity is used to evaluate language models with a fixed vocabulary size. As such, it is inapplicable to unnormalized language models[2], and is not comparable between language models with different vocabularies (S. F. Chen et al. 1998). Since models with different tokenization approaches and vocabularies will be used, perplexity evaluation alone is not sufficient in this paper. Therefore, this research makes use of BPC, a measure of perplexity normalized by character length. This approach is independent of both vocabulary size and tokenization. The BPC for model θ on test set $w_{1:n}$, with c characters is defined as follows

$$BPC(w_{1:n}) = \frac{1}{c} log_2 P_\theta(w_{1:n}) \quad (4)$$

2.3 Transformer Models

The transformer model, first proposed by Vaswani et al. 2017, is a neural network architecture that dispenses with recurrence and convolution architectures (Lecun et al. 1998) and focuses solely on attention mechanisms. The attention mechanism maps a Query (Q) and Key (K) vector to a weighted sum of Values (V) vectors to determine relation and context between tokens in the input sequence. The attention is mapped several times in parallel in what is known as multi-headed attention. Transformer models can make use of either an encoder (auto-encoding model) or decoder (autoregressive model) block in their architecture. This paper focuses on models that only make use of the decoder block. This block takes in a sequence $X = (x_1, ..., x_n)$ and produces a latent representation $Z = (z_1, ..., z_n)$ which is used to determine an output sequence $Y_M = (y_1, ..., y_M)$ one element at a time (Gillioz et al. 2020). Subsequently, it improves on previous neural network models as it can effectively process long-range dependencies of a sentence, while also being highly parallelizable.

However, due to the attention mechanism, transformers can be computationally expensive. This places a limit on the length of sequences that they are able to process. In the original transformer (Vaswani et al. 2017), full attention is used, where every token attends to every other token, resulting in a *quadratic* complexity. The following descriptions are given in terms of the respective model's original implementation (as a English pretrained CLM). The models used in this paper are trained on the isiZulu and seSotho datasets, but follow the architectures of the models mentioned below.

2.3.1 GPT-2

GPT-2, short for Generative Pre-trained Transformer 2, is a unidirectional transformer pretrained using language modelling on a corpus consisting of data scraped from outbound links gathered from Reddit. The resulting dataset, WebText, contained 45 million links, 8 million documents, and 40GB of text. It is a stack of decoders where the multi-headed attention module with cross-attention is removed from the decoder block (Ghojogh et al. 2020). It is a general-purpose learner, meaning that it was not trained for any specific task and thus can be applied across many NLP applications. It is a direct scale up of GPT (Radford, Narasimhan, et al. 2018), which trains by only conditioning the left context in all layers from deep unidirectional representations.

The attention mechanism of GPT-2 works as the Query (Q), Key (K), and Value (V) vectors dynamically generate weights for every token, representing the relationship and context of each token relative to each other (Luo et al. 2020). From the definition of self-attention, Q and K are the same vectors, and thus share the same dimensionality d_k. The attention is calculated individually and in parallel for each head, on a group of random parameter matrices on Query (w_i^Q), Key (w_i^K), and Value (w_i^V). As this is an autoregressive model, the attention scores are only applied to tokens on the left of the token being considered. The attention function is thus defined as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (5)$$

$$Head_i = Attention(Qw_i^Q, Kw_i^K, Vw_i^V) \quad (6)$$

[2]Models where the probability distribution of the tokens does not sum to 1

2.3.2 BigBird

BigBird is another transformer architecture that satisfies all the known theoretical properties of a full transformer, being an encoder-decoder architecture that leverages a multi-headed attention mechanism to process input sequences (Zaheer et al. 2020). It was pretrained following the same process as BERT (Devlin et al. 2018) using four publicly available datasets[3]. BigBird however, is able to extend upon BERT to be able to process sequences up to 8x longer while using similar hardware. These improvements come from the sparse attention mechanism used.

The sparse attention mechanism that BigBird employs makes use of 3 different components:

1. Random attention: Each query block attends to r random key blocks
2. Window local attention: Every query block at index j attends to the key blocks with index $j-(w-1)/2$ to $j+(w-1)/2$, including block j.
3. Global attention: Tokens g that attend to all other tokens in a sequence, and to whom all tokens attend to.

