LONDON'S GLOBAL UNIVERSITY

UCL

Language Model Adaptation for Low-Resource African Languages

Andrzej Szablewski[1]

MEng Computer Science

Supervisor: Prof. Pontus Lars Erik Saito Stenetorp

Submission date: 10th May 2024

[1] **Disclaimer:** This report is submitted as part requirement for the MEng Computer Science degree at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

Although thousands of languages are spoken worldwide, AI assistants powered by Large Language Models perform well only in those well-represented in terms of available data. Support for the remaining low-resource languages can be added to the existing Large Language Models through various adaptation techniques. However, the limited amount of high-quality training data favours data-efficient approaches, which this project investigates in the context of African languages. The initially studied tokeniser adaptation techniques allow for extending the vocabulary of a pre-trained model with the optimal number of new tokens from the previously unseen language. Subsequent experiments in data-efficient model adaptation explore how combining continuous pre-training and instruction-tuning may lead to more accurate cross-lingual model capabilities. Furthermore, the influence of adding English samples to the training dataset is studied. Although the model adaptation results are inconclusive, they open several directions for further research and underline the need for more efforts in low-resource natural language processing.

Contents

Acknowledgements

I am deeply grateful to Professor Stenetorp for his guidance and support throughout the project. His motivating suggestions and encouragement gave me a unique opportunity to explore the realm of academic research with an experienced guide and an exceptional mentor.

I would like to thank Jiayi Wang for the advice and introduction to the idea of in-context learning. I am grateful for countless conversations about multilingual language models, which helped me steer this project.

I appreciate the support from David Adelani and his tips regarding low-resource African languages.

Finally, I would like to thank my friends – Marek Masiak, who inspired my interest in natural language processing, and Szymon Duchniewicz, whose unwavering support was invaluable during the final months of the project.

Chapter 1

Introduction

1.1 Problem Description

In recent years, Large Language Models (LLMs) demonstrated remarkable natural language understanding and human-like text generation capabilities. They have been subsequently applied to classical Natural Language Processing (NLP) tasks and, notably, conversational chatbots. Popular LLM-powered services such as ChatGPT[1] and Microsoft CoPilot[2] quickly became default tools used by individuals in everyday tasks and enhancing productivity within companies. However, the benefits of these technological advances are not universally distributed. A significant disparity exists in the performance of LLMs across languages, depending on their presence in the model training data (Li et al., 2024), which nowadays come primarily from online sources such as Common Crawl repositories.[3] Hence, this performance gap is particularly evident between languages represented well on the internet (such as English, French, German, and Chinese) and the so-called *low-resource languages*, for which limited data and research attention are available (e.g. Igbo, Quechuan, Guarani). However, although poorly represented, low-resource languages are used by large numbers of native speakers from all around the world (e.g. Igbo – 44 million, Quechuan – 7 million, and Guarani – 6.5 million). On the other hand, many European languages are spoken by populations of similar size yet do not suffer from limited resources (e.g. Dutch – 30 million, Swedish – 13 million). Hence, the lack of adequate linguistic resources leads to uneven access to AI technologies, contributing to exclusion and marginalisation.

While there are significant efforts to bridge the gap between the two language groups, the low-resource setting makes the problem complex. It requires further

[1] https://chat.openai.com

[2] https://copilot.microsoft.com

[3] https://commoncrawl.org

research in several areas, including data annotation, benchmark development and data-efficient training. This project focuses on cross-lingual transfer learning and studies *language model adaptation* techniques applied to a subset of African languages. Such methods use existing, pre-trained models and leverage various strategies to link the already existing model capabilities with the ability to generate text in a new language (Artetxe et al., 2020). Combining the established adaptation methods such as those of Yong et al. (2023) with recently proposed improvements by Csaki et al. (2023) and continuously growing availability of training data (Oladipo et al., 2023), this project poses three major research questions and gradually develops corresponding answers.

1.2 Research Questions

Language Model adaptation usually involves a modification of its tokeniser as well as model fine-tuning (Artetxe et al., 2020; Aji et al., 2020). While the former is usually performed through token addition and the model's word embedding matrix expansion, recently, a new scheme based on token replacement has been proposed (Csaki et al., 2023). Moreover, the quality of language adaptation of Large Language Models benefits from a large amount of data in the adaptation language (Chen et al., 2023). While there have been numerous adaptation studies for high-resource languages such as (Yong et al., 2023), limited research explicitly explores the adaptation to several low-resource African languages. Finally, recent works have shown a positive influence of using a large amount of prompt-response formatted data on language adaptation (Csaki et al., 2024). However, there needs to be evidence that this approach works in a data-scarce environment.

This project aims to produce answers to the following three research questions (RQs):

- **RQ 1: What is the difference in the final model performance between tokeniser adaptation based on token replacement and token addition schemes?**

- **RQ 2: What is the influence of LLM adaptation in a low-resource setting on a set of downstream tasks in selected African languages?**

- **RQ 3: Does a combination of task-agnostic continuous pre-training, followed by instruction-tuning, achieve a better model adaptation performance for low-resource languages than the former solely?**

1.3 Report Structure

The structure of this report reflects the research nature of the project. Following this introduction, Chapter 2 introduces an overview of the necessary background information and an extensive review of the related works. A brief overview of the available data and evaluation benchmarks is supplied in the same chapter. Chapter 3 comprehensively describes the proposed adaptation methodology. It focuses on both tokeniser and model adaptations, providing the necessary high-level description of the implementation. Subsequently, Chapter 4 describes a concrete application of the adaptation methodology to a pre-trained Large Language Model using four African languages. The chapter includes a detailed description of the experimental and evaluation setup, allowing reproduction of the obtained results presented and discussed in Chapter 5. Finally, Chapter 6 summarises the contributions of the project and considers future directions of research.

Additional experimental setup information can be found in Appendix A, while the detailed results are presented in Appendix B. Finally, Appendix C contains additional project deliverables, including the Project Plan and the Interim Report. All of the experimental code required for the reproduction of the results have been made publicly available on GitHub.[4] The adapted tokenisers and language models are available on HuggingFace. [5]

[4] https://github.com/TheRootOf3/low-resource-language-model-adaptation

[5] https://huggingface.co/TheRootOf3/low-resource-language-model-adaptation

Chapter 2

Context

2.1 Background Information

2.1.1 Language Modelling and Word Embeddings

Language modelling is a fundamental task in natural language processing, essential to various applications such as machine translation, question answering and text summarisation. The grounds for the field have been set by the application of Shannon's information theory to measuring the entropy of a language (Shannon, 1951). A language model (LM) predicts the likelihood of sequences of words in a language. In particular, such a model can be used to predict the probability distribution of the next word w_n in a given context sequence $w_1, w_2, ..., w_{n-1}$:

$$P(w_n | w_1, w_2, ..., w_{n-1}). \tag{2.1}$$

Importantly, the probability of a sequence can be factorised into a product of the conditional probabilities:

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i | w_1, w_2, ..., w_{i-1}) \tag{2.2}$$

There are two common approaches to language modelling: purely statistical – based on word or sequence frequencies and statistical models, and neural – leveraging neural networks, which work particularly well for learning complex patterns in text. While they work differently, both groups require a significant dataset of text samples used for model training. One of the simplest statistical language models is the Bigram model, which approximates the probability of the next word using exclusively the probability of its predecessor, leveraging the Markov assumption:

$$P(w_n | w_1, w_2, ..., w_{n-1}) \approx P(w_n | w_{n-1}). \tag{2.3}$$

As an example, modelling a sequence `I love dogs` and using the Eq. 2.2 would involve the following computations:

$$P(\texttt{I love dogs}) \approx P(\texttt{I}|\texttt{<bos>}) \times P(\texttt{love}|\texttt{I}) \times P(\texttt{dogs}|\texttt{love}),$$

where `<bos>` is a special character indicating the beginning of a sequence. However, predicting the next word given only its predecessor has several issues, including simplicity in capturing dependencies in text, specifically the long-term ones. One solution to this problem is using more context words, leading to an n-gram model, which predicts the next word given the context of n words. While using 3-gram and 4-gram models improves the language modelling performance, this approach scales exponentially with n. On the other hand, a neural language model can be used to model longer sequences without explicitly learning the probability distribution over n consecutive words. While such models may differ in used neural architectures, they share a common optimisation goal of minimising the negative log-likelihood of the word sequence from Eq. 2.2, given the model parameters θ:

$$\arg\min_{\theta} -\sum_{i=1}^{n} \log P(w_i|w_1, ..., w_{n-1}; \theta). \tag{2.4}$$

While reasoning about words is plausible for humans, computers benefit from an alternative representation. A simple approach is to represent a pre-defined set of words using sparse one-hot encodings. In this method, a word is assigned an integer k and is then represented as a one-hot vector $\mathbf{v}$ with 1 in its kth entry ($v_{i=k} = 1$) and zeros everywhere else ($v_{i\neq k} = 0$). However, an average vocabulary V of a language model has tens of thousands of words, leading to a significant inefficiency in word representation. Furthermore, such word representations do not convey any semantic information.

In order to introduce meaning to word representations, the concept of word embedding has been introduced. Methods such as word2vec or GloVe add structure to the embedding space and represent words as multi-dimensional dense vectors. They exhibit relations, which can be further translated into semantic and syntactic relations between corresponding words. Nevertheless, the same word may have very different meanings depending on the context – e.g. the word `park` in `Let's go to the park` vs `You can't park here`. Therefore, advanced neural architectures have been developed to tackle this issue, resulting in contextual embeddings that adjust the meaning of words depending on their context.

2.1.2 Text Tokenisation

Dividing input text into sequences of words is not a straightforward task. A simple approach to this problem is to use the whitespace character as a natural word boundary. However, this has several disadvantages, including ignoring punctuation (e.g. both `why?` and `Anna's` would be considered single words) and splitting multi-word entities, such as `New York`. Critically, some languages, such as Chinese, do not use whitespace characters as word separators. Therefore, there is a need for a different scheme, possibly resulting in sub-word or multi-word entities, called *tokens*. Similarly, an algorithm splitting text into a sequence of tokens is called a *tokeniser*.

While the whitespace tokenisation does not work well with raw text, samples can be first normalised to remove unwanted punctuation or convert words to a standard format (e.g. lowercase). Furthermore, adding a set of tokenisation rules (e.g. keep a dot within digits, remove it if it is at the end of a sentence.) may result in sensible word split, such as:

`Dogs are barking, I am thinking.`

[`dogs, are, barking, i, am thinking`]

Moreover, techniques such as lemmatisation and stemming can be used to split words into morphemes further, reflecting their semantics and the syntax of a language better:

`Dogs are barking, I am thinking.`

[`dog, s, are, bark, ing, i, am, think, ing`]

With such tokenisation, a language model may learn that the token `ing` represents some continuity of action when combined with other tokens. Furthermore, the same tokenisation scheme applied to a new, previously unseen word, such as *sniffing*, may lead to a split `sniff` + `ing`. Although exposed to a new word, the model could infer that *sniffing* has something to do with the aforementioned continuity.[1]

Using a text tokeniser along language models requires deciding the contents of its fixed-size vocabulary V. One approach is to use the most frequent k tokens from a tokeniser training dataset, which represents the distribution of the language. Nevertheless, such a rule-based tokeniser exposed to a previously unseen word (e.g. `barkingandsniffing`) will mark it out-of-vocabulary (OOV) and either remove it or replace it with a special unknown token `<unk>`. This is a particular concern in a cross-lingual setting when a tokeniser designed for one language is used to tokenise texts in another. It is even more severe if languages differ in script (e.g.

[1] As well as dogs, of which the author is a particular admirer.

Latin, Arabic, Ge'ez). Hence, rule-based tokenisers do not scale well across languages without language-specific rules. Instead, subword tokenisers constructing a vocabulary based on statistical properties of the training dataset can be used (e.g. Byte-Pair Encoding, Unigram model).

Byte-Pair Encoding

Byte-Pair Encoding (BPE) is a data compression algorithm (Gage, 1994). It has been adopted as a tokenisation scheme because of it support for sub-word vocabulary and it does not no dependence on hardcoded tokenisation rules (Sennrich et al., 2016). A BPE tokeniser T gradually builds its vocabulary V from the small set of initial tokens until the earlier specified fixed vocabulary size k is reached (usually between tens of thousands and hundreds of thousands of tokens). Instead of using hardcoded rules, the algorithm learns tokens statistically, in the order corresponding to their frequency in a training dataset D. Each learned token is a combination of initial tokens or other already learned tokens. Such a combination is called a *merge*, and while the new token is added to the vocabulary V, a merge is added to the merge rules list R. Notably, the order of both the token and the merge rule matters. The high-level BPE tokeniser training algorithm is presented as follows:

1. Select the initial vocabulary V (usually the 256 ASCII characters) and the maximum vocabulary size k. Initialise the empty merge rule list R.
2. Split the training dataset D into initial tokens (character-level).
3. Find the most frequent combination of two adjacent tokens t_i, t_j in D.
4. Create a new token t_{ij} as a merge of t_i and t_j. Add t_{ij} to the merge rule list R
5. Replace all adjacent t_i, t_j in D with the new token t_{ij}.
6. Go back to 3. until $|V| = k$

As an example, the following one-sample training dataset D can be considered:

log frog blog dog,

the initial vocabulary V = [`b, d, f, g, l, o, r`] and $k = 10$. Splitting the sentence into initial tokens produces the following:

`l o g f r o g b l o g d o g`.

Counting the most frequent pair of two tokens, it is `o g` – 4 times. The first pair gets merged into a new token `og`, which is added to V, while the merge rule is added to R. After replacing `o g` with `og`, V, R and D look as follows:

`l og f r og b l og d og.`

$$V = [\texttt{b}, \texttt{d}, \texttt{f}, \texttt{g}, \texttt{l}, \texttt{o}, \texttt{r}, \texttt{og}], \quad R = [\texttt{o g}]$$

Continuing, the next most frequent token pair is `l og`, occurring twice:

`log f r og b log d og.`

$$V = [\texttt{b}, \texttt{d}, \texttt{f}, \texttt{g}, \texttt{l}, \texttt{o}, \texttt{r}, \texttt{og}, \texttt{log}], \quad R = [\texttt{o g}, \ \texttt{l og}]$$

Finally, all the remaining pairs occur only once, but following the alphabetical order `b log` is merged and added to the vocabulary.

`log f r og blog d og.`

$$V = [\texttt{b}, \texttt{d}, \texttt{f}, \texttt{g}, \texttt{l}, \texttt{o}, \texttt{r}, \texttt{og}, \texttt{log}, \texttt{blog}], \quad R = [\texttt{o g}, \ \texttt{l og}, \ \texttt{b log}]$$

When a trained tokeniser T is used to tokenise a text sample, it splits it into a sequence of initial tokens and subsequently replaces their combinations following the list of merge rules R.

A particularly advantageous aspect of BPE is that depending on the choice of initial vocabulary, it may eliminate the out-of-vocabulary problem. Using BPE across byte boundaries (byte-level BPE), allows to specify a small initial vocabulary of 256 values a byte can take and split longer Unicode characters into byte-level elements. While the Unicode characters are likely to be "reconstructed" and added to the vocabulary given their frequency, this trick permits avoiding using the `<unk>` token for OOV.

One of the tokenisation metrics is the token per word ratio, called *tokeniser fertility rate*. The ratio should be calculated on a previously unseen tokeniser dataset with a known number of words, such as a well-annotated treebank. Lower fertility means that words are, on average, split into less but longer tokens. This has several practical benefits in further language modelling. In particular, tokens consisting of multiple characters tend to have more semantic meaning than those of one or two characters. Furthermore, relatively low fertility means that longer text sequences can fit into the context window of the model, a particularly useful feature given the recent applications of Large Language Models to Retrieval-Augmented Generation (RAG) (Lewis et al., 2020).

2.1.3 Large Language Models

The recent advances in deep learning and available hardware allowed for a rapid increase in the size of neural language models. Hence, the term Large Language Models (LLMs) has been coined to refer to neural language models with a significant number of parameters[2] and trained on large text corpora. Most language models are based on the decoder of the Transformer neural architecture (Vaswani et al., 2023). Such models are referred to as *generative* and *autoregressive*, meaning they can generate new sequences and perform this by first predicting the next word given some context and then using the predicted word for further predictions. Hence, they are trained with a *causal language modelling objective*, as opposed to *masked language models*, such as BERT (Devlin et al., 2019), which learn word representations using bidirectional context (past and future words in a sequence) and therefore are not strict language models. To adapt a pre-trained LLM to a particular downstream task (e.g. question answering, machine translation), such a model can be fine-tuned on task-specific data, leveraging transfer learning. Critically, (Brown et al., 2020) showed that scaling model size induces emerging behaviours – model capabilities, which were not their explicit training objectives. Furthermore, such models have *In-Context Learning* (ICL) capabilities (Min et al., 2022), meaning they can be *prompted* to produce responses to user-specified queries by providing them as the context sequence. A sufficiently large model can subsequently infer from such context when performing tasks on which it was not initially trained. ICL can be used in a zero-shot setting, where only a prompt is provided, or a few-shot setting, where a prompt and a few examples are supplied to the model. Examples of in-context learning prompts are presented in Table 2.2.

Model	No. Params	Training Tokens	Year
OPT-1.3B (Zhang et al., 2022)	1.3B	0.18T	2022
Bloom 1B7 (Workshop et al., 2023)	1.7B	0.41T	2022
Olmo 1B (Groeneveld et al., 2024)	1B	2T	2024
GPT-3 (Brown et al., 2020)	175B	0.3T	2020
PaLM 2 (Anil et al., 2023)	540B	0.78T	2022
Llama 2 (Touvron et al., 2023)	70B	2T	2023

Table 2.1: Examples of Large Language Models. Notably, Bloom1B7 and PaLM 2 are multilingual models trained on a mix of data in 46 and over 100 natural languages, respectively.

[2] At the moment of writing this report, a model is considered large if it has about one billion parameters. However, the definition is likely to evolve, given the rapid progression of the field.

Nonetheless, LLMs require huge amounts of training data (order of 10^{12} bytes of text) and equally large amounts of dedicated computational resources. Hence, most are trained on almost exclusively English datasets containing no or minimal texts in African languages. As an example, the pre-training dataset of GPT-3 consisting of 570GB of text,[3] out of which 93.68% was English, 1.02% French and only 0.0011% Swahili, with no texts in Yoruba. This directly influences the performance of such models in these languages. A selection of LLMs is displayed in Table 2.1.

Setting	**Input Prompt**
Zero-shot example	Translate the following sentence to Polish: English: I love dogs. Polish:
Few-shot example (num shot = 2)	Translate the following sentence to Polish: English: I love dogs. Polish: Kocham psy. English: Dog is man's best friend. Polish: Pies jest najlepszym przyjacielem człowieka. English: Life is better with a dog by your side. Polish:

Table 2.2: An example of in-context learning with a zero-shot and a few-shot prompt for machine translation.

Generative Pre-Trained Transformer Architecture

The overall neural architecture of most Large Language Models follows the design of the Generative Pre-trained Transformer (Radford and Narasimhan, 2018). While the implementation details vary, reflecting modifications that increase the computational and modelling performance, the overall design is maintained. According to the Transformer's nomenclature, the GPT model is a decoder that accepts a fixed-length sequence of tokens and predicts a probability distribution over the next word. The entire neural network can be split into different elements (Fig. 2.1), and while they use similar building blocks, each element has a different function:

- **Word Embeddings**[4] $E \in \mathbb{R}^{|V| \times emb}$, where $|V|$ is the model vocabulary size,

[3] Data from `https://github.com/openai/gpt-3`

[4] The term *Word Embeddings* is used for historical reasons. While it suggests that embeddings represent *words*, LLMs split words into *tokens*, which are often sub-words or span over word boundaries.

and *emb* is the embedding dimension. Word embeddings are direct representations of tokens in the initial latent space of a model and, depending on $|V|$, may constitute a significant share of all model parameters.

- **Positional Encodings** $P \in \mathbb{R}^{ctx \times emb}$, where *ctx* is the length of the input sequence (model context). Positional encodings are either constant or learnable and are added to word embeddings to reflect their positions in the input sequence.

- **Transformer Blocks**, usually repeated multiple times and stacked vertically group of elements.

  - **Multi-Head Masked Attention**, a component used to capture mutual dependencies and similarities between the latent representation of tokens. Each attention head assigns a set of keys, queries and values to each element in a sequence using a set of corresponding parameters W_K, W_Q, W_V. Intuitively, they are applied to calculate how much each token representation in a sequence "attends" to all the previous ones, while multiple heads are used to learn many attention patterns. Subsequently, the attention outputs are projected for the further feedforward neural network.

  - **Feedforward Neural Network**, a shallow linear neural network necessary for transforming the outputs of the attention mechanism. A transformer block would only scale the input token representations without this component.

  - **Layer Normalisation Layers, Dropout Layers and Residual Connections**, which are used for improved optimisation stability and allow for training deeper networks.

- **Language Modelling Head**, a final element mapping the final network outputs into a normalised vector in $\mathbb{R}^{|V|}$. Depending on the model architecture, this element can be implemented as an independent linear neural network layer or reuse the word embeddings. A softmax normalisation is finally applied to represent the probability distribution over the next token.

The architecture size can be modified vertically (more transformer blocks) and horizontally (larger model context *ctx* and embedding space *emb*, more extensive vocabulary $|V|$). A decent understanding of the design and function of each of these components and tokenisation is critical to comprehending the model adaptation method proposed in this work.

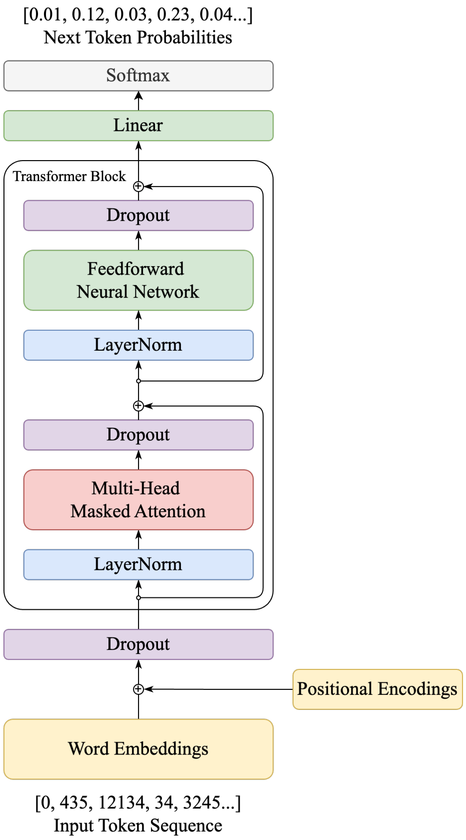


Figure 2.1: The neural architecture of GPT2 (Radford et al., 2019).

2.1.4 Overview of Chosen African Languages

The lack of textual resources is a common problem for most languages spoken locally in Africa. While there are significantly more low-resource languages than those with easily available data, the scope of this study needs to be narrowed. The choice of the explored languages is motivated by the availability of text corpora, suitable benchmarks and the prior work in the field. In particular, significant progress in up-sourcing has been achieved through the collaboration between the UCL Natural Language Processing group members and the NLP researchers from Masakhane[5] – a grassroots organisation working specifically with African languages. Hence, building on the group's efforts and leveraging the connection to the African network make it natural to focus on the languages they particularly explored. Therefore, *Hausa*, *Yoruba*, and *Igbo* are selected. In addition to these three national languages of Nigeria, *Amharic* – spoken in Ethiopia, is chosen to study how the proposed adaptation framework applies to languages with a non-Latin script. Table 2.3 presents an overview of the chosen languages.

[5] `https://www.masakhane.io`

Language	Code	Script	Family	Subgrouping	#Speakers
Amharic	amh	Ge'ez	Afro-Asiatic	Ethio-Semitic	60M
Hausa	hau	Latin	Afro-Asiatic	Chadic	88M
Igbo	ibo	Latin	Atlantic-Congo	Benue-Congo	44M
Yoruba	yor	Latin	Atlantic-Congo	Benue-Congo	47M

Table 2.3: Overview of chosen African languages. Numbers of speakers refer to both L1 and L2 speakers, and come from Wikipedia. Notably, all languages considered have more L1 and L2 speakers than German.

2.2 Related Works

2.2.1 LLM Language Adaptation Methods

Due to the large amount of data required for training a Large Language Model, developing dedicated models for low-resource languages is not viable. Hence, multilingual pre-training is one approach to developing Large Language Models for low-resource languages. Numerous attempts include models such as Bloom (Workshop et al., 2023), XLM-R (Conneau et al., 2020), mT5 (Xue et al., 2021), and PaLM 2 (Anil et al., 2023), which have been trained on a mix of English datasets and data in other languages. Although models trained this way display benefits in mono- and cross-lingual tasks, adding support for a language previously missing from the pre-training dataset involves costly re-training of the entire model. Furthermore, while these models display a decent performance in cross-lingual scenarios, Chang et al. (2023) showed that their performance in monolingual tasks improves only up to a certain number of added languages, after which it degrades. Nonetheless, the limited amount of language data is the primary reason for the poor performance of most multilingual LLMs in low-resource languages. Hence, a concept of *model adaptation* has been introduced, where an already existing pre-trained language model (possibly multilingual) is adapted to a new language through fine-tuning and related data-efficient methods.

One of the challenges in such adaptation is the often distinct vocabulary used in the model and the new language. Works such as (Artetxe et al., 2020; Aji et al., 2020; Pfeiffer et al., 2021) show the possibility of adding new tokens to a tokeniser and re-training the word embeddings in a model to reflect the new vocabulary. By freezing some or all of the remaining parameters (also referred to as the transformer body) and continuously pre-training with the original objective but data in a new language, the authors matched the performance of models pre-trained jointly on a bilingual dataset. Furthermore, Ács (2019) extensively studies a BPE tokeniser fertility using treebanks in various languages. Subsequently, Rust et al. (2021) explore the performance gap between multilingual and monolingual language models concerning their performance on monolingual tasks. The authors not only find that a designated monolingual tokeniser is key to better performance but also confirm that replacing the multilingual model's tokeniser with a dedicated monolingual version improves the model's performance.

Yong et al. (2023) demonstrate adapting a series of BLOOM models to new languages, following different fine-tuning strategies. Comparing the continuous pre-training with parameter-efficient fine-tuning techniques such as MAD-X (Pfeiffer et al., 2020) or $(IA)^3$ (Liu et al., 2022), they discover that adapter-based methods outperform the continuous pre-training only when the models under adaptation are sufficiently large (>3B). Furthermore, they experimentally show that a dataset of at least 100M tokens is required for a successful adaptation. On the other hand, Chen et al. (2023) extend the notion of adaptation even further. They show that regularly resetting model embeddings during pre-training leads to an increased *language plasticity* of the model, making it more suitable for further language adaptation. This method allows for a language and task adaptation through a sole training of the new embedding layer (for a new language) and fine-tuning the transformer body (for a dedicated downstream task). Notably, they report a significant increase in model performance when the size of the language adaptation dataset reaches 10M tokens. However, this method requires an appropriately pre-trained model, which is out of this project's scope.

In their work, Csaki et al. (2023) experiment with instruction-tuning as an adaptation technique for two high-resource languages: Hungarian and Thai. Their method involves initial continuous pre-training of a model in task-agnostic language data, followed by instruction-tuning with prompt-response formatted dataset. Furthermore, they experiment with replacing tokens in the model's vocabulary. Although promising results were achieved, their method uses huge datasets (100GB per language) unavailable in the low-resource setting. Finally, Lin et al. (2024) show a joint adaptation to more than 500 languages through continuous pre-training of the base 7B model. The proposed method achieves the best results for a range of low-resource methods on few-shot text classification tasks. While it does not use instruction-tuning datasets, the authors tune only a small set of overall model parameters and the embedding matrix.

Parameter-efficient fine-tuning techniques (PEFT) are used exceptionally well in data-deficient scenarios. Inserting adapter parameters into the architecture of pre-trained models allows updates of a limited selection of model parameters with fewer steps, requiring less training data. One example is Low-Rank Adaptation (LoRA) (Hu et al., 2022), which decomposes selected parameters into products of lower-rank matrices. LoRA updates only the decompositions, freezing the rest of the model parameters. Authors motivate the method with an intuition that optimising a full model updates a limited number of parameters. Critically, they show that fine-tuning only the LoRA-targetted attention keys and queries matches the performance of full model fine-tuning. While LoRA replaces parameters with smaller alternatives for fine-tuning, MAD-X introduces bottleneck adapters between existing parameters. Such an approach may limit the model's performance, and hence, $(\mathrm{IA})^3$ introduces new learnable parameters for element-wise rescaling of activations in feedforward layers. Given that PEFT methods are used in some works for language adaptation, studying their further application in the low-resource regime may be an exciting research direction.

2.2.2 Training Data

High quality data comes from human-crafted *corpora* such as Project Gutenberg[6], Wikipedia, or historically, the Brown Corpus. However, the lack of similar resources for many languages poses a significant challenge. While automatic machine translation approaches could be used to translate English corpora to other languages, they overlook the important cultural context, which is often implicitly present in locally used languages. Another difficulty in developing human-crafted corpora is the need for a diverse set of native speakers. Despite the superior quality of manually crafted data, using such corpora no longer suffices in training the growing size LLMs. Instead, the largest source of training data are CommonCrawl[7] repositories containing archived internet web pages. Given its nature, the CommonCrawl corpus contains data in most low-resource languages as long as web pages are written in those languages. However, the data is heavily polluted with non-natural texts, undesired content and duplicates. Therefore, several cleaned versions of the CommonCrawl dataset have been published, including mC4 (Raffel et al., 2019), a multilingual version of the Colossal Clean Crawled Corpus (C4). The corpus contains data for 108 languages, including some spoken in Africa – particularly Amharic, Hausa, Igbo and Yoruba. Furthermore, Oladipo et al. (2023) further revised and extended the mC4 contents, focusing exclusively on African languages. Their analysis suggests a good quality of mC4 contents and results in a new document-level dataset – WURA. On the other hand, ImaniGooghari et al. (2023) published Glot500-c, a large combina-

[6] `https://www.gutenberg.org`

[7] `https://commoncrawl.org`

tion of available multilingual datasets covering 500 languages with a particular focus on those lacking good quality training data sources.

On the other hand, instruction-following capabilities of a model can be induced through *instruction-tuning* – a set of techniques aligning a pre-trained language model to respond to user instructions. These techniques include simple fine-tuning as well as methods based on Reinforcement Learning with Human Feedback (RLHF) (Christiano et al., 2023; Ouyang et al., 2022). Usually, instruction-following is achieved by training on prompt-response formatted data. At the beginning of this project, a limited number of instruction-tuning datasets were available in low-resource languages, while almost all of them were machine-translated. However, the recently published Aya Dataset (Singh et al., 2024) contains varying prompt-response formatted samples for the selected African languages.

2.2.3 Model Evaluation & Benchmarks

Large Language Models can be evaluated using intrinsic and extrinsic metrics. While intrinsic evaluation shows how well the model achieves the training objective – language modelling, and therefore minimising the negative log-likelihood, the extrinsic evaluation involves measuring model performance on a set of downstream tasks. The most common intrinsic metric is perplexity (PPL), defined as the exponentiated average negative log-likelihood of a sequence of tokens $w_1, w_2, ..., w_n$:

$$PPL(w_1, w_2, ..., w_n) = e^{-\frac{1}{n}\sum_{i=1}^{n}\log P(w_i|w_1,w_2,...,w_{i-1})} \tag{2.5}$$

Intuitively, perplexity measures model certainty about the next predicted word. Measuring model perplexity requires a sequence of tokens, which was not present in the training dataset. Furthermore, such a sequence should have a similar distribution to the training dataset, reflecting the training objective well. Hence, a dataset used for model training is usually split into a training part – fed to the model for weights optimisation, and a validation part – used to measure the model performance on unseen data.

On the other hand, metrics used in the extrinsic evaluation are task-related, while such evaluations require task-specific datasets. The development of benchmark datasets for low-resource languages is crucial to the progression of the field, given the need for an independent and fair evaluation of the proposed methods. Researchers from Masakhane have contributed to the work on developing benchmarking datasets for African languages in common NLP tasks, such as topic classification, question answering, and named entity recognition (NER). In their work, Ojo et al. (2024) evaluate the largest LLMs with the In-Context Learning approach. While the models are prompted in a zero-shot setting, samples from benchmark datasets can also be supplied as examples in a few-shot regime and used for model fine-tuning preceding the evaluation. Table 2.4 presents the available benchmarks for African languages.

Benchmark	**Task**	**amh**	**hau**	**ibo**	**yor**
AfriSenti^a	Sentiment Classification	✓	✓	✓	✓
AfriQA^b	Question Answering		✓	✓	✓
MAFAND-MT^c	Machine Translation	✓	✓	✓	✓
AfriMTE^d	Machine Translation		✓	✓	✓
MasakhaNews^e	Topic Classification	✓	✓	✓	✓
SIB-200^f	Topic Classification	✓	✓	✓	✓
MasakhaNER^g	Named Entity Recognition	✓	✓	✓	✓

Table 2.4: Available benchmarks for downstream tasks in African languages. *a*: (Muhammad et al., 2023), *b*: (Ogundepo et al., 2023), *c*: (Adelani et al., 2022a), *d*: (Wang et al., 2024), *e*: (Adelani et al., 2023), *f*: (Adelani et al., 2024), *g*: (Adelani et al., 2022b).

Chapter 3

Methodology & Implementation

The proposed methodology of adapting pre-trained Language Models trained in a *base language* to a previously unseen *target language* consists of two essential stages: *Tokeniser Adaptation* and *Model Adaptation*. The former introduces modifications to the tokenisation scheme by replacing or adding tokens specific to the target language. Furthermore, it adjusts the embedding parameters of the pre-trained model accordingly. On the other hand, Model Adaptation involves continuous pre-training of the model and subsequent instruction tuning on language-specific datasets. Since this study focuses on model adaptations to African languages, both stages assume a low-resource and low-compute environment.

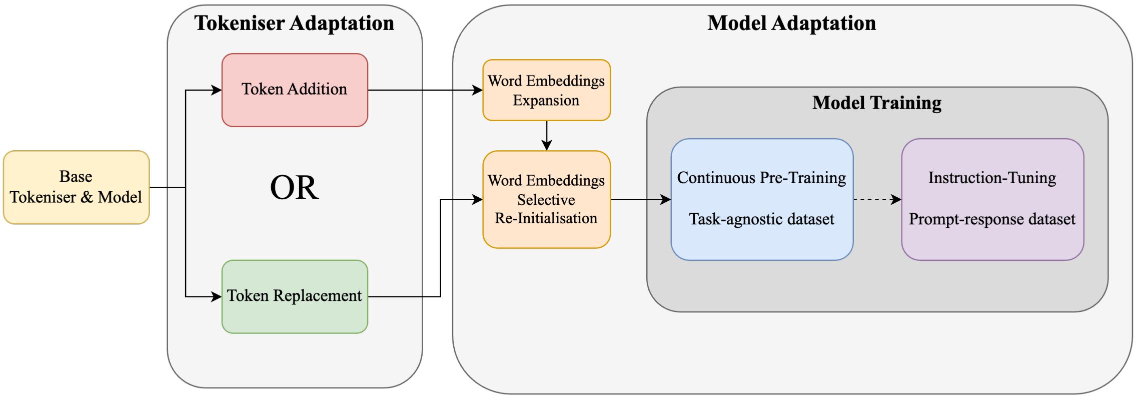


Figure 3.1: Overview of the adaptation methodology.

3.1 Tokeniser Adaptation

A Language Model based on learnable embedding parameters models the text through tokenisation and subsequent mapping to the embedding space. Using multi-dimensional representations of tokens to learn relations between them, the model implicitly achieves the language modelling objective and can generate new text given the provided context. Hence, the embedding space of a trained model is tightly related to the text tokenisation scheme used during model training.

However, applying two different tokenisation schemes to the same input text results in two distinct sequences of tokens (e.g. `think` + `ing` vs `thin` + `king`). Furthermore, this leads to different embedding space representations of the original text and, consequently, inference abilities of a model. In particular, using a tokeniser trained in language A to tokenise text in a different language, B may result in tokenisations which do not follow the token distribution of the latter and, therefore, do not reflect its morphological properties. Subsequently, such tokenisations are likely to fall out of the distribution of token sequences learned by the model, resulting in inaccurate inference in practice. Intuitively, the set of tokens an English-dedicated tokeniser can produce includes common English morphemes or words. However, the same tokeniser applied to text in a language with very different vocabulary, such as Yoruba, will not result in a tokenisation consisting of Yoruba morphemes or words but mostly meaningless (in the sense of Yoruba language) combinations of letters representing English tokens. The lack of Yoruba-specific tokens in the English tokeniser's vocabulary causes this issue. An adaptation method is proposed to mitigate the lack of language-specific tokens in a tokeniser used in model pre-training.

A pre-trained Language Model uses an original *base tokeniser* T_b trained mostly using a monolingual dataset D_b in a base language L_b. To adapt T_b to a target language L_t, it is necessary to use a sample of a monolingual, task-agnostic dataset in a target language, D_t, which is large enough to represent a great majority of L_t's vocabulary, phrases and its overall linguistic structure. The next step is to create a new, *target language tokeniser* T_t. To ensure the similarity of the nature of learnt vocabulary between the base and target language tokenisers, the new tokeniser uses the same algorithm as the base tokeniser (e.g. Byte-Pair Encoding, Unigram Model). While training a BPE tokeniser is a computationally expensive and time-consuming process, it must be performed only once to obtain a list of vocabulary specific to the target language.

Importantly, the target language tokeniser is not directly used in the Model Adaptation stage (Section 3.2) because it would inherently involve relearning all token embedding parameters of the pre-trained models. Given that the embedding matrix contains many learnable parameters of a Language Model, learning them from scratch requires correspondingly much training data, which is limited for low-resource languages.

Instead, a fixed number of k tokens from the target language vocabulary V_t is incorporated into the base tokeniser. This work studies two approaches of base tokeniser modifications:

1. **Increasing the vocabulary size V_b and extending it with new tokens from the target language tokeniser**, a concept common across literature (Artetxe et al., 2020; Pfeiffer et al., 2021). While it preserves all existing tokens in V_b, it also involves increasing the capacity of the model's embedding matrix by increasing its dimension.

2. **Replacing chosen tokens from the base tokeniser with the new tokens from the target language tokeniser** introduced by Csaki et al. (2023) and further refined in this work. Although it removes some tokens from the original vocabulary, it keeps the size of V_b constant and does not introduce additional computation during training and inference.

One of the advantages of using a BPE tokeniser is its algorithmic detail, which makes the tokeniser learn tokens in the order reflecting their frequencies in the tokenisation dataset. Hence, given the goal of minimising the ratio of tokens per word (fertility), adding or replacing the most frequent k tokens unique to the target language tokeniser T_t is reasonable.

Furthermore, it is essential to note that neither approach modifies the dimension of the embedding space itself, keeping it identical to that in the pre-trained model. Otherwise, additional adapters or modifying further layers of the model (e.g. positional encodings) would be required. Intuitively, this would involve longer training in the Model Adaptation stage and, therefore, is out of scope for this project.

3.1.1 Token Replacement

The replacement process follows the idea proposed by Csaki et al. (2023). It can be split into initially removing a token from the vocabulary of the base tokeniser and subsequently associating a new token from the target language tokeniser's vocabulary with the removed token's ID. However, some tokens cannot be easily removed in BPE tokenisers due to their presence in a list of merge rules. Removing a token, which is a building block of another token, would involve further undesired modifications to the vocabulary. Luckily, the least frequent tokens will usually be *final*, not constituting any merge rules. On the other hand, adding new tokens to V_b involves adding their merge rules from the target language tokeniser.

The most frequent non-overlapping k tokens from the target language tokeniser T_t are used to choose the set of new tokens. The opposite strategy can be applied to selecting tokens to remove in the base tokeniser T_b. Intuitively, selecting the least frequent non-overlapping k tokens influences the tokenisations of texts in the base language L_b the least. However, in addition to being non-overlapping and least frequent, tokens selected for removal must also be final. This is crucial to avoid the abovementioned issue of removing tokens in the vocabulary but leaving them in some of the merge rules of the base tokeniser.