Finally, the sparse attention mechanism of BigBird means that queries attend to r random keys, $w/2$ tokens to the left and $w/2$ tokens to the right, and g global tokens (which can be chosen from existing tokens or extra added tokens). Through this attention mechanism BigBird is able to approximate self attention, while being able to scale linearly with sequence length.

2.3.3 Longformer

Longformer, Long Transfromer, was proposed as an improvement on the original transformer model (Vaswani et al. 2017) where memory and computational requirements scale quadratically with sequence length. Longformer continues from the RoBERTa (Y. Liu et al. 2019) released checkpoint, and is pretrained on long and short documents that include text from the Books corpus and English wikipedia sets used in RoBERTa.

Longformer's attention mechanism is a drop-in replacement for the original self attention (Beltagy et al. 2020), combining:

1. Sliding Window: First proposed by Wang et al. 2019, Longformer makes use of a fixed-size sliding window that surrounds each token. Given a fixed window size w, each token attends $\frac{1}{2}w$ tokens on each side
2. Dilated Sliding Window: Building on the sliding window attention, a dilated sliding window is used to increase receptive field without increasing computation. A window of size w has gaps of size dilation d
3. Global Attention: Task motivated global tokens are used on a select number of input tokens g which attend to every token, and have every token attend to it. This allows the model to generate full sequence representations while still maintaining linear complexity as the number of global tokens are relatively small

In transformers that follow the original self attention, the attention scores are determined following equation (5). This is an expensive computation due to the matrix multiplication of QK^T as both Q and K have n projections (Beltagy et al. 2020). The sparse attention mechanism of Longformer computes only a fixed number of diagonals of QK^T, and thus is able to scale linearly.

A visual representation of the attention weights given to each token by GPT-2, BigBird, and Longformer are illustrated in Figure 2.

[3]Books (Zhu et al. 2015), CC-News (Guu et al. 2020), Stories (Trinh et al. 2018), and Wikipedia

Corpus	Set		
	Training	Valid	Test
IsiZulu	3146.6	213.3	190.9
SeSotho	1429.5	182.1	213.3

Table 1: Dataset sizes, reported in thousands of words for training, validation, and testing sets. The IsiZulu dataset contains text gathered from the NCHLT isiZulu Text Corpora and the Isolezwe corpora, while the SeSotho dataset comes only from the NCHLT Sesotho Text Corpora

3 Experimental setup

3.1 Datasets

In this research we focus on language modelling for IsiZulu and SeSotho. The data used was pulled from two sources, namely: the National Center for Human Language Technology (**NCHLT**), and the **Isolezwe** newspaper.

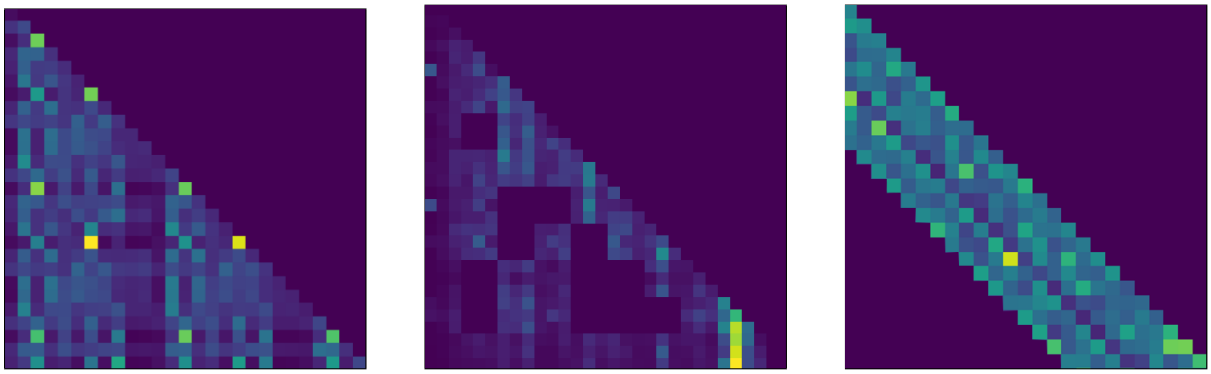

Figure 2: Attention weight plots for GPT-2 (left), BigBird (middle), and Longformer (right)

The data corpora from the NCHLT was made available by the South African Centre for Digital Language Resources (SADiLaR). The NCHLT corpus that was used contains text that includes articles and wideband speech from approximately 200 speakers per language, in all eleven of South Africa's national languages (Barnard et al. 2014). The data corpora was built up using over 50 hours of orthographically transcribed speech in each language, as well as data scrapes from national websites. Languages across South African Benue-Congo language family were processed, with the largest source coming from IsiZulu[4], and the second largest coming from SeSotho[5]. The corpora range from 1 to 3 million tokens.

Isolezwe is an isiZulu newspaper launched in 2002. News articles from this newspaper were scraped and consolidated by the Newstools initiative[6]. This dataset has a similar size to the NCHLT isiZulu Text Corpora, and was used in addition to this corpus to increase the size of the dataset and provide a second evaluation domain.

The datasets have been preprocessed by Mesham et al. 2021 to contain text that is representative of the language and free from erroneous or repetitive data. The datasets were split into training, validation, and test sets following an 80/10/10 split. Table 1 provides information on the dataset sizes and splits.

[4]Available at: https://repo.sadilar.org/handle/20.500.12185/321
[5]Available at: https://repo.sadilar.org/handle/20.500.12185/336
[6]Available at: https://github.com/newstools

3.2 Model Optimization

The BPE preprocessing was done using the Huggingface tokenizers library[7] and code provided by Mesham et al. 2021. In this research paper only the decoder architecture for each transformer model was used. This was done to permit left-context-only for language modelling (P. J. Liu et al. 2018).

Hyperparameter optimization was done experimentally through the use of the Weights and Biases (W&B) Sweeps. Each sweep consisted of twenty separate runs, where a different set of parameter values were assigned to the model following the Bayesian statistical model (Stigler 2018). Linear and log uniform distributions were used to test for different parameter values for each model. Some of the values tested include:

- Hidden Size: Log uniform distribution for values between 144 and 768
- Learning rate: Linear distribution for values between 10^{-4} and 10^{-8}, in increments of 10^{-1}
- Warmup steps: Linear distribution for values from 500 to 2500, incrementing by 500
- Block Size: Log uniform distribution from 8 to 64
- Number of random tokens: Linear distribution from 0 to 5, incrementing by 1

The training loop executed 5 epochs, and the set of parameters that resulted in the lowest loss when evaluating on the validation set was used for experimentation. The sweeps were conducted on all transformer models, with the results shown in Figures 5, 6, and 7. Furthermore, the BPE vocabulary size was set to 8000 and 2200 for isiZulu and SeSotho respectively, as this is where the best performance was found.

[7]https://github.com/huggingface/tokenizers

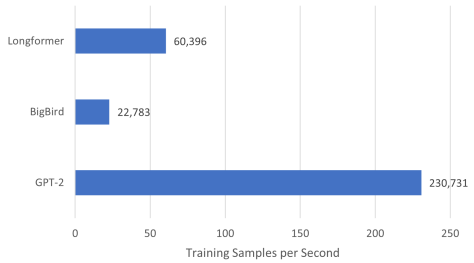

Figure 3: Amount of samples that were able to be processed per second when training for each model

3.2.1 GPT-2

In this paper, we train a GPT-2 model (from the architecture described in 2.3.1) on the language datasets mentioned in section 3.1. The model used was implemented using PyTorch through the open-source Huggingface transformers library[8]. We use WordPiece embeddings (Y. Wu et al. 2016), a tokenization algorithm that generates sub-word tokens.

The training data was inputted into the model in blocks of 128 consecutive tokens, with a batch size of 32. This was created using a sliding window of stride 16 as well as the truncation of sequences that were too long. The model was trained for 200k steps for the isiZulu dataset, and 50k steps for the seSotho dataset. This was done as it was found that the models stopped improving (the loss of the validation set did not decrease for four evaluation runs) after these respective training steps. The validation evaluation was conducted every 10k steps and 100 steps respectively.

8 attention heads and 8 hidden layers were used, with a hidden layer size of 768. An initial learning rate of 10^{-4} was used that followed a learning rate schedule that linearly decreased to zero during the course of training. This model performed best when a warmup period consisting of 2000 steps was implemented, where the learning rate maintained constant at 10^{-4}, and thereafter began its linear schedule.