On the other hand, adding the most frequent non-overlapping tokens from the target language tokeniser to the base tokeniser has a great advantage: their merge rules can be reused and integrated into the new tokeniser. Replacing merge rules is possible due to the design of the replacement scheme and the properties of BPE. Firstly, the tokenisers share the same initial vocabulary, which guarantees that a merge rule of any token added to the vocabulary during training will consist of either tokens from the initial vocabulary or recursively tokens which were built this way. Furthermore, using the most frequent tokens from T_t guarantees that their merge rules can only use either initial tokens, other tokens present in the k most frequent tokens from V_t, or tokens already present in V_b with a lower id (because of the non-overlapping requirement).

The token replacement algorithm is presented below:

1. Assuming the following:
   - A base tokeniser T_b with a vocabulary list V_b and a merge rules list R_b.
   - A target language tokeniser T_t with a vocabulary list V_t and a merge rules list R_t.
   - A number of tokens to replace $k_{replace}$.
2. Create a list V_{new} containing the IDs of $k_{replace}$ first tokens from V_t which are not in V_b.
3. Create a list $V_{b-non-overlapping}$ containing the IDs of tokens from V_b which are not in V_t.
4. Create a set $V_{b-final}$ containing the IDs of final tokens from V_b.
5. Create a list V_{old} of the IDs from $V_{b-non-overlapping}$, which are also in $V_{b-final}$. Remove all elements but the last $k_{replace}$ ids from V_{old}.
6. For an index i ranging from 1 to $k_{replace}$:

(a) Find the $(k_{replace} - i)$th token from V_{old} in V_b and replace it with the ith token from V_{new}.

(b) In R_b, find the merge rule r_b generating the $(k_{replace} - i)$th token from V_{old}.

(c) In R_t, find the merge rule r_t generating the ith token from V_{new}.

(d) Replace r_b with r_t in R_b.

3.1.2 Token Addition

Similarly to the token replacement approach, token addition considers the most frequent k non-overlapping tokens from the vocabulary of a target language tokeniser. The vocabulary V_b is extended by k selected tokens and added in the order reflecting their frequencies.[1] Tokenisers based on the Byte-Pair Encoding algorithm treat manually added tokens differently from those created during tokeniser training because the former lack corresponding merge rules. When encoding input text, a BPE tokeniser first searches for occurrences of added tokens in text and, if found, substitutes them with added token placeholders. Once this process is done, the tokeniser encodes the input text using ordered merge rules, gradually building the final tokenisation and omitting the already tokenised added token placeholders. Therefore, no merge rules are added.

Increasing the vocabulary size involves resizing the model's embedding matrix. Such addition of learnable parameters influences the duration of training and subsequent inference and may result in slower model learning. Although optimising the model efficiency is outside this project's scope, it is still worth adjusting its architecture, considering the details of the hardware used for model training and inference. In the commonly used AI hardware architectures, the matrix multiplication performance (*multiply-add* operations) depends on matrix dimensions. Hence, the size of an embedding matrix is increased by a number of added tokens rounded up to a suitable multiple of a power of 2.[2]

3.2 Model Adaptation

Model adaptation consists of two main elements: modifying the embedding matrix to reflect the changes introduced during tokeniser adaptation and a subsequent model

[1] Depending on the implementation details of a Byte-Pair Encoding tokeniser, the ordering of added tokens may influence the encoding time.

[2] Link to GPU architecture documentation.

training on target language data.

3.2.1 Embedding Matrix Initialisation

Extending the already pre-trained embedding matrix with incorrectly initialised additional entries may influence the stability of fine-tuning. Therefore, they need to be initialised adequately. On the other hand, the token replacement approach does not extend the embedding matrix. However, an unmodified embedding of a replaced token carries the semantic meaning of its predecessor. While the model is trained in the further stages to re-learn the correct meaning of new tokens, re-initialising the replaced embeddings will eliminate their initial bias (Kocmi and Bojar, 2017). Multiple initialisation approaches have been considered:

- **Zero Initialisation** – sets an embedding to a zero vector $\mathbf{0}$.
- **Gaussian Initialisation** – uses samples drawn from a normal distribution $w \sim \mathcal{N}(\mu = 0, \sigma^2)$ to initialise embedding parameters. σ^2 is usually chosen to be small (∼1e-2).
- **Xavier Initialisation** – initialises parameters according to a uniform distribution parametrised with the number of learnable parameters in the target layer and the preceding layer.
- **Averaged Token Embeddings** – uses an average of the remaining, pre-trained embeddings to initialise a new embedding. Two variants of embedding averaging are considered:
    - Averaging all of the remaining embeddings $\{e_1, e_2, ..., e_n\} \in E$:

      $$e_{new} = \frac{1}{n}\sum_{i=1}^{n} e_i,$$

      which has been original proposed by Eric Mitchell[3] and results in a theoretic bound on the Kullback-Leibler Divergence between an unmodified pre-trained model and the one after embedding matrix modifications.
    - Averaging embeddings of tokens into which the new token t_{new} would be split under the base tokenisation. If

      $$T_b(t_{new}) = \{t_1, t_2, ...t_m\},$$

[3] Link to the mentioned thread.

with corresponding embeddings $\{e_1, e_2, ...e_m\}$, then

$$e_{new} = \sum_{i=1}^{m} e_i.$$

This method has been suggested by Csaki et al. (2024), who showed that it leads to a faster model convergence during training.

A method that introduces slight noise in the overall model parameter initialisation outperforms zero initialisation. Moreover, Hewitt (2021) showed that zero initialisation might lead to significant optimisation instabilities and, subsequently, a large Kullback-Leibler Divergence between an unmodified pre-trained model and the one after embedding matrix modifications. However, the author also suggested that introducing slight noise in the embedding initialisation results in a similar problem to zero initialisation. While one approach to tackle this issue is to use an average of all existing embeddings, this work applies its refinement proposed by Csaki et al. (2024) based on averaging only selected tokens. Using an arithmetic average of token embeddings produced by the original tokeniser results in averaging over embeddings of subword units, which intuitively may bear some semantic properties related to the new token.

3.2.2 Model Training Methodology

A pre-trained model with modified embeddings is trained in a pipeline consisting of two stages, each with a different goal. Initially, the model is trained on a larger corpus of long, task-agnostic samples in a target language with a causal language modelling objective. This *continuous pre-training* approach aims to adapt the model's parameters to the new language and can be viewed as a language-tuning stage. Namely, it aims to adjust the existing word embeddings and learn the replaced or added ones. Nevertheless, pre-training usually involves a significant amount of data (e.g. pre-training dataset of OPT – 180B tokens, Llama 2 – 2T tokens). However, the available resources for many African languages are limited (less than 1B tokens) and differ in quality. Hence, this work uses a maximum of 100M tokens in a target language for this stage. Furthermore, it follows the results from the study by Yong et al. (2023), who showed a significant performance increase between models trained on an adaptation dataset of 10^7 and 10^8 tokens. Furthermore, a model is trained only on a single epoch, lowering the computational requirements for such adaptation. All data samples are tokenised with a tokeniser adapted to the target language and data-packed, meaning each sample spans the entire context window of the model.

Recent studies showed that fine-tuning a model on multitask prompts in a tar-

Task-agnostic sample	**Prompt-response sample**
Text: The Black Stars have been eliminated from the 2019 AFCON in the round of 16 after being defeated on penalties – The match which ended 1-1 after extra time saw a first half goal from team captain, Dede Ayew, being disallowed by the referee Victor Gomes – Ghanaians who were incensed by the decision took to social media to insult...	**Prompt**: What are the biggest challenges facing space exploration today? **Response**: Space exploration faces several significant challenges, both technological and financial. Some of the biggest hurdles include: Developing new propulsion technologies: Current rockets...

Table 3.1: Examples of a task-agnostic and prompt-response text. The task-agnostic samples comes from the WURA dataset, while the prompt-response come from the Aya dataset.

get language further improves the adaptation efficiency (Yong et al., 2023; Csaki et al., 2023). Hence, the second stage combines samples from a smaller, prompt-response dataset. This *instruction-tuning* stage leverages the same CLM objective and is used to enhance the instruction-following capabilities of the model. Similarly, if they even exist, prompt-response datasets for many African languages are significantly smaller than for English. Nevertheless, given the high quality of usually human-crafted instruction-tuning datasets, this stage works as an additional step in language-tuning. However, the tokenised concatenations of prompt-response pairs are not data-packed.

3.2.3 Avoiding Catastrophic Forgetting

Catastrophic forgetting is a phenomenon observed in artificial neural networks, where learning new information causes the sudden and drastic erasure of previously acquired knowledge. Using target language data to fine-tune an English pre-trained language model is an instance of such a procedure. It may result in a significant loss of English modelling abilities. One proposed solution to this problem is adding instances used in pre-training to the fine-tuning dataset. Extending the fine-tuning dataset with samples that were used to learn the pre-trained distribution of the model helps to prevent catastrophic forgetting (Kirkpatrick et al., 2017). Given that the language adaptation aims to adapt a pre-trained model to a new language while leveraging the knowledge learnt during pre-training, preserving the original model capabilities could also translate into better downstream task performance in a target language. In addition to tokeniser modification strategies, this work explores how the amount of English data in the dataset used for continuous pre-training influences the model performance. Table 3.2 presents three different proportions used.

Furthermore, in the instruction-tuning stage, a proportion of 75% Target language and 25% English is used.

Variant	**Target language**	**English**
Variant 1	100%	0%
Variant 2	75%	25%
Variant 3	50%	50%

Table 3.2: Three variants of the continuous pre-training dataset.

3.2.4 Learnable Model Parameters

Another approach to deal with catastrophic forgetting is fine-tuning only a selection of model parameters. Moreover, fine-tuning all model parameters intuitively requires more extended training and more data, which is limited in the setup of this project. Some works have shown that it is possible to freeze some parameters and update only the remaining ones. On the other hand, approaches such as LoRA offer a comparable fine-tuning accuracy achieved through tuning a small set of additional parameters. Such parameter-efficient training results in faster convergence and is better at preserving existing knowledge (Hu et al., 2022). Hence, this work combines freezing, traditional fine-tuning, and LoRA-based methods. Targetting attention queries and values with LoRA is particularly useful because these parameters directly capture dependencies between tokens and their further representations. Furthermore, a method similar to that proposed by (Lin et al., 2024) is applied, where in addition to the parameters targetted by LoRA, the embedding matrix is fine-tuned without any decomposition. Moreover, positional encodings and the language modelling head (if not the same as the embedding matrix) are tuned to reflect the different from English morphology of the target language. While fine-tuning these vast sets of parameters (word embeddings can constitute even 50% of all model parameters) diminishes the advantages of LoRA, it constitutes a necessary language adaptation of word embeddings. All the remaining parameters are frozen, and both continuous pre-training and instruction-tuning stages tune the same set of parameters.

Chapter 4

Experiments

4.1 Experimental Setup

4.1.1 Model and Tokeniser

Since a tokeniser is tied to a model, both cannot be selected separately. However, only models using a Byte-Pair Encoding tokeniser are considered, given that the token addition and replacement methods are proposed for this algorithm. In the model adaptation experiments, an instance of a pre-trained model is adapted to one of the four target languages. Hence, instances of OPT (Zhang et al., 2022) are used because of the almost exclusively English pre-training dataset. Considering the available computational resources and the planned experiments, the OPT-1.3B version with roughly 1.3 billion parameters is selected. The model follows the modified GPT architecture and uses a BPE tokenisation scheme. Furthermore, the average size of its vocabulary (∼50k) places it in between the popular open source models such as Llama 2, Mistral 7B (Jiang et al., 2023) with ∼32k tokens, Llama 3[1] with ∼128k tokens, and Bloom, Gemma (Team et al., 2024) with ∼256k tokens. Moreover, OPT-1.3B also supports flash-attention (Dao et al., 2022), an efficient implementation of the attention mechanism, leveraging kernel fusion and the fast SRAM GPU memory. Detailed architectural parameters of the mode has been presented in Table 4.1.

Implementations of the tokeniser and the model come from the HuggingFace Transformers[2] Python library, which provides model source code as well as pre-trained weights. Furthermore, the library provides an efficient tokeniser implementation in Rust and uses the PyTorch framework[3] for model definition and training.

[1] `https://ai.meta.com/blog/meta-llama-3/`

[2] `https://github.com/huggingface/transformers`

[3] `https://github.com/pytorch/pytorch`

Parameter	Value
Parameter Count	1,315,758,080
Attention Heads	32
Hidden Layers	24
Vocabulary Size	50,265
Embedding Dimension	2,048
Feedforward Hidden Dimension	8,192
Context Length	2,048
Tokenisation algorithm	Byte-Pair Encoding
Pre-training Tokens	180B
Pre-training Dataset Language	English

Table 4.1: Overview of the OPT-1.3B model.

4.1.2 Tokenisation and Training Data

Tokenisation data

Full language-specific subsets of the WURA dataset are used to train target language tokenisers, leveraging its advantageous focus on African languages. Furthermore, WURA contains diversified content from online news websites and, therefore, relatively high-quality texts. Each sample in the dataset is an instance of an article split into a `headline` and its `content`. Furthermore, WURA is split into training (90%) and validation (10%) subsets, which allows for training language tokenisers on the former and their evaluation on the latter. The number of documents and the average document length of the selected language subsets of WURA are presented in Table 4.2.

Language	No. Documents	Mean and (std. dev.) doc. Length
Amharic	135,863	3,329.9 (4,953.0)
Hausa	359,881	2,343.1 (4,118.1)
Igbo	51,386	3,137.5 (5,000.5)
Yoruba	73,473	2,080.3 (5,054.8)

Table 4.2: Overview of the language subsets from WURA. Values are given for the `test` split of the dataset. Average document length is given in characters and calculated using entries from the `content` column.

The surprisingly high standard deviation in the document length indicates that the documents are of significantly varying lengths, which needs to be considered when sampling from the dataset. However, further analysis reveals that the high variance is caused by outliers since more than 95% of all documents in all considered languages are shorter than 10,000 characters. The tokeniser training and evaluations are performed using a dual-socket system of 2x AMD EPYC 75F3 with 32 cores each and 512GB of RAM.

Continuous Pre-Training Data

A language-specific dataset for continuous pre-training should be large and built from relatively long, consistent, high-quality samples. WURA dataset (Table 4.2) provides all of the features as the most extensive available African-centric dataset. Alternatives, such as Glot-500c, provide short-context samples, which are less suitable for this stage.

As mentioned in Section 3.2.3, target language samples must be mixed with the original pre-training dataset. However, some of the subsets of the original OPT pre-training corpora are no longer publicly available.[4] Hence, a dataset with a similar format and content distribution should be used. Dolma (Soldaini et al., 2024) is a recently published corpus containing cleaned English documents from sources such as CommonCrawl, Reddit, Project Gutenberg and Wikipedia. Its representative sample – Dolma v1.6-sample, contains roughly 10B tokens, reflecting the distribution of the entire corpus. Hence, it is further used as a substitute for the missing OPT pre-training dataset and mixed with the language-specific pre-training data.

No. Documents	Mean and (std. dev.) doc. Length
13,095,416	2,613.1 (7,843.3)

Table 4.3: Overview of the Dolma v1.6-sample dataset. The dataset is not split into training and validation subsets. Average document length is given in characters and calculated using entries from the `text` column.

Instruction-Tuning Data

Instruction-tuning is performed using the Aya Dataset, which contains human-crafted prompt-response pairs. Although the more extensive set of instructions, Aya Collection, is available, the former is preferred due to its higher quality. Initial analysis showed that several dataset entries have missing values, while some machine-translated samples from the Aya Collection are of low quality. However, as presented in Table 4.4, there is a significant difference in the data availability between languages, which needs to be considered when assessing adaptation efficiency. Nevertheless, the Aya Dataset contains a set of English samples, which can be mixed with the language subsets.

[4] Specifically, its significant part, *the Pile*, is no longer being hosted online.

Language	No. Documents	Mean and (std. dev.) doc. Length
English	3,944	623.8 (820.6)
Amharic	1,207	140.0 (380.8)
Hausa	3,512	375.8 (854.4)
Igbo	1,534	350.1 (534.3)
Yoruba	11,758	1,987.2 (4,768.5)

Table 4.4: Overview of the language subsets from Aya Dataset. Values are given for the `train` split of the dataset. Average document length is given in characters and calculated using combined entries from the `inputs` and `targets` columns.

4.1.3 Model Training Technology

Model training is similar for both the continuous pre-training and instruction-tuning. Training is performed in a distributed environment with the HuggingFace Accelerate[5] API, using an HPC cluster setup of four Nvidia GeForce RTX4090 GPUs, each with 24GiB of memory. To decrease the memory requirement and shorten the training time, all models are trained in a mixed precision with the `bfloat16` format, providing better optimisation stability. Further performance optimisation efforts include using the DeepSpeed framework[6] and its zero-dependency optimiser (ZeRO) Stage 3 for sharding model parameters, gradients and optimiser states in data-parallel training. Moreover, both training stages use flash-attention 2, additionally decreasing the per-iteration training time.

4.1.4 Model Training Parameters

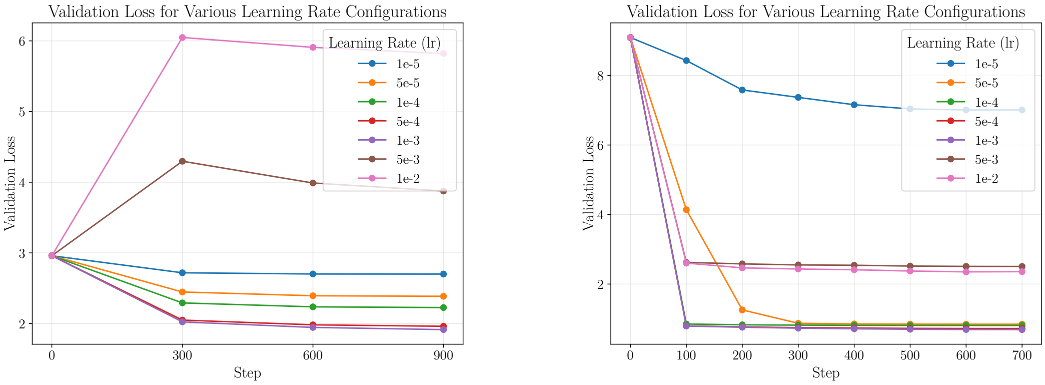
Both training stages use the original training objective – causal language modelling, and the original AdamW optimiser and linear learning rate scheduler. Neither weight decay nor dropout are used. Usually, these techniques are applied in the early stage of the pre-training to prevent overfitting. A maximum mini-batch size of 2 per device is used for continuous pre-training. To simulate an even larger mini-batch size, gradients are accumulated for 8 steps, leading to a global mini-batch size of 64. For instruction-tuning, the same mini-batch size of 2 is used, with 4 gradient accumulation steps, resulting in the global mini-batch size of 32. All models are trained for 1 epoch and up to 100M tokens, resulting in different numbers of training steps depending on the tokenisation used and the proportion of English data. The number of continuous pre-training steps ranges from 383 to 1524, while instruction-tuning ranges from 90 to 280 (except Yoruba – 933). Full details on the number of training steps are in the Appendix, Table A.2. Models are evaluated on the validation

[5] `https://github.com/huggingface/accelerate`

[6] `https://github.com/microsoft/DeepSpeed`

sets every 250 steps during the continuous pre-training and every 100 steps during instruction-tuning. LoRA targetted parameters (attention key and query parameters) are decomposed into matrices with the rank of 8 and the scaling factor $\alpha = 32$. Furthermore, a dropout of 0.1 is applied to these parameters. All models are trained with a seed of 42. Using LoRA for fine-tuning results in updating only about 11% of all learnable model parameters. A full list of model training hyperparameters is presented in Appendix A.1 in Table A.1.

The initial learning rate 1e-3 for both stages has been selected using the validation loss obtained in the experimental runs. The search has been performed over lr $\in$ {1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5}, continuous pre-training English proportions of 0% and 50%, as well as two languages – Yoruba and Hausa. A hyperparameter search has been performed for continuous pre-training and instruction-tuning using WURA, Dolma and Aya Dataset. In both cases, an optimal value 1e-3 avoided model divergence, resulting in stable optimisation. Detailed information about hyperparameter tuning is presented in Appendix A.2, Figures A.1, A.2.



(a) A mix of 75% Yoruba from WURA and 25% English from Dolma v1.6-sample.

(b) A mix of 75% Yoruba and 25% English from Aya Dataset.