3.2.2 BigBird

A custom autoregressive model, based on the architecture of BigBird (Section 2.3.2), was used as the basis for training in the experiments that were conducted. The aforementioned model was trained on the isiZulu and seSotho datasets, which were tokenized using a modified tokenizer that was trained on the South African Benue-Congo languages that were mentioned in Section 3.1. This model was also implemented using the Huggingface transformers library [8].

For BigBird, the training data was fed into the model in consecutive blocks of size 936 with a batch side of 64. Again a sliding window of stride 16 was used with longer sequences being truncated. The validation loss stopped improving after 28k steps for isiZulu and 8k steps for seSotho, and so training was only conducted for that long. The evaluation of the validation set took place every 1k and 500 steps respectively. The same learning rate used for the GPT-2 model was used again, however with a warmup phase of 1500 steps.

The model had 8 hidden layers and 8 attention heads, with a hidden layer size of 768. The attention dropout probability along with the hidden layer dropout probability was set to 0.1. Our BigBird model performed best when an attention block size of 8 was used alongside 4 random tokens. The number of global tokens was set to 2, and the strategy used was that of BigBird ITC (internal transformer construction) where already existing tokens were chosen to be global.

3.2.3 Longformer

Our Longformer model was trained using the autoregressive version of the model described in Section 2.3.3 to perform casual language modelling on isiZulu and seSotho. A custom tokenizer was trained on the language datasets using the LongformerTokenizer Huggingface library. A PyTorch implementation of the Huggingface Trainer API was used to train and evaluate this model.

The training input from both datasets were fed into the model in 768 consecutive blocks in batches of 32. For training, the tokenizer used a stride of length 16, and truncated sequences that were too long. Furthermore, the model was trained for 50k steps with validation every 1k on the isiZulu corpus, and 12k with validation every 500 steps on the seSotho corpus. After this time the model started to overfit the data and so training was stopped.

Our Longformer model had a hidden layer size of 768, with 8 attention heads and 8 hidden layers. During experimentation it was found that the model was relatively insensitive to changes on the number of hidden layers and attention heads, but still performed best when both were set to 8. A weight decay value of 0.3 was

[8] https://github.com/huggingface/transformers

used to ensure that the model did not report huge losses on the training set. The learning rate was set to 10^{-4}, with 1500 warmup steps, which then saw the learning rate linearly decrease to 0. Finally, and uniquely to Longformer, an attention window of size 24 (for all layers) demonstrated the best results. A dilation value of 1 was used, meaning that there was no dilation in the window size[9].

4 Results and Discussion

4.1 Results

The results for the test sets are given in Table 2. Of the three models, BigBird and Longformer performed relatively similarly, with GPT-2 performing the worst of the three.

On the isiZulu dataset, with the bigger BPC vocabulary size, Longformer had the best performance when compared to the two other models. When training, Longformer was able to achieve its lowest BPC at 50k optimization steps, compared to the 200K of GPT-2. BigBird was able to achieve its best accuracy on 22K fewer optimization steps, but was still not able to match that of Longformer.

On the smaller seSotho dataset, Longformer again outperformed all the other models, but was almost matched by BigBird (with a difference of only 0.02 in BPC evaluation). It achieved this result whilst using the same small vocabulary size as the other models, and with a similar number of optimization steps as BigBird.

In addition to the BPC valuations, Figures 3 and 4 illustrate the timing characteristics for each model. Although BigBird was able to reach its best performance in fewer optimization steps when compared to the other two models, it took the longest to train. BigBird did however show a greater change in training time as it progressed, while GPT-2 and Longformer began to plateau after the first hour of training. Longformer and GPT-2 performed similarly on this metric, however GPT-2 was the fastest out of all the models, being able to train around 230 samples per second, almost 10x more than BigBird and 4x more than Longformer.

[9]All code can be found at https://github.com/jaedonvs/Efficient-Transformer-Language-Models

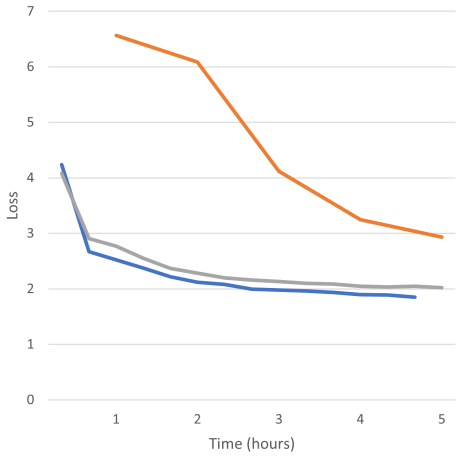

Figure 4: Validation loss of GPT-2 (grey), BigBird (orange), and Longformer (blue) while training on the isiZulu dataset

4.2 Discussion

The results show that the relative performance of the models is on par with what can be expected when compared to results done in similar experiments using text8 and WikiText2 datasets (Radford, J. Wu, et al. 2019). Moreover, our BigBird model performs comparatively when compared to Longformer as is shown in the original paper (Zaheer et al. 2020).

The BigBird and Longformer models consistently produced similar results. This could be explained through their attention mechanism and how it differs to the one implemented in GPT-2. Due to the agglutinative nature of Nguni and Sotho-Tswana languages, it can be hypothesised that the tokens closer to each other hold more significance in the meaning of the sentence than those far away. This could be used to explain why the sliding window attention strategy (used in both BigBird and Longformer) was able to outperform an attention mechanism that processed the entire sentence (as was used in GPT-2).

When it came to the training and evaluation times for each model, GPT-2 outperformed both BigBird and Longformer. We theorize that this is due to the simplicity of the attention mechanism used. The block sparse attention mechanism employed by BigBird requires significantly more overhead when deciding which tokens to attend to (choose a series of random tokens as well as defining the blocks to omit), which results in increased time for each optimization step. Similar reasoning can be applied to Longformer.

Model	IsiZulu			SeSotho		
	Steps	Vocab	BPC	Steps	Vocab	BPC
GPT-2	200K	8000	1.38	50K	2200	1.48
BigBird	28K	8000	1.32	8K	2200	1.36
Longformer	50K	8000	**1.27**	12K	2200	**1.34**

Table 2: Results from the casual language modelling objective performed on the isiZulu and seSotho test corpora. The isiZulu dataset contains text from the NCHLT and Isolezwe, while the seSotho dataset only contain text from the NCHLT corpus. Results are reported for the number of optimization steps taken to reach lowest validation loss, vocabulary size, and BPC (lower is better).

However, without the additional work of assigning random tokens, as well as a simplified window strategy, the Longformer model is able to train in a time closer to that of GPT-2 opposed to BigBird.

We found that, for each dataset, the language models that implemented a sparse attention mechanism performed better than the model that implemented a full attention mechanism. This supports our hypothesis that sparse attention improves on full attention, even on low resource languages.

5 Conclusion

The experiments conducted in this paper illustrate that through various optimization approaches and improved regularization, transformer models on small datasets are able to reproduce similar results to transformers on larger datasets. Further, model developments on small English datasets are also able to contribute to the success of language modelling on South African languages. All language models performed well, with GPT-2 being the best performing model in terms of time, while both Longformer and BigBird demonstrating impressive accuracy when evaluated on each dataset. Overall, Longformer was the best performing model when taking both speed and accuracy into account, which was as expected.

Our results also show that new transformer models are able to be accurately modified to perform on African languages, but that there are still improvements to be made. Furthermore, we showed that BPC is the most effective tokenization approach for open vocabulary language modelling, especially when applied to the agglutinative Nguni and Sotho-Tswana language datasets.

6 Acknowledgments

The basis of this paper was inspired by the work of Mesham et al. 2021, and supported by the South African Center for High Resource Computing.

References

Chen, Stanley F, Douglas Beeferman, and Roni Rosenfeld (1998). "Evaluation metrics for language models". In.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Mesthrie, Rajend and Mesthrie Rajend (2002). *Language in South Africa*. Cambridge University Press.

Mollá, Diego and Ben Hutchinson (2003). "Intrinsic versus extrinsic evaluations of parsing systems". In: *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?*, pp. 43–50.