Figure 4.1: Validation loss is calculated every 300 iterations of continuous pre-training and every 100 iterations of instruction-tuning stages.

4.2 Experiment Design

4.2.1 Tokeniser Adaptation

The effect of token replacement and adaptation on the tokeniser's fertility is studied initially. For each target language, a dedicated tokeniser with a vocabulary size of the original OPT BPE tokeniser (50,265) is trained. Subsequently, for each value of k, tokens from the target language tokeniser (Table 4.5) are added to the original OPT BPE tokeniser. Similarly, the same procedure is repeated for replacing k target language tokens with k tokens from the base tokeniser. Furthermore, in the case of token replacement, all final non-overlapping tokens from the base tokeniser are substituted, resulting in a different number of replaced tokens across languages. This results in 15 adapted tokenisers per the four considered African languages. To evaluate the in-language increase in fertility and the potential decrease in fertility in English texts, each instance of the modified tokeniser is used to tokenise the target language and English validation subsets of WURA. Furthermore, they are compared to the baseline OPT BPE tokeniser and tokenisers from popular models: Llama 2, Gemma and Bloom.

Tokeniser Modification	**Values of k**
Replacement	100, 500, 1000, 2000, 5000, 10000, 15000, All
Addition	100, 500, 1000, 2000, 5000, 10000, 15000

Table 4.5: Considered numbers of replaced and added tokens. Note that All refers to the all possible replacements dictated by the number of final non-overlapping tokens in the base tokeniser and depends on the target language.

4.2.2 Tokeniser Modifications and Model Performance

Using the tokeniser adaptation study results, a selection of language-specific tokenisers is used to tokenise model training datasets. In addition to the original OPT BPE tokeniser, four modified tokenisers are chosen per language. Several model training sessions are performed to measure the influence of adapted tokenisers and corresponding word embedding entries on the model's performance. Initially, each of the 5 tokenisers used per language is used to create a tokenised instance of the continuous pre-training dataset. Each dataset is sampled from the language-specific subsets of the WURA train set and is limited to a fixed size of 100M tokens. Importantly, due to the varying vocabulary size across tokenisers applied to the same dataset, a potentially different number of dataset samples was used in each case. The instruction-tuning dataset follows the same tokenisation approach. Furthermore, to study the effect of the amount of English pre-training data in the continuous pre-training dataset, three variants of the target language - English proportions are considered and presented in Table 3.2. The instruction-tuning stage uses a fixed proportion of 75% target language and 25% English data. Summarising, pre-training dataset tokenisation results in 3 proportion variants, each with four languages and four tokenisers. Additionally, each language in variant 1 is tokenised with the base tokeniser. This results in 52 instances of model training in the first stage. Similarly, the single variant of the instruction-tuning dataset is tokenised using four base tokenisers and four language-specific tokenisers for each language. This results in 20 dataset tokenisations, subsequently used to instruction-tune 52 continuously pre-trained models. The complete list of model training configurations is presented in Appendix A.1 Table A.2.

4.3 Evaluation Strategy

4.3.1 Tokenisation

There are three primary components of the tokeniser adaptation evaluation. Initially, a standard metric of the ratio of tokens per work (fertility rate) is used. Each language-adapted tokeniser is used to tokenise both WURA validation subsets in the target language and English. Given no access to the actual word count of the dataset, the number of words is estimated using the rule-based punctuation tokenisation algorithm from the Natural Language Toolkit[7] Python library. Furthermore, a fertility rate improvement per the number of modified tokens is reported to show how the fertility rate changes with the number of modified tokens. Finally, the evaluation compares the tokenisation time of the same datasets under different tokenisers to identify whether the tokeniser adaptation methods affect the performance of tokenisation. The results of this evaluation were directly used to decide which tokenisers should be used in the model adaptation stage.

4.3.2 Trained Language Models

The evaluation of adapted models is performed using downstream task benchmarks. Following the study of Ojo et al. (2024), this work measures model capabilities using an in-context learning approach with verbalised instruction prompts. Each model is evaluated on the set of tasks, corresponding datasets and metrics presented in Table 4.6. However, due to the relatively small size of the adapted LLMs, zero-shot evaluation has been changed to few-shots for all tasks. Furthermore, an empty response ratio is calculated for each of the tasks. Evaluations allow per-language model comparisons using differently adapted tokenisers and a comparison against the baseline fine-tuned on the same datasets. Moreover, models are evaluated after each training stage to investigate the effect of the instruction-tuning. Finally, validation perplexity is reported as an intrinsic evaluation metric. However, as mentioned

[7] `https://www.nltk.org`

by Yong et al. (2023), perplexity is not a great measure of LLM capabilities, particularly instruction-following skills. Example model evaluation prompts are presented in Appendix A.4. The following is a brief overview of the evaluation benchmarks:

- **AfriQA** – A cross-lingual, open-retrieval question answering dataset for African languages. The answers to asked questions are provided in the context paragraph supplied to the model.
- **MAFAND-MT** – A machine translation dataset. Sentences in the African languages selected for this study have their English translations.
- **MasakhaNew**s – A news topic classification dataset. Dataset containing a set of news articles in each language. Each of the articles should be assigned one of the up to 8 general labels such as *business* or *technology.*
- **AfriSenti** – A sentiment classification task of tweets in a set of African languages. The task involves assigning one of the *positive*, *negative* and *neutral* labels to each tweet.
- **MasakhaNER** – A dataset containing sentences in African languages and the corresponding NER tags of selected entities.

Task	**Benchmark Name**	**Metrics**
English QA	AfriQA	SQuAD F1 Score
Cross-lingual QA	AfriQA	SQuAD F1 Score
MT Target Lang to English	MAFAND-MT	ChrF^{b}
MT English to Target Lang	MAFAND-MT	ChrF^{b}
News Topic Classification	MasakhaNews	F1 Score
Sentiment Classification	AfriSenti	F1 Score
Named Entity Recognition	MasakhaNER	F1 Score

Table 4.6: A summary of used benchmarks and corresponding metrics. Note that AfriQA does not provide samples in Amharic, and hence, this language is not evaluated for QA. *a*: Character-Level F-score.

Chapter 5

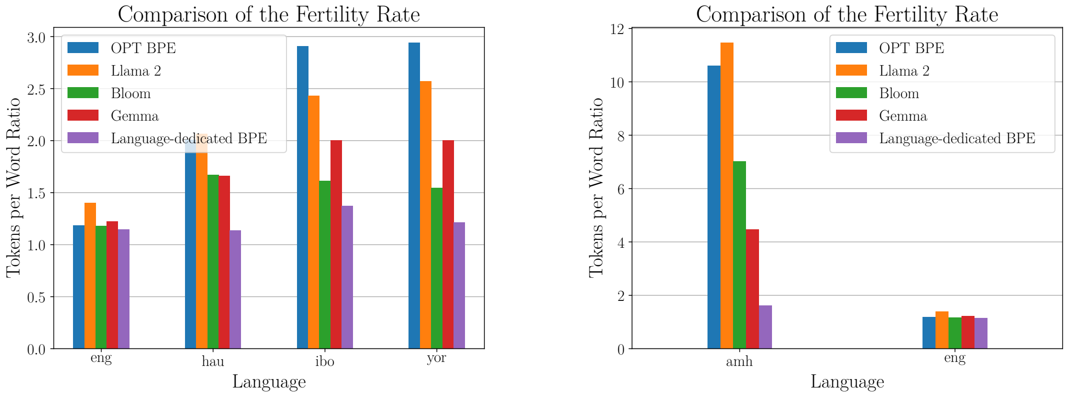
Results & Discussion

5.1 Tokeniser Fertility Analysis

5.1.1 Tokenisers of Large Language Models

To explore the landscape of the fertility rate of tokenisers from the commonly used models, they have been applied to texts in English and the selected African Languages. The results are presented in Figure 5.1, which provides interesting intuition about the training data of evaluated tokenisers. First of all, the English-dominated training sets of the tokenisers result in their optimal fertility rate between 1.0 and 1.5 on English validation data. Notably, all language-dedicated tokenisers have a fertility rate within the same range. Considering the other languages, a significant increase in the tokens per word ratio for OPT and Llama is caused by the lack of African language data in their training datasets. Interestingly, the increased vocabulary size of the Gemma tokeniser partially improves its fertility rate, although having almost exclusively an English training dataset. Nevertheless, Bloom's tokeniser uses the same number of tokens but was trained on a multilingual dataset and consistently achieves a fertility rate of around 1.5 in all languages using Latin script.

On the other hand, only the tokeniser trained specifically on Amharic texts handles tokenisation of this language well. While all tokenisers perform poorly (Figure 5.1a), the exceptionally high fertility rates achieved by OPT and Llama 2 effectively translate to character and byte-level tokenisations. Therefore, such tokens are re-used to represent multiple words, which may be reflected in their generic embeddings. On the other hand, the lower fertility rate and the increased average length of tokens in tokenised texts allow their better representation in a latent space. Hence, tokeniser adaptation may play a vital role in the LLM adaptation.



(a) African languages using Latin script. English for comparison.

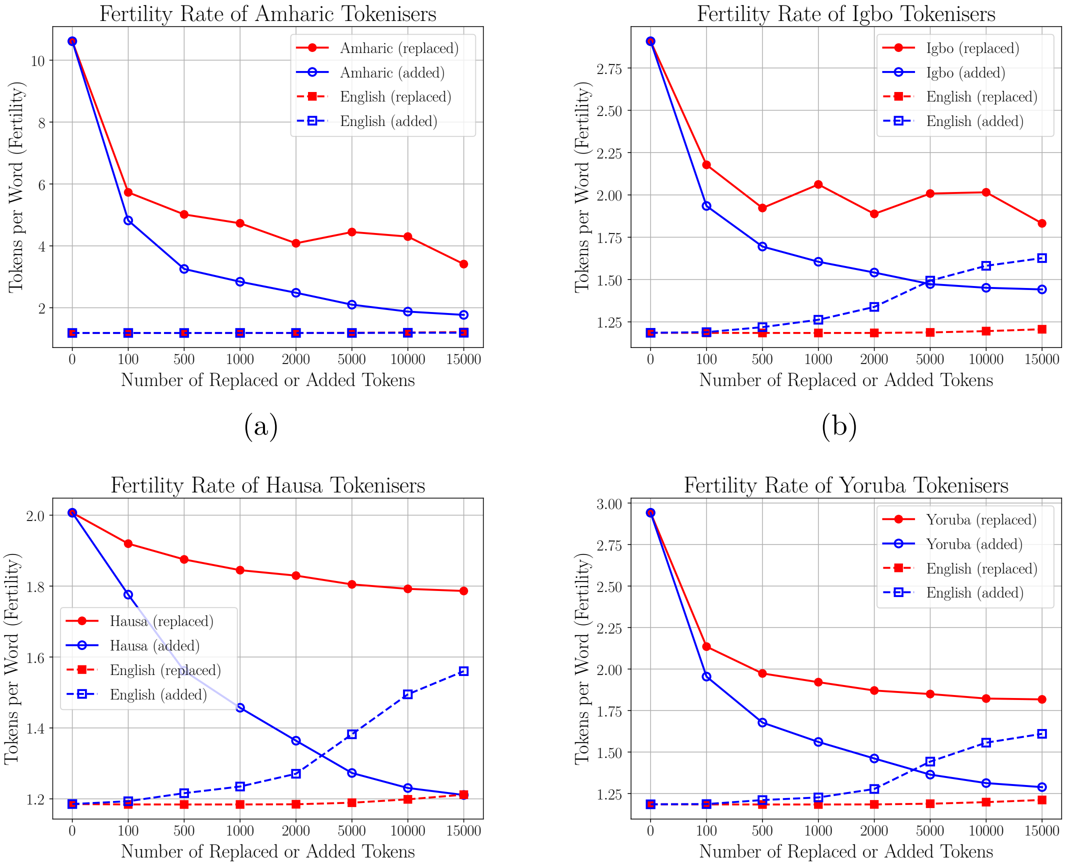
(b) Amharic uses a non-Latin script – Ge'ez. English for comparison.

Figure 5.1: Fertility rate of tokenisers of popular LLMs.

5.1.2 Token Addition and Replacement Analysis

While both tokeniser adaptation schemes were expected to lower the fertility rate of the original OPT tokeniser on the target language data, they do so with very different dynamics, as presented in Figure 5.2. First, the per-language experiments show consistent fertility rate trends across all studied languages. Interestingly, the most significant difference in the tokens per word ratio on target language data occurs after adding or replacing the first $k = 100$ tokens, regardless of the target language. On the other hand, the same tokenisers applied to the English subset of WURA do not indicate meaningful differences in fertility rates. Hence, modifying a few tokens results in significantly lower fertility on a target language data while maintaining the same performance on data in the base language. This is a promising result for further model adaptation, which leverages cross-lingual training data.

However, despite their similar performance with $k = 100$, token addition and replacement exhibit significant differences as k increases. Adding target language tokens to the base tokeniser leads to a faster fertility decrease on language-specific data, in contrast to token replacement. This trend holds true for all languages, but the gap between added and replaced tokens is particularly pronounced for Hausa (Figure 5.2c). Moreover, both approaches have distinct effects on fertility in the base language. Surprisingly, the English data fertility of tokenisers with replaced tokens remains almost unchanged, while the fertility of those with added tokens increases with growing k. The algorithmic details of Byte-Pair Encoding (Section 2.1.2) and the tokeniser adaptation methodology (Section 3.1.2) provide further insights into these unexpected behaviours. To tokenise a sequence of characters (bytes), a BPE tokeniser uses replaced tokens in a standard way, building them through the corresponding merge rules. However, because they replace the least frequent tokens, the tokeniser may already have different intermediate tokenisations of the same character (byte) sequence. Hence, depending on the context within the sequence of characters (bytes), the frequency-based tokeniser may consider the replaced tokens inferior to alternative tokenisations. On the other hand, added tokens are searched and replaced directly in the text before applying tokeniser merge rules, hence leading to a significant decrease in tokeniser fertility on target language texts. The author suspects the same mechanism to be responsible for the increasing fertility in English data. Detailed fertility rate results are available in Appendix B.1, Table B.1.



(a)

(b)

(c)

(d)

Figure 5.2: Fertility rates for base tokenisers modified by adding or replacing *k* tokens from target language tokenisers.

Furthermore, replacing and adding the same number of tokens results in different tokenisation times of language-specific subsets of WURA (Figure 5.3). Importantly, both tokeniser modification schemes result in shorter tokenisation. Intuitively, the performance improvements result from the lower tokeniser fertility and, equivalently, larger average length of tokens. Splitting the same dataset into fewer tokens requires fewer tokeniser merges and, therefore, less time. Furthermore, tokenisers with extended vocabularies are faster because they start by replacing substrings with token placeholders, hence decreasing the number of required merges.

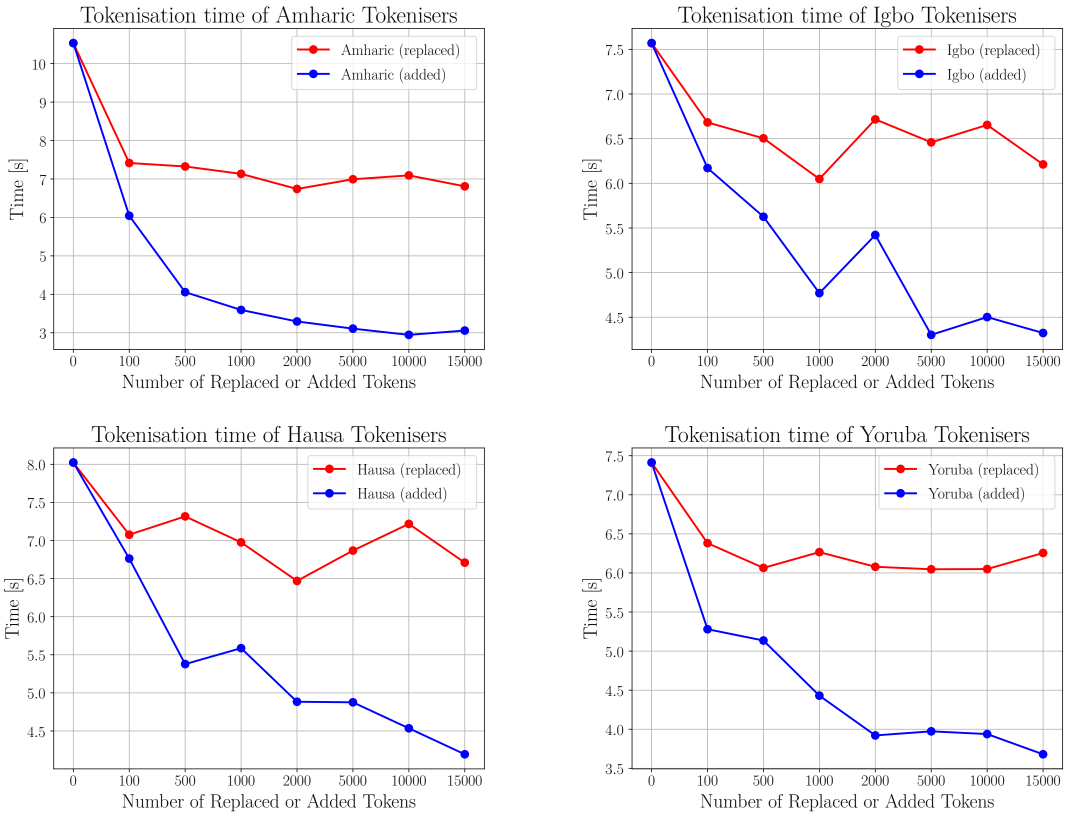


Figure 5.3: Tokenisation time for base tokenisers modified by adding or replacing k tokens from target language tokenisers. All pre-tokenisation datasets contain ∼5M words.

The obtained experimental results display the existence of a trade-off between target language fertility, base language fertility and tokeniser performance. Furthermore, modifications of tokens need to be reflected during model training through re-learning the corresponding word embeddings. While a preferred tokeniser usually has lower fertility, significantly extending the model's vocabulary or re-learning existing word embeddings may require large amounts of data. To further explore the advantages of tokeniser adaptation, Figure 5.4 presents the improvements in fertility rates per language, normalised by the number of modified tokens. The normalised increase in fertility rate decreases exponentially, and for most of the considered languages, modifying the first 100 tokens has around four times more influence on decreasing fertility than modifying the next 400 tokens.