Barnard, Etienne, Marelie Davel, Charl van Heerden, Febe Wet, and Jaco Badenhorst (Jan. 2014). "The nchlt speech corpus of the south african languages". In: *SLTU 2014*, pp. 194–200.

Zhu, Yukun, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 19–27. DOI: 10.1109/ICCV.2015.11.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). *Attention Is All You Need*. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

Liu, Peter J., Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer (2018). *Generating Wikipedia by Summarizing Long Sequences*. DOI: 10.48550/ARXIV.1801.10198. URL: https://arxiv.org/abs/1801.10198.

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. (2018). "Improving language understanding by generative pre-training". In.

Stigler, Stephen M. (2018). "Richard Price, the First Bayesian". In: *Statistical Science* 33.1, pp. 117–125. ISSN: 08834237, 21688745. URL: https://www.jstor.org/stable/26770983 (visited on 10/27/2022).

Trinh, Trieu H. and Quoc V. Le (2018). *A Simple Method for Commonsense Reasoning*. DOI: 10.48550/ARXIV.1806.02847. URL: https://arxiv.org/abs/1806.02847.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

Manning, Christopher D., Prabhakar Raghavan, and Schütze Hinrich (2019). "Introduction to information retrieval". In: Cambridge University Press.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. (2019). "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8, p. 9.

Wang, Zhiguo, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang (2019). *Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering*. DOI: 10.48550/ARXIV.1908.08167. URL: https://arxiv.org/abs/1908.08167.

Beltagy, Iz, Matthew E. Peters, Arman Cohan, and (2020). *Longformer: The Long-Document Transformer*. DOI: 10.48550/ARXIV.2004.05150. URL: https://arxiv.org/abs/2004.05150.

Chowdhary, KR1442 (2020). "Natural language processing". In: *Fundamentals of artificial intelligence*, pp. 603–649.

Ghojogh, Benyamin and Ali Ghodsi (Dec. 2020). *Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey*. DOI: 10.31219/osf.io/m6gcn. URL: osf.io/m6gcn.

Gillioz, Anthony, Jacky Casas, Elena Mugellini, and Omar Abou Khaled (2020). "Overview of the Transformer-based Models for NLP Tasks". In: *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pp. 179–183. DOI: 10.15439/2020F20.

Guu, Kelvin, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang (2020). *REALM: Retrieval-Augmented Language Model Pre-Training*. DOI: 10.48550/ARXIV.2002.08909. URL: https://arxiv.org/abs/2002.08909.

Liao, Yi, Xin Jiang, and Qun Liu (2020). *Probabilistically Masked Language Model Capable of Autoregressive Generation in Arbitrary Word Order*. DOI:

10.48550/ARXIV.2004.11579. URL: https://arxiv.org/abs/2004.11579.

Luo, Xiao, Haoran Ding, Matthew Tang, Priyanka Gandhi, Zhan Zhang, and Zhe He (Dec. 2020). "Attention Mechanism with BERT for Content Annotation and Categorization of Pregnancy-Related Questions on a Community QA Site". In: vol. 2020. DOI: 10.1109/BIBM49941.2020.9313379.

Provilkov, Ivan, Dmitrii Emelianenko, and Elena Voita (July 2020). "BPE-Dropout: Simple and Effective Subword Regularization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 1882–1892. DOI: 10.18653/v1/2020.acl-main.170. URL: https://aclanthology.org/2020.acl-main.170.

Zaheer, Manzil, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed (2020). "Big Bird: Transformers for Longer Sequences". In: DOI: 10.48550/ARXIV.2007.14062. URL: https://arxiv.org/abs/2007.14062.

Li, Junyi, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen (2021). *Pretrained Language Models for Text Generation: A Survey*. DOI: 10.48550/ARXIV.2105.10311. URL: https://arxiv.org/abs/2105.10311.

Mesham, Stuart, Luc Hayward, Jared Shapiro, and Jan Buys (2021). *Low-Resource Language Modelling of South African Languages*. DOI: 10.48550/ARXIV.2104.00772. URL: https://arxiv.org/abs/2104.00772.

Chaudhuri, Koushiki Dasgupta (Jan. 2022). *Building language models in NLP*. URL: https://www.analyticsvidhya.com/blog/2022/01/building-language-models-in-nlp/#:~:text=A%20language%20model%20in%20NLP,appear%20next%20in%20the%20sentence..

Jurafsky, Dan and James H. Martin (2022). "Evaluating Language Models". In: *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson.

A Appendix

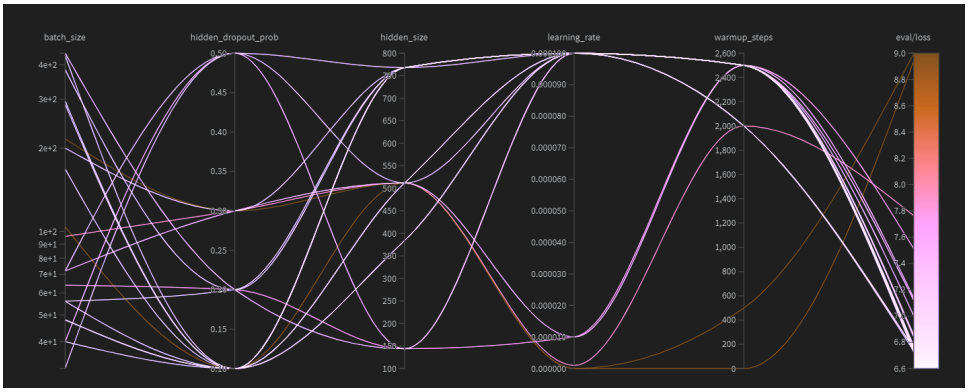

Figure 5: GPT-2 Sweep

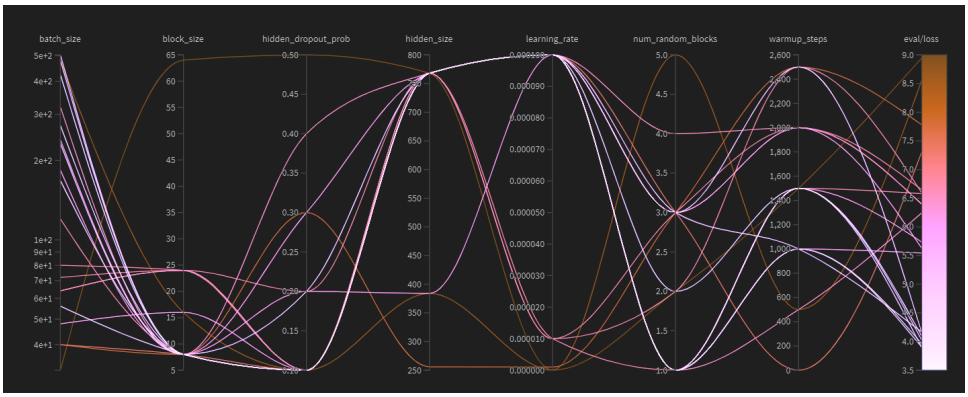

Figure 6: BigBird Sweep

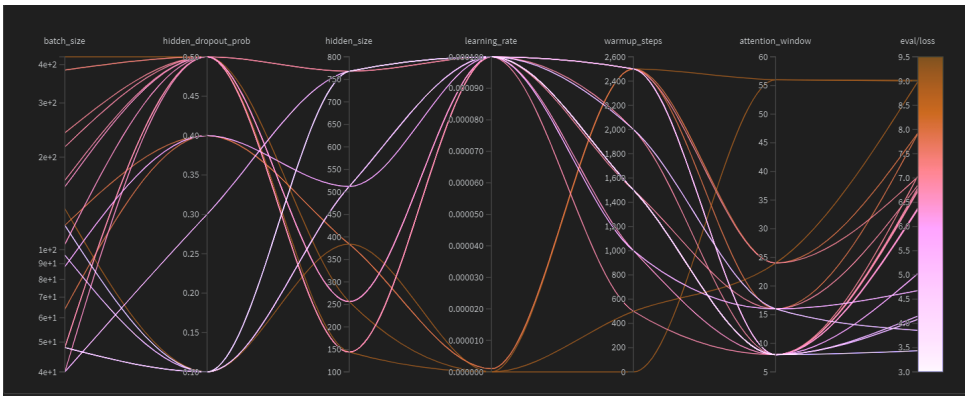

Figure 7: Longformer Sweep