While decreasing the tokenisation time is not the primary goal of this project, selecting tokenisers for further model adaptation experiments needs to be based on the

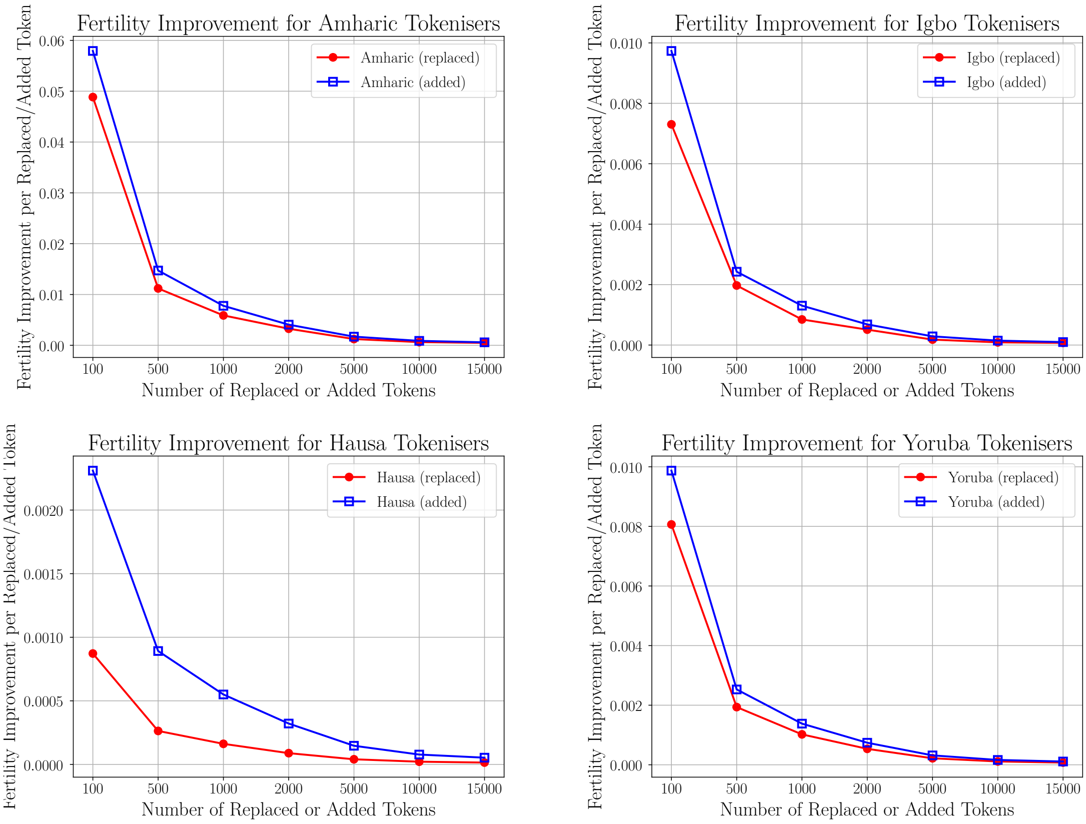
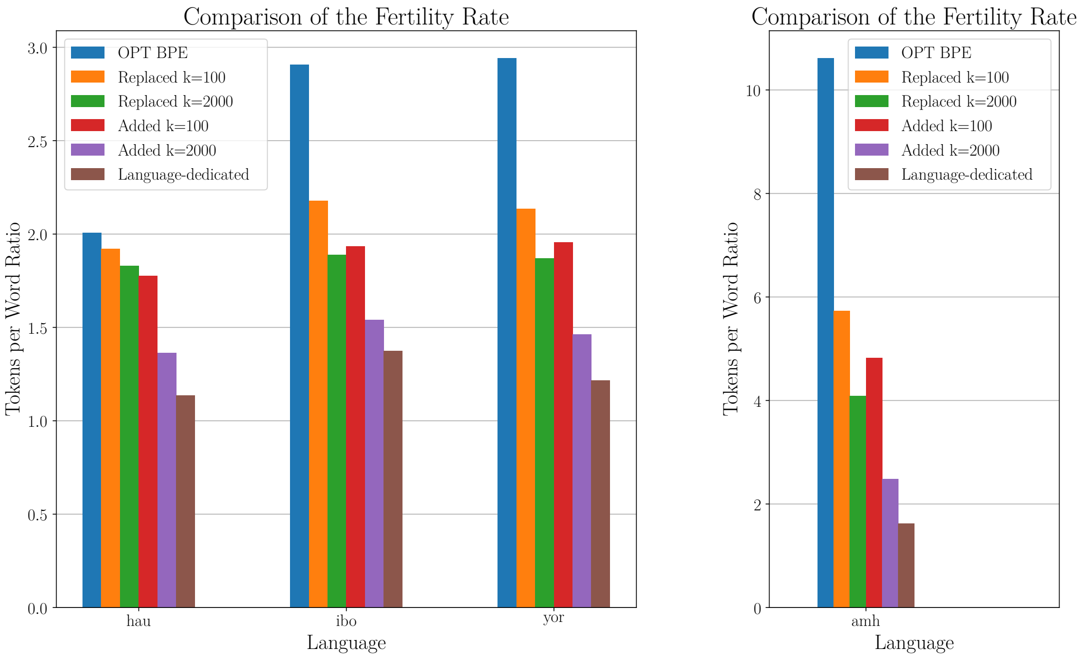


Figure 5.4: Fertility rate improvement with respect to the base tokeniser normalised by the number of modified (added or replaced) tokens.

fertility of both target and base languages and the number of changes that need to be introduced to the embedding matrix. To study the influence of replacing a small number of tokens, all tokenisers with 100 replaced and 100 added tokens are selected. Furthermore, the decrease in fertility beyond 2000 replaced tokens is marginal and, in some cases, behaves non-monotonically (Figures 5.2a and 5.2b). However, tokeniser replacement does not seem to work for Hausa very well, and replacing more tokens is required to match the fertility rate improvement achieved by tokenisers in other languages. Nevertheless, all tokenisers with 2000 replaced and 2000 added tokens are studied for consistency. Figure 5.5 shows the per-language fertility rate comparison between the base tokeniser (OPT), target language tokeniser, and selected adapted tokenisers. Notably, replacing 2000 tokens is equivalent to extending the vocabulary by only 100. Finally, tokenisers with an added 2000 tokens achieve fertility comparable to the language-dedicated tokenisers. Detailed fertility rates are presented in Table 5.1.



(a) Hausa, Igbo and Yoruba.

(b) Amharic.

Figure 5.5: Fertility rate of the base tokeniser and target language tokenisers, compared to the adapted tokenisers (replace-100, add-100, replace-2000, add-2000) selected for the further model adaptation.

Lang.	**OPT**	**R-100**	**R-2000**	**A-100**	**A-2000**	**Lang-Dedicated**
amh	10.61	5.73	4.09	4.82	2.49	1.62
hau	2.00	1.91	1.82	1.78	1.36	1.13
ibo	2.90	2.18	1.89	1.94	1.55	1.38
yor	2.95	2.13	1.87	1.96	1.46	1.22

Table 5.1: Detailed fertility rates of the tokenisers selected for the model adaptation stage. Note the preferred fertility range of 1-1.5. **R** refers to replaced tokens, while **A** indicates added tokens.

5.2 Model Adaptation Analysis

The initial experimental results indicated a near-zero performance of all models in a zero-shot in-context learning setting. Hence, all experiments and all tasks use a 3-shot ICL evaluation methodology. Although there is no single model outperforming the OPT-1.3B baseline in all tasks, almost every task-language pair has an adapted model, which beats the baseline, as shown in Figure 5.6. Furthermore, Table 5.2 presents detailed information regarding model and tokeniser adaptation details of the best performing models. Interestingly, languages differ in the improvement achieved in each task, suggesting no clear answer to Research Question 2. For example, a notable increase in sentiment classification of Amharic texts (13%) does not translate to other languages, with only a small increase for Hausa (4%) and marginal improvement for Igbo. Such an increase in Amharic may be caused by adding the vocabulary expansion and added support for the Ge'ez script. On the other hand, an improvement obtained in one task does not automatically translate to another, similar task. While there are no significant improvements in Machine Translation to English, model performance in the opposite task improves similarly for all languages. Interestingly, these asymmetric improvements may indicate the undesired effect of catastrophic forgetting, resulting in poorer capabilities of English text generation. However, this is not the case in question answering, where there is an observable improvement in both English and cross-lingual settings. This result displays the possible impact of additional instruction-tuning with samples often formatted as question-answer pairs. The results for the MasakhaNER benchmark are not presented since neither of the models has proven to be capable of Named Entity Recognition in the in-context learning format.

Model adaptations were closely monitored through inspecting the training and validation losses. Although the convergence of all models, the smallest loss values in the continuous pre-training were achieved by models adapted with the original OPT tokeniser. While the training loss curves have similar shapes, the adaptation of models with 2000 added tokens consistently resulted in significantly higher loss values. This is as expected and shows that models with more significant changes to the embedding matrix result in higher perplexity. Notably, models with replaced 2000 tokens achieved similar loss values to models were 100 tokens were added, indicating that token replacement introduces less disorder to the model functioning. On the other hand, instruction-tuning losses do not differ across different tokenisers. Training loss curves are in Appendix A.3.

Further analysis of model adaptation is split into three subsections, each studying the problem from the perspective of a different variable. Nevertheless, a common result obtained across all experiments is the inconsistent adaptation accuracy across evaluated target languages and tasks. This is particularly visible when comparing the evaluation results between tasks involving shorter and longer generations. Furthermore, the results differ depending on the adaptations to languages with the same script as the model's base language (Latin – Hausa, Igbo, Yoruba) and those with a different script (Ge'ez – Amharic).

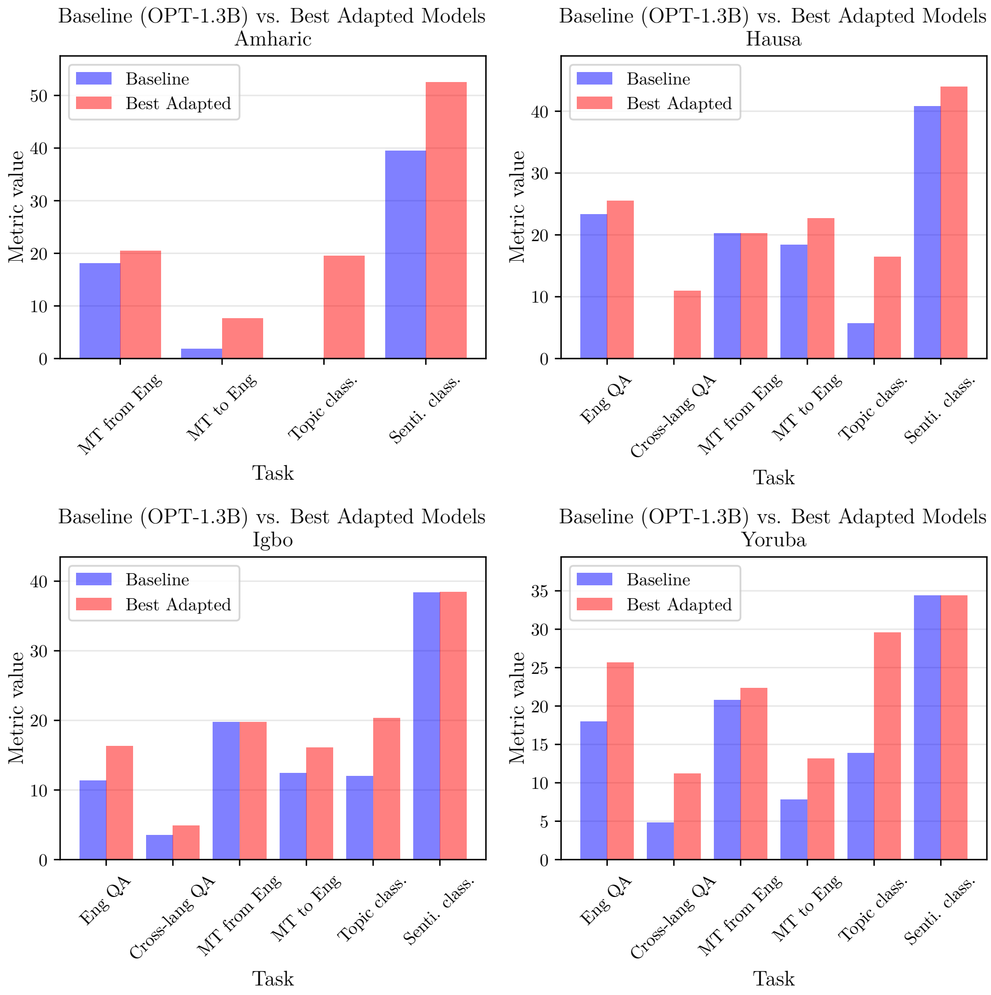
Finally, while the adapted models generally achieve better results than the baseline, the improvement is insignificant and does not compare to larger LLMs (Ojo et al., 2024). One of the suspected reasons is the nature of the selected evaluation tasks, which require cross-lingual model capabilities to reason and generate text in both English and a target language. While sentiment and topic classification tasks involve generating single-class labels in English, machine translation and question answering require predicting longer texts, possibly in the target language. Furthermore, AfriQA is a cross-lingual benchmark that utilises capabilities such as answer retrieval from a provided context paragraph. While LLMs can learn and display such behaviours, the limited scale of this study and the selection of the 1.3B base model may have contributed to the inconsistency in results. Furthermore, the use of currently unavailable monolingual benchmarks in model evaluation or the adjustment of existing cross-lingual methods, as well as increasing the number of examples provided in prompts, could benefit the quality of this study.

Task	**Amharic**	**Hausa**	**Igbo**	**Yoruba**
English QA	-	A-100_2^b	A-2000_1^b	A-100_1^b
Cross-lingual QA	-	A-2000_3^b	A-2000_2^b	R-2000_1^b
MT Target Lang to English	A-100_3^a	*Baseline*	*Baseline*	R-2000_3^a
MT English to Target Lang	A-2000_1^a	OPT-Tok_1^b	A-2000_3^a	R-2000_3^a
News Topic Classification	A-2000_2^a	A-2000_3^a	R-2000_3^b	OPT-Tok_1^b
Sentiment Classification	A-100_2^a	A-2000_3^a	R-2000_3^a	*Baseline*

Table 5.2: Best model performance in 3-shot evaluation for task-language pairs. Superscripts: a – continuous pre-training only, b – continuous pre-training and instruction-tuning. Subscripts refer to continuous pre-training dataset variants.

5.2.1 Tokeniser Adaptation Evaluation

While modifying the tokeniser generally improves the performance of model adaptation for almost all task-language pairs (Table 5.2), no clear result indicates which methods should be used. While token addition is more successful for Amharic and


Baseline (OPT-1.3B) vs. Best Adapted Models Amharic

Baseline (OPT-1.3B) vs. Best Adapted Models Hausa

Baseline (OPT-1.3B) vs. Best Adapted Models Igbo

Baseline (OPT-1.3B) vs. Best Adapted Models Yoruba

Figure 5.6: 3-shot evaluation of the baseline OP-1.3B and adapted models. Note that each *Best Adapted* refers to the best of all model for the particular task-language pair – full details are presented in Table 5.2. Hence, the performance of a *Best Adapted* on a particular task may be very different to other tasks.

Hausa, models adapted through token replacement perform better in Yoruba. The results of the experiments also indicate that the optimal choice depends on language and task. Nevertheless, task performance and the number of modified tokens are more consistent across the studied pairs. The gathered results indicate that the accuracy increases with more modified tokens, apart from machine translations to English and English question answering. Both tasks require strong English understanding capabilities. Hence, it is possible that for such tasks, a smaller vocabulary in a target language achieves better performance.

Plots in Figure 5.7 show various task performances of Amharic-adapted models. Although tokeniser adaptation leads to consistently improving machine translation, it decreases the model's ability to classify the sentiment of texts from 50% to almost 0%, indicating the model is not producing sentiment labels any more. Interestingly,

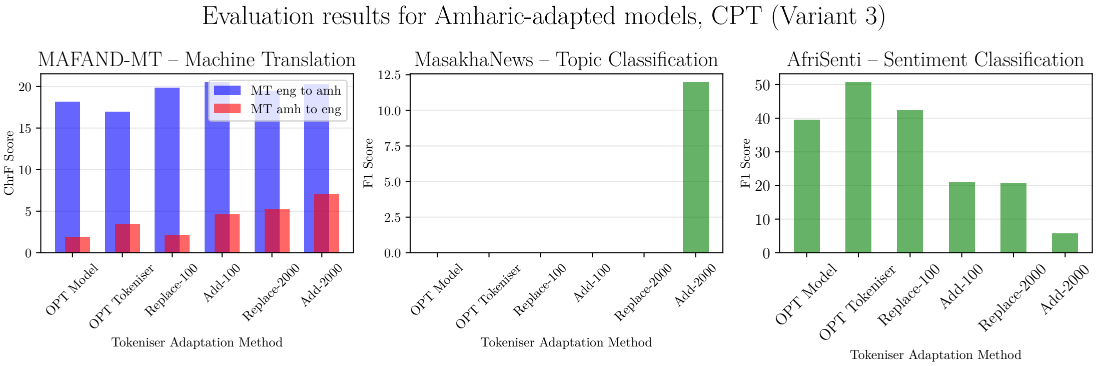


Figure 5.7: 3-shot evaluation of Amharic-adapted models, trained solely through continuous pre-training with a data mix of 50% English and 50% target language. The empty results for topic classification indicate it achieved F1 scores of 0 for all adapted models but the last *Add-2000*.

only the model with the added 2000 tokens can classify the topic of Amharic news articles, while the remaining methods always result in non-label generations. Such behaviour could result from the different scripts used for the target language, where Amharic has no direct support in the tokeniser. The surprising decrease in AfriSenti contradicts this intuition, especially given that the last two tasks do not require generations in the target language.

However, the decrease in AfriSenti and the lack of MasakhaNews classifications do not occur for Hausa (Figure 5.2c). Instead, models with more modified tokens (2000 added or replaced) achieve the best results for almost all tasks. Notably, the significant increase in F1 Score of more than 15pp on the topic classification task and more than 20pp on the sentiment classification shows a positive impact of the tokeniser adaptation compared to the model using the original tokeniser. Furthermore, models with adapted tokenisers achieve better results in the question-answering benchmark and do not improve on machine translation. On the other hand, tokeniser adaptation results in better translations for Igbo and Yoruba (Figures 5.2b and 5.2d), possibly due to different characteristics of the pre-training data, such as shorter document length. However, the choice of the tokeniser adaptation method in these languages has a significant influence on the news classification capabilities. For Igbo-adapted models, tokenisers with more modified tokens increase the F1 score more significantly. At the same time, for Yoruba-adapted models, the score almost doubles compared to the original tokeniser when 100 tokens are modified, increasing less when 2000 tokens are added or replaced. This indicates that additional factors may influence the final model performance, such as the quality and quantity of training data per language; hence, the answer to Research Question 1 requires further studies.

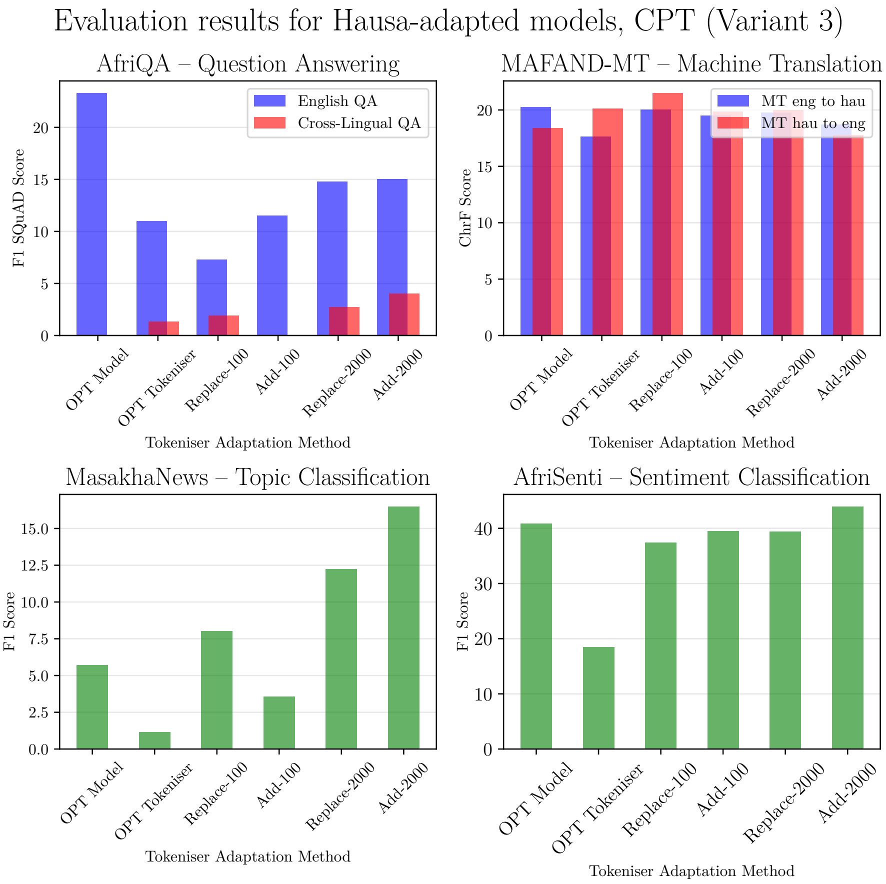
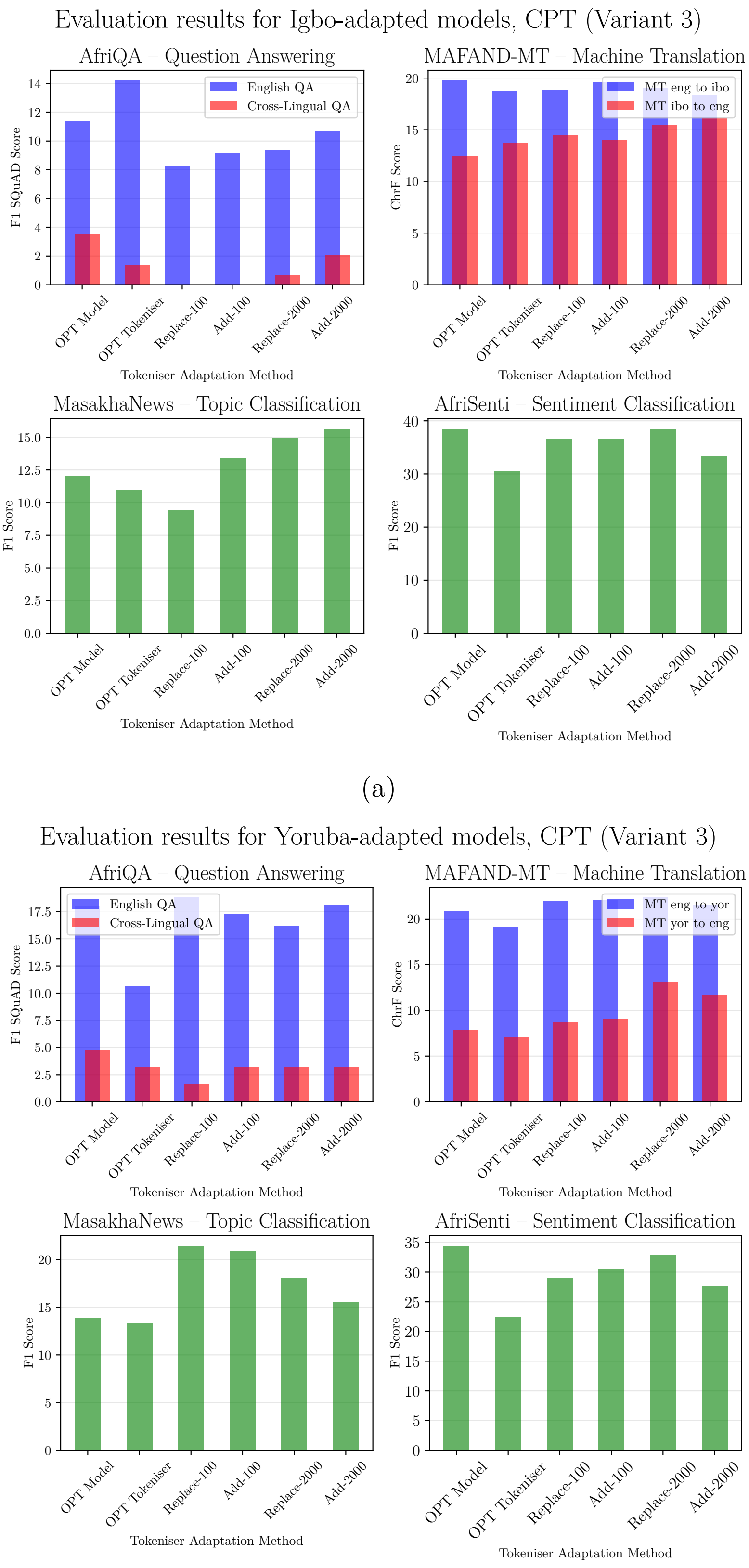


Figure 5.8: 3-shot evaluation of Hausa-adapted models, trained solely through continuous pre-training with a data mix of 50% English and 50% target language.



(a)

(b)

Figure 5.9: 3-shot evaluation of Igbo-adapted and Yoruba-adapted models, trained solely through continuous pre-training with a data mix of 50% English and 50% target language.

5.2.2 Dataset Variants

Furthermore, adding base language texts to the continuous pre-training dataset has been investigated. Similarly to the previous section, the model adaptation performance depends on the chosen task and language. As shown in 5.10, three dataset variants mentioned in Section 3.2.3 are considered. In the case of Hausa, the model's performance on most tasks benefits from introducing English texts to the dataset. Interestingly, a significant improvement across all adapted tokenisers is achieved in the sentiment classification task. Furthermore, this improvement grows with the amount of English data in the dataset (from 0% through 25% up to 50%). Similarly, the topic classification performance of models trained on a dataset containing an equal amount of English and target language data is also higher than those trained solely on target language data. This may indicate that bilingual training not only helps mitigate the effects of catastrophic forgetting but also improves the cross-lingual capabilities of the model. However, adding base language to the pre-training dataset improves machine translation to the target language but does not increase the reverse. Interestingly, English question answering does not differ significantly across variants, which, similarly to machine translation, is counter-intuitive, given that the presence of English data should enhance inference in this language.

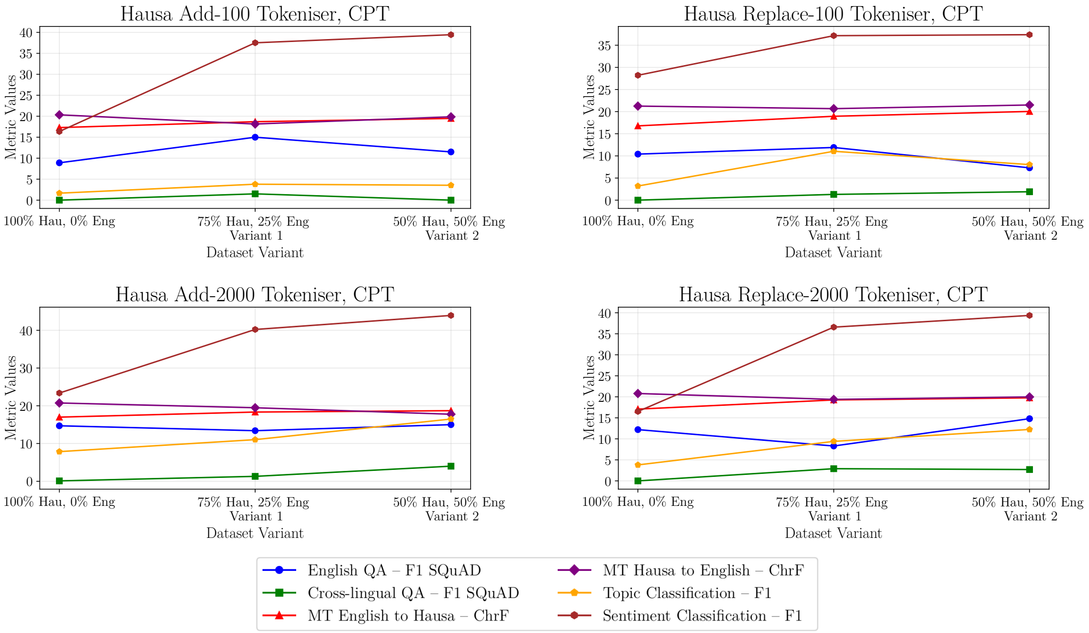


Figure 5.10: 3-shot evaluation of Hausa-adapted models, trained solely through continuous pre-training on different dataset variants.

Nevertheless, the same reasoning cannot be applied to Yoruba-adapted models (Figure 5.11). Adding English data to the continuous pre-training dataset no longer benefits the previously increased sentiment classification performance. Furthermore, a comparative analysis of different target languages suggests that the model performance is also generally unrelated to the tokeniser adaptation scheme. Topic classification is an exception to this conclusion, maintaining its improvement across all studied tokeniser adaptations.

The obtained experimental results are inconclusive in whether mixing target language data with the base language of a model contributes to increased model performance. A possible reason for the inconsistent results is the used English samples, which, although from a similar distribution, are not sampled from the actual pre-training dataset of OPT-1.3B. Nevertheless, benchmark evaluations do not show any significant decrease, suggesting that training on bilingual datasets does not worsen the model's performance. Additionally, studying Table 5.2, which shows the best models for each task-language pair, most best-performing models used variants 2 or 3. Hence, further study is necessary to confirm the effect of bilingual fine-tuning.

Additional plots displaying per-task performance for Amharic and Igbo have been attached in the Appendix B.2 as Figures B.9 and B.11, respectively.

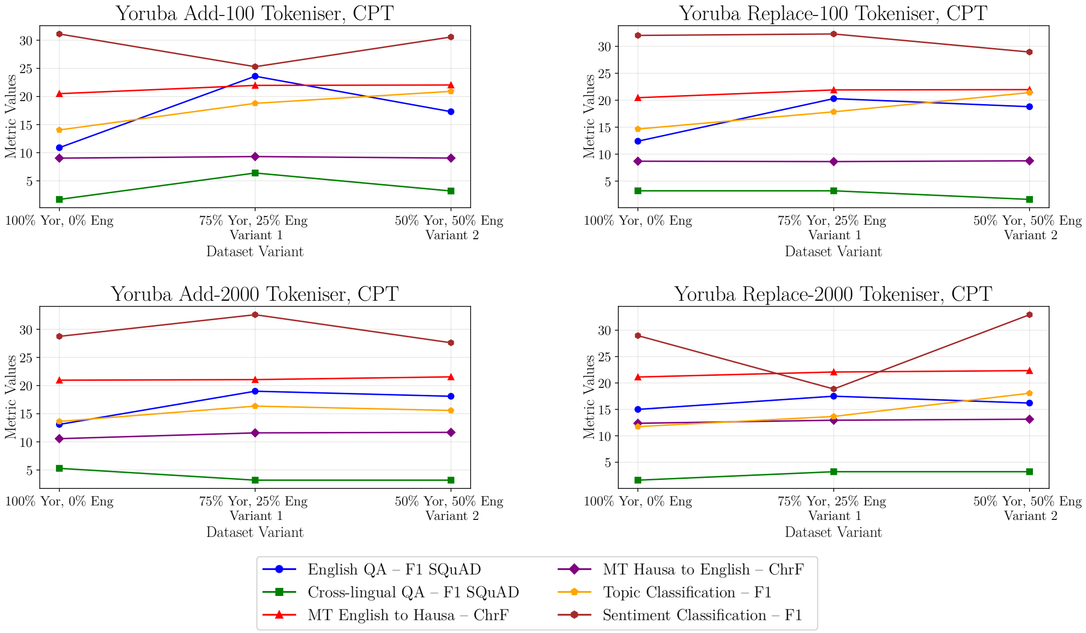


Figure 5.11: 3-shot evaluation of Yoruba-adapted models, trained solely through continuous pre-training on different dataset variants.

5.2.3 Continuous Pre-Training and Instruction Tuning

Finally, models adapted through continuous pre-training (CPT) have been further instruction-tuned (IT). Given the lack of apparent negative influence of using mixed datasets on the final model performance, variant 3 (50% English, 50% target language) has been selected for further instruction-tuning. Furthermore, 25% of the prompt-response dataset itself constitutes English samples. Instruction-tuning benefits and disadvantages around the same number of models from Table 5.2. Interestingly, there is a set of tasks, for which it seems advantageous – all the best models for question answering were instruction-tuned. This is intuitive, given the prompt-response format of the Aya Dataset used in this stage, which fits the question-answering task. On the other hand, neither of the preferred models for Amharic uses instruction-tuning. This is likely due to the poor data quality and availability– Amharic has the shortest average length and least samples in the Aya Dataset (Table 4.4).

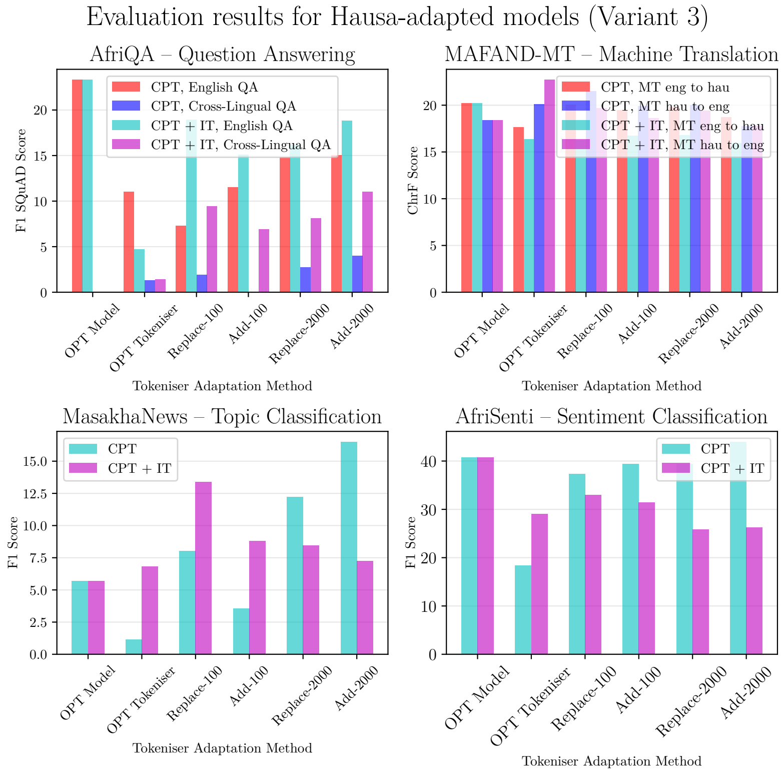
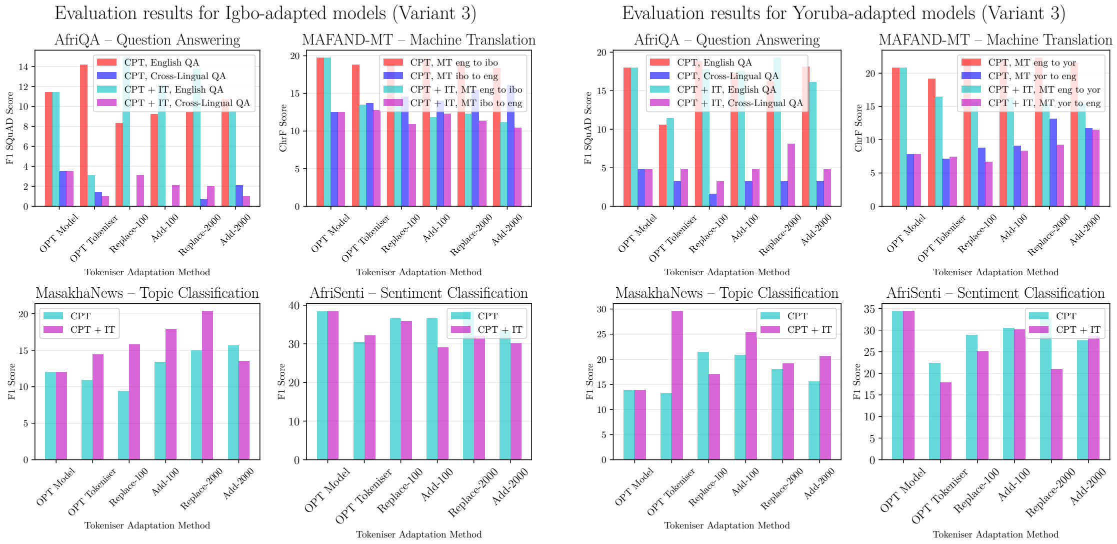


Figure 5.12: 3-shot evaluation of Hausa-adapted models. CPT refers to continuous pre-training on 50% English and 50% target language task-agnostic data. IT means instruction-tuning on 25% English and 75% target language prompt-response formatted samples.

Studying the per-language influence of instruction-tuning, Hausa displays significant and consistent improvements in question answering, where a combination of both training stages improves the quality of model generations. Interestingly, applying additional instruction-tuning to models with different numbers of modified tokens results in different results. While those with replaced or added 100 tokens benefit from training on prompt-response formatted data, models with modified 2000 tokens achieve much better performance without it. Similarly, while all models with adapted tokenisers are better in sentiment classification after continuous pre-training only, those with more significant changes to the vocabulary get worse when instruction-tuned.



(a) (b)

Figure 5.13: 3-shot evaluation of Igbo-adapted and Yoruba-adapted models. CPT refers to continuous pre-training on 50% English and 50% target language task-agnostic data. IT means instruction-tuning on 25% English and 75% target language prompt-response formatted samples.

For both Igbo (Figure 5.13a) and Yoruba (Figure 5.13b), instruction-tuning consistently worsens the performance of machine translation. Since it happens in both translation directions, it may be caused by the structure of the machine translation task. Moreover, while English question-answering capabilities change inconsistently, the same task in the target language generally benefits from instruction-tuning. Furthermore, topic and sentiment classification results do not present results consistent across languages and even not within tasks, with instruction-tuning sometimes increasing and sometimes decreasing the performance of models. Additionally, there does not seem to be a strong relation between the number of training steps (number of prompt-response samples) and performance improvement. Although instruction-tuned with a dataset roughly eight times larger than those of Amharic and Igbo, Yoruba-adapted models do not display better performance.

In summary, while instruction-tuning improves cross-lingual question answering, the improvements on other tasks are generally not consistent. Hence, this study does not produce a definite answer to Research Question 3, indicating the need for further research on the adequacy of language adaptation through instruction-tuning.

Chapter 6

Future Work and Conclusions

This project investigated methods of adapting Large Language Models to low-resource African languages. Although the availability of linguistic resources is particularly limited, following the correct adaptation methodology yielded improvements in model performance. A thorough understanding of the tokenisation algorithm and an overview of the recently published literature allowed modifications to the tokeniser vocabulary through token addition and replacement. Furthermore, applying such tokeniser adjustments to a set of African languages – Amharic, Hausa, Igbo and Yoruba, resulted in a comprehensive study of the tokeniser fertility rate. After discovering that some commonly used tokenisers do not result in satisfactory tokens per word ratio, they have been adapted by gradually replacing or adding target language tokens. While for all of the studied languages, adding only a small number of tokens significantly decreases the fertility rate, replacing the same number of tokens decreases the ratio less. Furthermore, the fertility rate of tokenisers has been discovered to decrease exponentially with the number of modified tokens, which suggests that replacing more tokens does not necessarily result in better performance of adapted models. Moreover, the results gathered during the Byte-Pair Encoding tokeniser adaptation study indicate that adding tokens to the tokeniser vocabulary may decrease the tokenisation time and negatively influence its fertility rate in the original language. The study of BPE tokeniser fertility could be extended and further applied to commonly used open-source language models and other common tokenisation schemes such as SentencePiece, leading to potential improvements in their performance as well as monolingual and multilingual adaptations.

Following the study of tokenisation schemes, several experiments have explored model adaptation. Although the adapted models achieve better performance than the baseline model in a selection of evaluated downstream tasks and languages, some of the experiments are inconclusive. There is no clear evidence on which tokeniser adaptation methods achieve better downstream task performance. Although following token replacement does not introduce additional learnable parameters to a model, further studies are necessary to verify whether it is beneficial. Likewise, further research is needed to determine whether mixing target language data with English texts improves model adaptation. While gathered evaluation results indicate no decrease in model performance when training on additional English content, the potential improvements depend on the language and the task. Finally, an additional model adaptation step of instruction-tuning, while beneficial to question answering, may result in decreased model capabilities in machine translation. Although the inconsistencies, the author hypothesises that increasing the size of prompt-response datasets or following more advanced tuning strategies may lead to better results. Finally, many of the available evaluation benchmarks rely on additional cross-lingual capabilities of tested models. Although initial modifications were made to the evaluation strategy, the choice of such benchmarks could have been a mistake, contributing to the inconclusiveness of the studied research questions. While the scale of this study did not allow adaptations of bigger language models, they could likely take advantage of the emerging behaviours, leading to more consistent results. Hence, there is a need for further studies of LLM adaptations for in-context learning tasks, given their emergence as one of the key methods in low-resource natural language processing.

Bibliography

References

David Adelani, Jesujoba Alabi, Angela Fan, Julia Kreutzer, Xiaoyu Shen, Machel Reid, Dana Ruiter, Dietrich Klakow, Peter Nabende, Ernie Chang, Tajuddeen Gwadabe, Freshia Sackey, Bonaventure F. P. Dossou, Chris Emezue, Colin Leong, Michael Beukman, Shamsuddeen Muhammad, Guyo Jarso, Oreen Yousuf, Andre Niyongabo Rubungo, Gilles Hacheme, Eric Peter Wairagala, Muhammad Umair Nasir, Benjamin Ajibade, Tunde Ajayi, Yvonne Gitau, Jade Abbott, Mohamed Ahmed, Millicent Ochieng, Anuoluwapo Aremu, Perez Ogayo, Jonathan Mukiibi, Fatoumata Ouoba Kabore, Godson Kalipe, Derguene Mbaye, Allahsera Auguste Tapo, Victoire Memdjokam Koagne, Edwin Munkoh-Buabeng, Valencia Wagner, Idris Abdulmumin, Ayodele Awokoya, Happy Buzaaba, Blessing Sibanda, Andiswa Bukula, and Sam Manthalu. 2022a. A few thousand translations go a long way! leveraging pre-trained models for African news translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3053–3070, Seattle, United States. Association for Computational Linguistics.

David Adelani, Graham Neubig, Sebastian Ruder, Shruti Rijhwani, Michael Beukman, Chester Palen-Michel, Constantine Lignos, Jesujoba Alabi, Shamsuddeen Muhammad, Peter Nabende, Cheikh M. Bamba Dione, Andiswa Bukula, Rooweither Mabuya, Bonaventure F. P. Dossou, Blessing Sibanda, Happy Buzaaba, Jonathan Mukiibi, Godson Kalipe, Derguene Mbaye, Amelia Taylor, Fatoumata Kabore, Chris Chinenye Emezue, Anuoluwapo Aremu, Perez Ogayo, Catherine Gitau, Edwin Munkoh-Buabeng, Victoire Memdjokam Koagne, Allahsera Auguste Tapo, Tebogo Macucwa, Vukosi Marivate, Mboning Tchiaze Elvis, Tajuddeen Gwadabe, Tosin Adewumi, Orevaoghene Ahia, Joyce Nakatumba-Nabende, Neo Lerato Mokono, Ignatius Ezeani, Chiamaka Chukwuneke, Mofetoluwa Oluwaseun Adeyemi, Gilles Quentin Hacheme, Idris Abdulmumin, Odunayo Ogundepo, Oreen Yousuf, Tatiana Moteu, and Dietrich Klakow. 2022b. MasakhaNER 2.0: Africa-centric transfer learning for named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4488–4508, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

David Ifeoluwa Adelani, Hannah Liu, Xiaoyu Shen, Nikita Vassilyev, Jesujoba O. Alabi, Yanke Mao, Haonan Gao, and Annie En-Shiun Lee. 2024. Sib-200: A simple, inclusive, and big evaluation dataset for topic classification in 200+ languages and dialects. *Preprint*, arXiv:2309.07445.

David Ifeoluwa Adelani, Marek Masiak, Israel Abebe Azime, Jesujoba Alabi, Atnafu Lambebo Tonja, Christine Mwase, Odunayo Ogundepo, Bonaventure F. P. Dossou, Akintunde Oladipo, Doreen Nixdorf, Chris Chinenye Emezue, Sana Alazzawi, Blessing Sibanda, Davis David, Lolwethu Ndolela, Jonathan Mukiibi, Tunde Ajayi, Tatiana Moteu, Brian Odhiambo, Abraham Owodunni, Nnaemeka Obiefuna, Muhidin Mohamed, Shamsuddeen Hassan Muhammad, Teshome Mulugeta Ababu, Saheed Abdullahi Salahudeen, Mesay Gemeda Yigezu, Tajuddeen Gwadabe, Idris Abdulmumin, Mahlet Taye, Oluwabusayo Awoyomi, Iyanuoluwa Shode, Tolulope Adelani, Habiba Abdulganiyu, Abdul-Hakeem Omotayo, Adetola Adeeko, Abeeb Afolabi, Anuoluwapo Aremu, Olanrewaju Samuel, Clemencia Siro, Wangari Kimotho, Onyekachi Ogbu, Chinedu Mbonu, Chiamaka Chukwuneke, Samuel Fanijo, Jessica Ojo, Oyinkansola Awosan, Tadesse Kebede, Toadoum Sari Sakayo, Pamela Nyatsine, Freedmore Sidume, Oreen Yousuf, Mardiyyah Oduwole, Kanda Tshinu, Ussen Kimanuka, Thina Diko, Siyanda Nxakama, Sinodos Nigusse, Abdulmejid Johar, Shafie Mohamed, Fuad Mire Hassan, Moges Ahmed Mehamed, Evrard Ngabire, Jules Jules, Ivan Ssenkungu, and Pontus Stenetorp. 2023. MasakhaNEWS: News topic classification for African languages. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 144–159, Nusa Dua, Bali. Association for Computational Linguistics.

Alham Fikri Aji, Nikolay Bogoychev, Kenneth Heafield, and Rico Sennrich. 2020. In neural machine translation, what does transfer learning transfer? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7701–7710, Online. Association for Computational Linguistics.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin

Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tyler A. Chang, Catherine Arnett, Zhuowen Tu, and Benjamin K. Bergen. 2023. When is multilinguality a curse? language modeling for 250 high- and low-resource languages. *Preprint*, arXiv:2311.09205.

Yihong Chen, Kelly Marchisio, Roberta Raileanu, David Ifeoluwa Adelani, Pontus Stenetorp, Sebastian Riedel, and Mikel Artetxe. 2023. Improving language plasticity via pretraining with active forgetting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. *Preprint*, arXiv:1706.03741.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Zoltan Csaki, Bo Li, Jonathan Li, Qiantong Xu, Pian Pawakapan, Leon Zhang, Yun Du, Hengyu Zhao, Changran Hu, and Urmish Thakker. 2024. Sambalingo: Teaching large language models new languages. *Preprint*, arXiv:2404.05829.

Zoltan Csaki, Pian Pawakapan, Urmish Thakker, and Qiantong Xu. 2023. Efficiently adapting pretrained language models to new languages. *Preprint*, arXiv:2311.05741.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Preprint*, arXiv:2205.14135.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Olmo: Accelerating the science of language models. *Preprint*, arXiv:2402.00838.

John Hewitt. 2021. Initializing new word embeddings for pretrained language models.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Ayyoob ImaniGooghari, Peiqin Lin, Amir Hossein Kargaran, Silvia Severini, Masoud Jalili Sabet, Nora Kassner, Chunlan Ma, Helmut Schmid, André Martins, François Yvon, and Hinrich Schütze. 2023. Glot500: Scaling multilingual corpora and language models to 500 languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Tom Kocmi and Ondřej Bojar. 2017. An exploration of word embedding initialization in deep-learning tasks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 56–64, Kolkata, India. NLP Association of India.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*, abs/2005.11401.

Zihao Li, Yucheng Shi, Zirui Liu, Fan Yang, Ninghao Liu, and Mengnan Du. 2024. Quantifying multilingual performance of large language models across languages. *Preprint*, arXiv:2404.11553.

Peiqin Lin, Shaoxiong Ji, Jörg Tiedemann, André F. T. Martins, and Hinrich Schütze. 2024. Mala-500: Massive language adaptation of large language models. *Preprint*, arXiv:2401.13303.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Preprint*, arXiv:2205.05638.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *Preprint*, arXiv:2202.12837.

Shamsuddeen Hassan Muhammad, Idris Abdulmumin, Abinew Ali Ayele, Nedjma Ousidhoum, David Ifeoluwa Adelani, Seid Muhie Yimam, Ibrahim Sa'id Ahmad, Meriem Beloucif, Saif M. Mohammad, Sebastian Ruder, Oumaima Hourrane, Pavel Brazdil, Felermino Dário Mário António Ali, Davis David, Salomey Osei, Bello Shehu Bello, Falalu Ibrahim, Tajuddeen Gwadabe, Samuel Rutunda, Tadesse Belay, Wendimu Baye Messelle, Hailu Beshada Balcha, Sisay Adugna Chala, Hagos Tesfahun Gebremichael, Bernard Opoku, and Steven Arthur. 2023. Afrisenti: A twitter sentiment analysis benchmark for african languages. *Preprint*, arXiv:2302.08956.

Odunayo Ogundepo, Tajuddeen R. Gwadabe, Clara E. Rivera, Jonathan H. Clark, Sebastian Ruder, David Ifeoluwa Adelani, Bonaventure F. P. Dossou, Abdou Aziz DIOP, Claytone Sikasote, Gilles Hacheme, Happy Buzaaba, Ignatius Ezeani, Rooweither Mabuya, Salomey Osei, Chris Emezue, Albert Njoroge Kahira, Shamsuddeen H. Muhammad, Akintunde Oladipo, Abraham Toluwase Owodunni, Atnafu Lambebo Tonja, Iyanuoluwa Shode, Akari Asai, Tunde Oluwaseyi Ajayi, Clemencia Siro, Steven Arthur, Mofetoluwa Adeyemi, Orevaoghene Ahia, Anuoluwapo Aremu, Oyinkansola Awosan, Chiamaka Chukwuneke, Bernard Opoku, Awokoya Ayodele, Verrah Otiende, Christine Mwase, Boyd Sinkala, Andre Niyongabo Rubungo, Daniel A. Ajisafe, Emeka Felix Onwuegbuzia, Habib Mbow, Emile Niyomutabazi, Eunice Mukonde, Falalu Ibrahim Lawan, Ibrahim Said Ahmad, Jesujoba O. Alabi, Martin Namukombo, Mbonu Chinedu, Mofya Phiri, Neo Putini, Ndumiso Mngoma, Priscilla A. Amuok, Ruqayya Nasir Iro, and Sonia Adhiambo. 2023. Afriqa:

Cross-lingual open-retrieval question answering for african languages. *Preprint*, arXiv:2305.06897.

Jessica Ojo, Kelechi Ogueji, Pontus Stenetorp, and David Ifeoluwa Adelani. 2024. How good are large language models on african languages? *Preprint*, arXiv:2311.07978.

Akintunde Oladipo, Mofetoluwa Adeyemi, Orevaoghene Ahia, Abraham Owodunni, Odunayo Ogundepo, David Adelani, and Jimmy Lin. 2023. Better quality pre-training data and t5 models for African languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 158–168, Singapore. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. *Preprint*, arXiv:2012.15613.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *Preprint*, arXiv:1508.07909.

C. E. Shannon. 1951. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30(1):50–64.

Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F. Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Souza Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergün, Ifeoma Okoh, Aisha Alaagib, Oshan Mudannayake, Zaid Alyafeai, Vu Minh Chien, Sebastian Ruder, Surya Guthikonda, Emad A. Alghamdi, Sebastian Gehrmann, Niklas Muennighoff, Max Bartolo, Julia Kreutzer, Ahmet Üstün, Marzieh Fadaee, and Sara Hooker. 2024. Aya dataset: An open-access collection for multilingual instruction tuning. *Preprint*, arXiv:2402.06619.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Jiayi Wang, David Ifeoluwa Adelani, Sweta Agrawal, Marek Masiak, Ricardo Rei, Eleftheria Briakou, Marine Carpuat, Xuanli He, Sofia Bourhim, Andiswa Bukula, Muhidin Mohamed, Temitayo Olatoye, Tosin Adewumi, Hamam Mokayed, Christine Mwase, Wangui Kimotho, Foutse Yuehgoh, Anuoluwapo Aremu, Jessica Ojo, Shamsuddeen Hassan Muhammad, Salomey Osei, Abdul-Hakeem Omotayo, Chiamaka Chukwuneke, Perez Ogayo, Oumaima Hourrane, Salma El Anigri, Lolwethu Ndolela, Thabiso Mangwana, Shafie Abdi Mohamed, Ayinde Hassan, Oluwabusayo Olufunke Awoyomi, Lama Alkhaled, Sana Al-Azzawi, Naome A. Etori, Millicent Ochieng, Clemencia Siro, Samuel Njoroge, Eric Muchiri, Wangari Kimotho, Lyse Naomi Wamba Momo, Daud Abolade, Simbiat Ajao, Iyanuoluwa Shode, Ricky Macharm, Ruqayya Nasir Iro, Saheed S. Abdullahi, Stephen E. Moore, Bernard Opoku, Zainab Akinjobi, Abeeb Afolabi, Nnaemeka Obiefuna, Onyekachi Raphael Ogbu, Sam Brian, Verrah Akinyi Otiende, Chinedu Emmanuel Mbonu, Sakayo Toadoum Sari, Yao Lu, and Pontus Stenetorp. 2024. Afrimte and africomet: Enhancing comet to embrace under-resourced african languages. *Preprint*, arXiv:2311.09828.

BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim,

Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes,

Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2023. Bloom: A 176b-parameter open-access multilingual language model. *Preprint*, arXiv:2211.05100.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zheng Xin Yong, Hailey Schoelkopf, Niklas Muennighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, Genta Winata, Stella Biderman, Edward Raff, Dragomir Radev, and Vassilina Nikoulina. 2023. BLOOM+1: Adding language support to BLOOM for zero-shot prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11682–11703, Toronto, Canada. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Judit Ács. 2019. Exploring bert's vocabulary.

Appendix A

Model Training & Evaluation Details

A.1 Model Training Summary

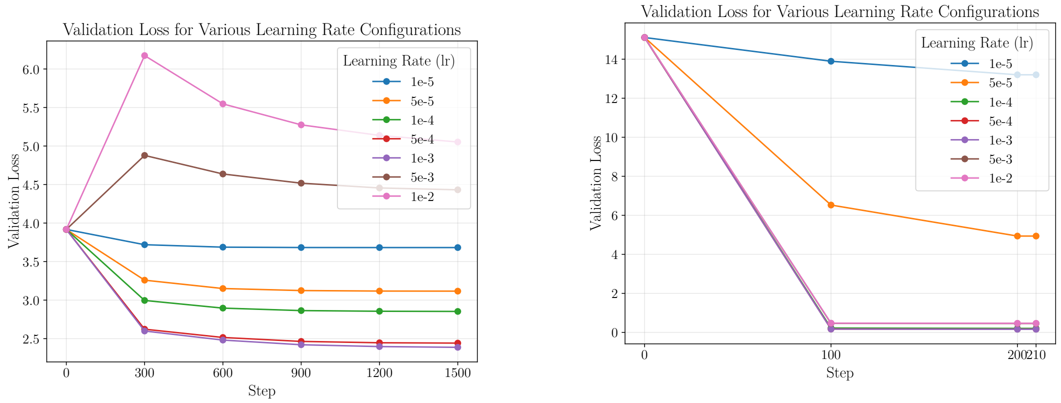
Parameter	**Cont. Pre-Training**	**Instruction-Tuning**
Optimiser	AdamW	AdamW
Learning Rate	1e-3	1e-3
Learning Rate Scheduler	Linear	Linear
Weight Decay	0	0
Dropout	0	0
Seed	42	42
LoRA Dropout	0.1	0.1
LoRA Rank r	8	8
LoRA Scaling Factor α	32	32
Per Device Mini-Batch Size	2	2
Gradient Accumulation Steps	8	4
Global Mini-Batch Size	64	32
Training Epochs	1	1

Table A.1: Overview of the model training parameters.

WURA Eng. Prop.	Lang.	Tokeniser	No. Modified Tokens	WURA Steps	Aya Steps
0%	amh	base	0	762	72
0%	hau	base	0	762	210
0%	ibo	base	0	762	90
0%	yor	base	0	762	700
0%	amh	add	100	762	96
0%	amh	add	2000	762	96
0%	amh	replace	100	762	96
0%	amh	replace	2000	762	96
0%	hau	add	100	761	280
0%	hau	add	2000	761	280
0%	hau	replace	100	762	280
0%	hau	replace	2000	761	280
0%	ibo	add	100	513	120
0%	ibo	add	2000	405	120
0%	ibo	replace	100	580	120
0%	ibo	replace	2000	500	120
0%	yor	add	100	512	933
0%	yor	add	2000	383	933
0%	yor	replace	100	559	933
0%	yor	replace	2000	489	933
25%	amh	add	100	101	96
25%	amh	add	2000	101	96
25%	amh	replace	100	101	96
25%	amh	replace	2000	101	96
25%	hau	add	100	1015	280
25%	hau	add	2000	1015	280
25%	hau	replace	100	1016	280
25%	hau	replace	2000	1015	280
25%	ibo	add	100	684	120
25%	ibo	add	2000	539	120
25%	ibo	replace	100	773	120
25%	ibo	replace	2000	667	120
25%	yor	add	100	683	933
25%	yor	add	2000	511	933
25%	yor	replace	100	745	933
25%	yor	replace	2000	652	933
50%	amh	add	100	152	96
50%	amh	add	2000	152	96
50%	amh	replace	100	152	96
50%	amh	replace	2000	152	96
50%	hau	add	100	1522	280
50%	hau	add	2000	1522	280
50%	hau	replace	100	1524	280
50%	hau	replace	2000	1522	280
50%	ibo	add	100	1025	120
50%	ibo	add	2000	809	120
50%	ibo	replace	100	1159	120
50%	ibo	replace	2000	1000	120
50%	yor	add	100	1024	933
50%	yor	add	2000	766	933
50%	yor	replace	100	1117	933
50%	yor	replace	2000	978	933

Table A.2: Listing of model trainings, corresponding datasets and tokenisers used. WURA represents the continuous pre-training stage, while Aya represents the instruction-tuning stage. A constant proportion of 25% English data was used in all instruction-tuning datasets.

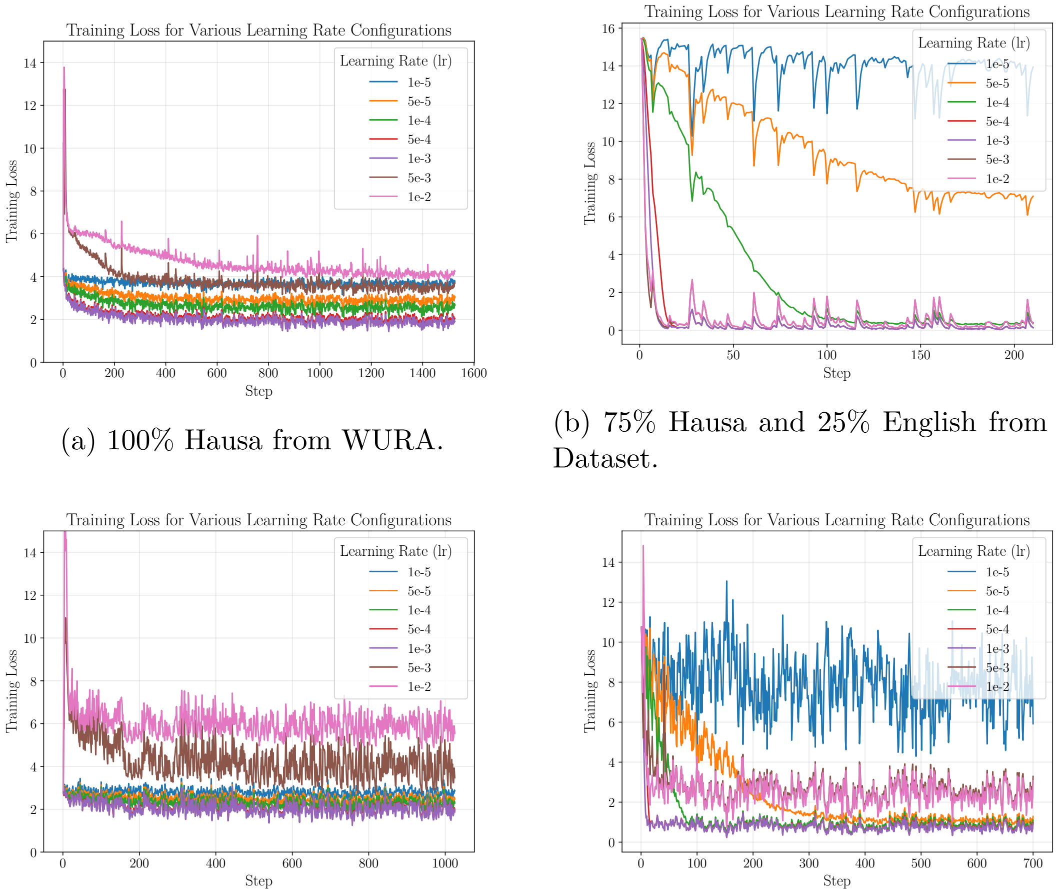
A.2 Hyperparameter Tuning



(a) 100% Hausa from WURA.

(b) 75% Hausa and 25% English from Aya Dataset.

Figure A.1: Summary of validation losses. Validation loss is calculated every 300 iterations of continuous pre-training and every 100 iterations of instruction-tuning stages. Note the low number of instruction-tuning steps is caused by the low availability of instruction-tuning data for Hausa.



(a) 100% Hausa from WURA.

(b) 75% Hausa and 25% English from Aya Dataset.

(c) 75% Yoruba from WURA and 25% English from Dolma v1.6-sample.

(d) 75% Yoruba and 25% English from Aya Dataset.

Figure A.2: Summary of training losses achieved in the hyperparameter tuning. Training loss is calculated every iteration.

A.3 Training Losses

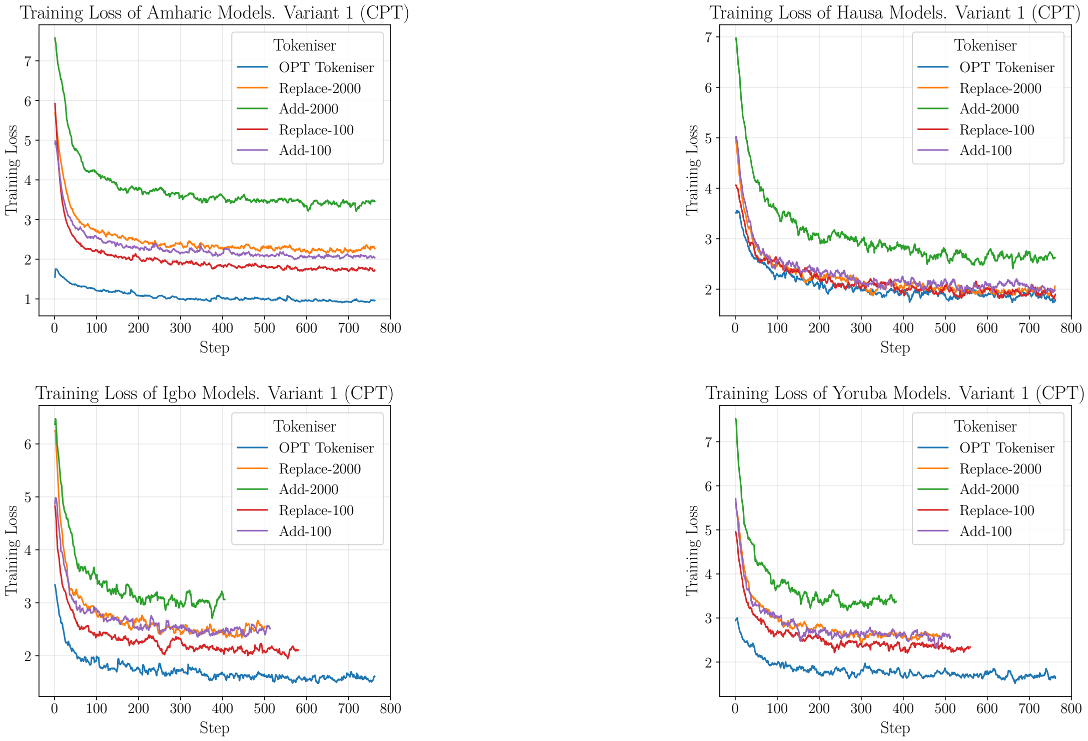


Figure A.3: Training losses of models adapted on Variant 1, CPT only. Training curves are smoothed using Exponential Moving Average with $\alpha = 0.1$.

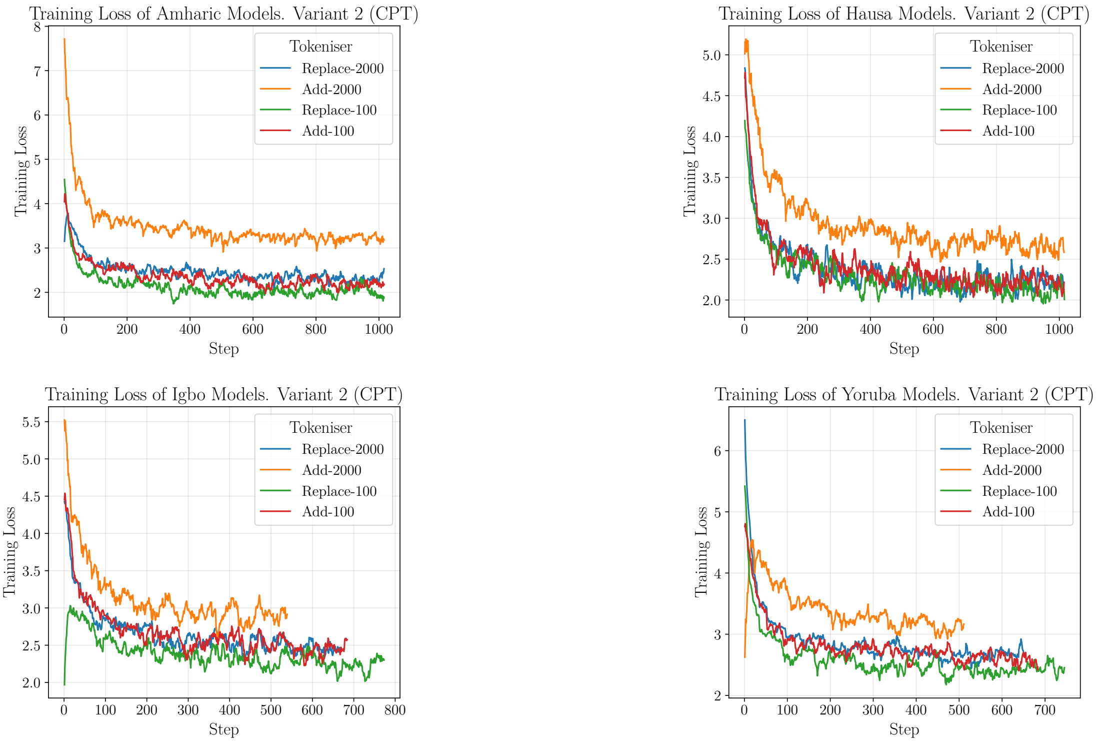


Figure A.4: Training losses of models adapted on Variant 2, CPT only. Training curves are smoothed using Exponential Moving Average with $\alpha = 0.1$.

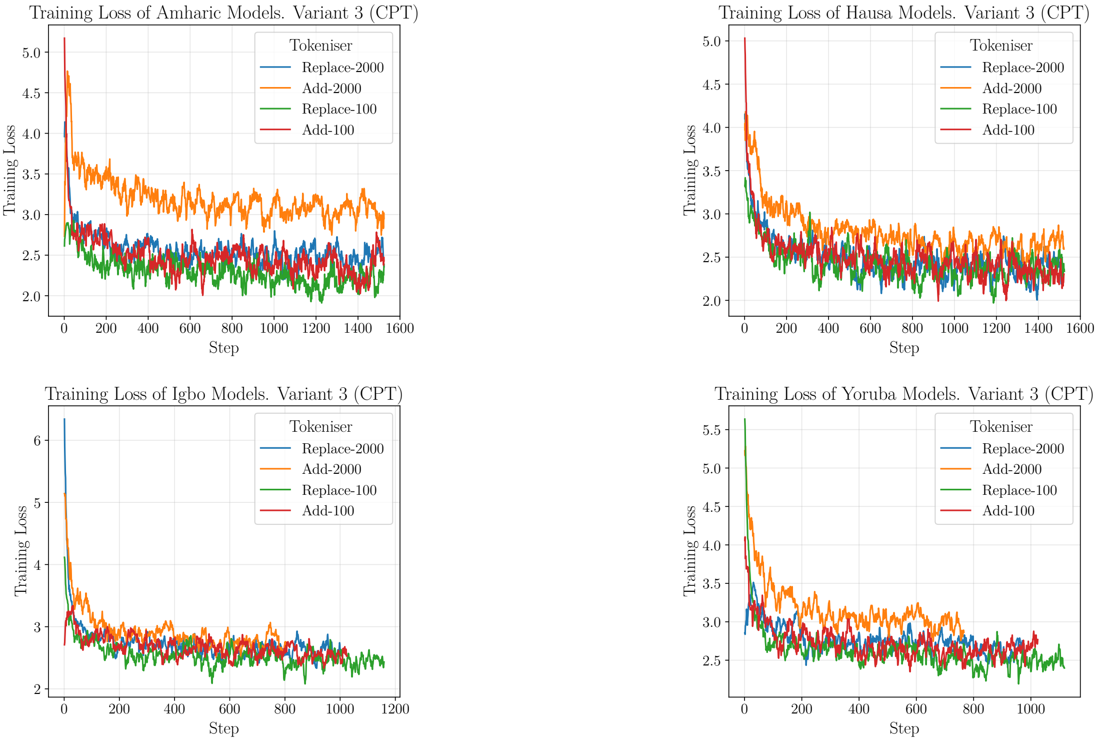


Figure A.5: Training losses of models adapted on Variant 3, CPT only. Training curves are smoothed using Exponential Moving Average with $\alpha = 0.1$.

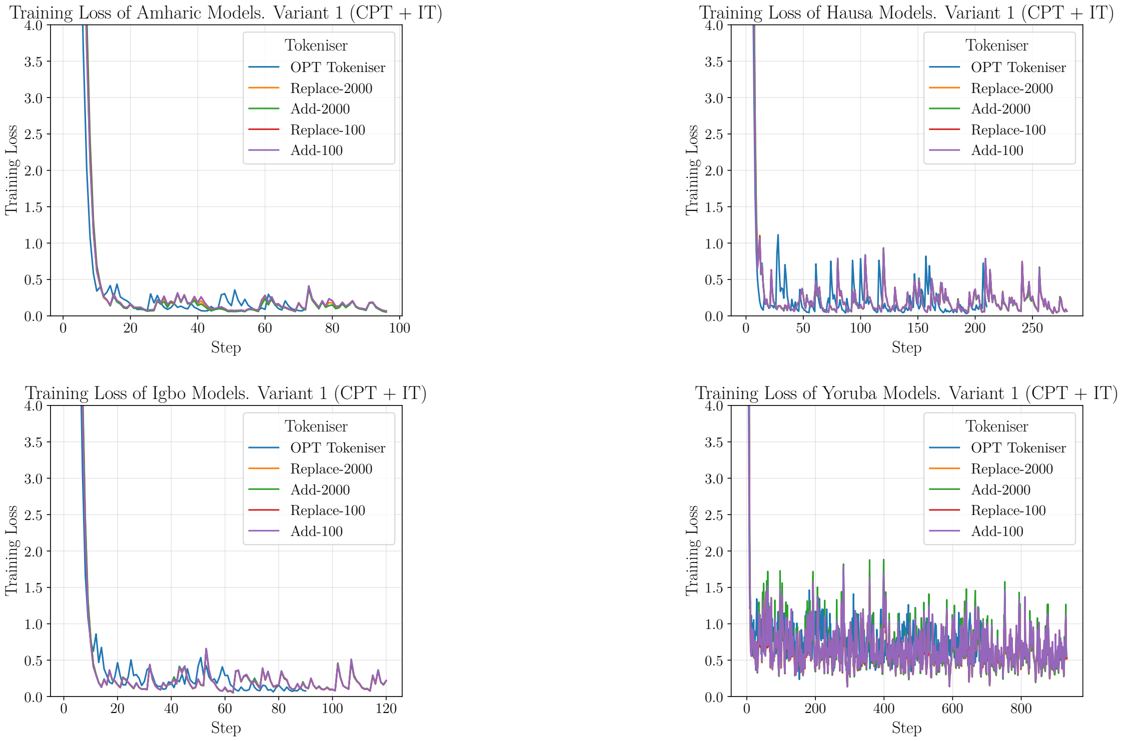


Figure A.6: Training losses of models adapted on Variant 1, CPT + IT. Training curves are smoothed using Exponential Moving Average with $\alpha = 0.5$.

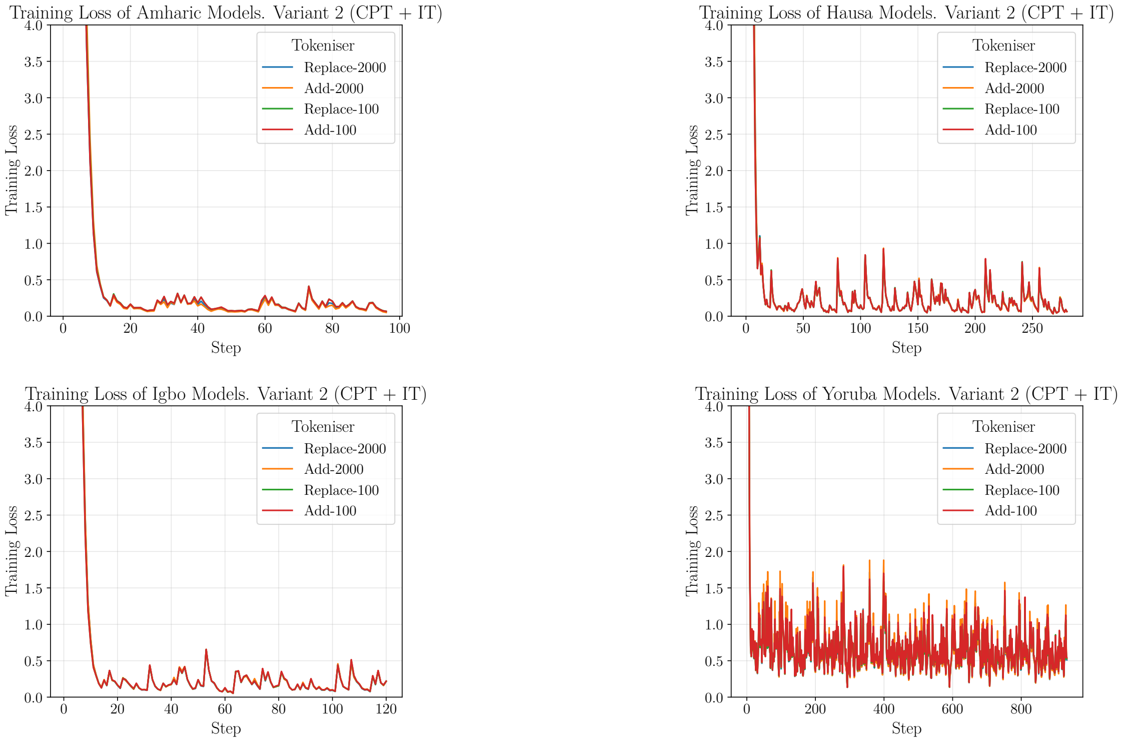
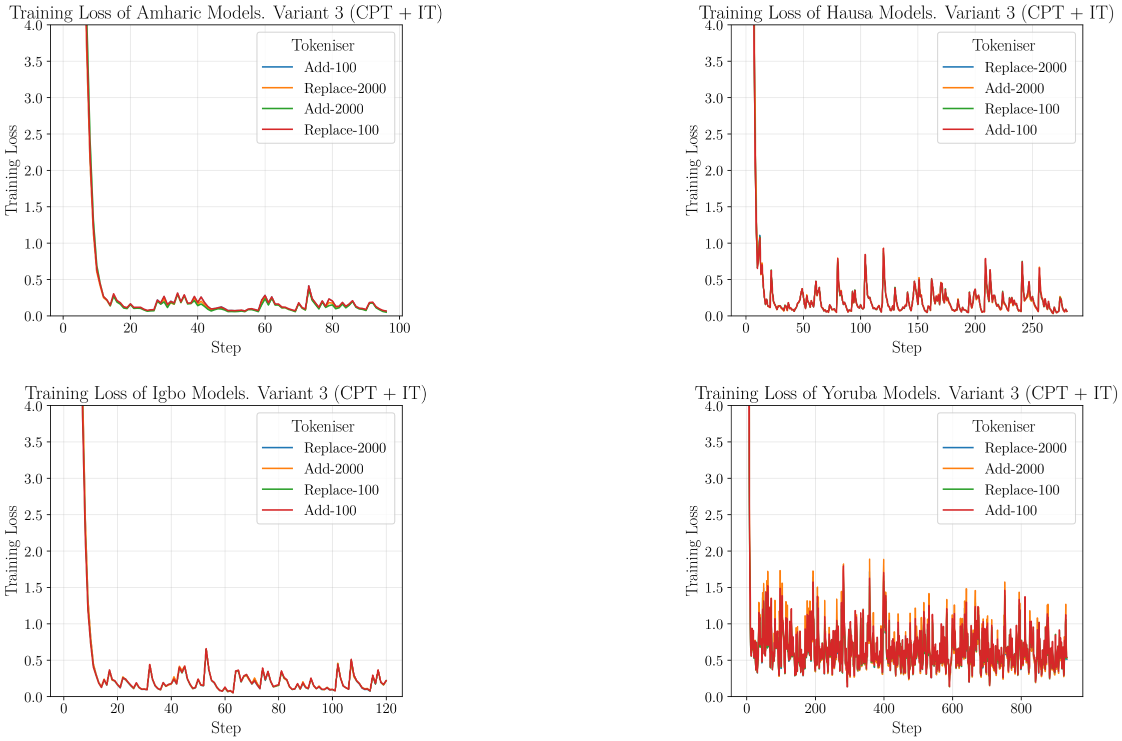


Figure A.7: Training losses of models adapted on Variant 2, CPT + IT. Training curves are smoothed using Exponential Moving Average with $\alpha = 0.5$.


Training Loss of Amharic Models. Variant 3 (CPT + IT)

Training Loss of Hausa Models. Variant 3 (CPT + IT)

Training Loss of Igbo Models. Variant 3 (CPT + IT)

Training Loss of Yoruba Models. Variant 3 (CPT + IT)

Figure A.8: Training losses of models adapted on Variant 3, CPT + IT. Training curves are smoothed using Exponential Moving Average with $\alpha = 0.5$.

A.4 Model Evaluation Prompt Format

Machine Translation

Translate the Yoruba sentence below to English.
 	
Sentence: {Yoruba Sentence 1}
Translation: {English Sentence 1}
 	
Sentence: {Yoruba Sentence 2}
Translation: {English Sentence 2}
 	
Sentence: {Yoruba Sentence 3}
Translation: {English Sentence 3}
 	
Sentence: {Yoruba Sentence 4}
Translation:

Table A.3: 3-shot evaluation prompt format for the machine translation task.

News Topic Classification

Use only the following topic labels: entertainment, health, politics, religion or sports.
 	
Text: {Text 1}
Label: {Label 1}
 	
Text: {Text 2}
Label: {Label 2}
 	
Text: {Text 3}
Label: {Label 3}
 	
Text: {Text 4}
Label:

Table A.4: 3-shot evaluation prompt format for the news topic classification task.

Question Answering

Context: {Context 1}
Question: {Question 1}
Answer: {Answer 1}
 	
Context: {Context 2}
Question: {Question 2}
Answer: {Answer 2}
 	
Context: {Context 3}
Question: {Question 3}
Answer: {Answer 3}
 	
Context: {Context 4}
Question: {Question 4}
Answer:

Table A.5: 3-shot evaluation prompt format for the question answering task.

Sentiment Classification

Use only the following sentiment labels: positive, neutral, negative.
 	
Text: {Text 1}
Sentiment: {Label 1}
 	
Text: {Text 2}
Sentiment: {Label 2}
 	
Text: {Text 3}
Sentiment: {Label 3}
 	
Text: {Text 4}
Sentiment:

Table A.6: 3-shot evaluation prompt format for the sentiment classification task.

Appendix B

Detailed Results

B.1 Tokeniser Evaluation

Tokeniser	Amharic	Hausa	Igbo	Yoruba
OPT BPE	10.61	2.01	2.91	2.94
Replaced-100	5.73	1.92	2.18	2.14
Replaced-500	5.02	1.88	1.92	1.97
Replaced-1000	4.73	1.84	2.06	1.92
Replaced-2000	4.08	1.83	1.89	1.87
Replaced-5000	4.45	1.8	2.01	1.85
Replaced-10000	4.3	1.79	2.02	1.82
Replaced-15000	3.41	1.79	1.83	1.82
Added-100	4.82	1.78	1.93	1.95
Added-500	3.25	1.56	1.7	1.68
Added-1000	2.84	1.46	1.61	1.56
Added-2000	2.48	1.36	1.54	1.46
Added-5000	2.1	1.27	1.47	1.36
Added-10000	1.88	1.23	1.45	1.31
Added-15000	1.77	1.21	1.44	1.29
Lang-dedicated	1.62	1.14	1.37	1.22

Table B.1: Detailed fertility rate results for all studied tokenisers.

B.2 Model Evaluation

B.2.1 Validation Perplexities

Tokeniser	**Amharic**	**Hausa**	**Igbo**	**Yoruba**
Replace-100	8.58	9.72	10.74	11.69
Add-100	9.98	10.32	12.25	13.07
Replace-2000	11.16	10.09	12.76	13.72
Add-2000	19.99	13.42	14.87	18.31

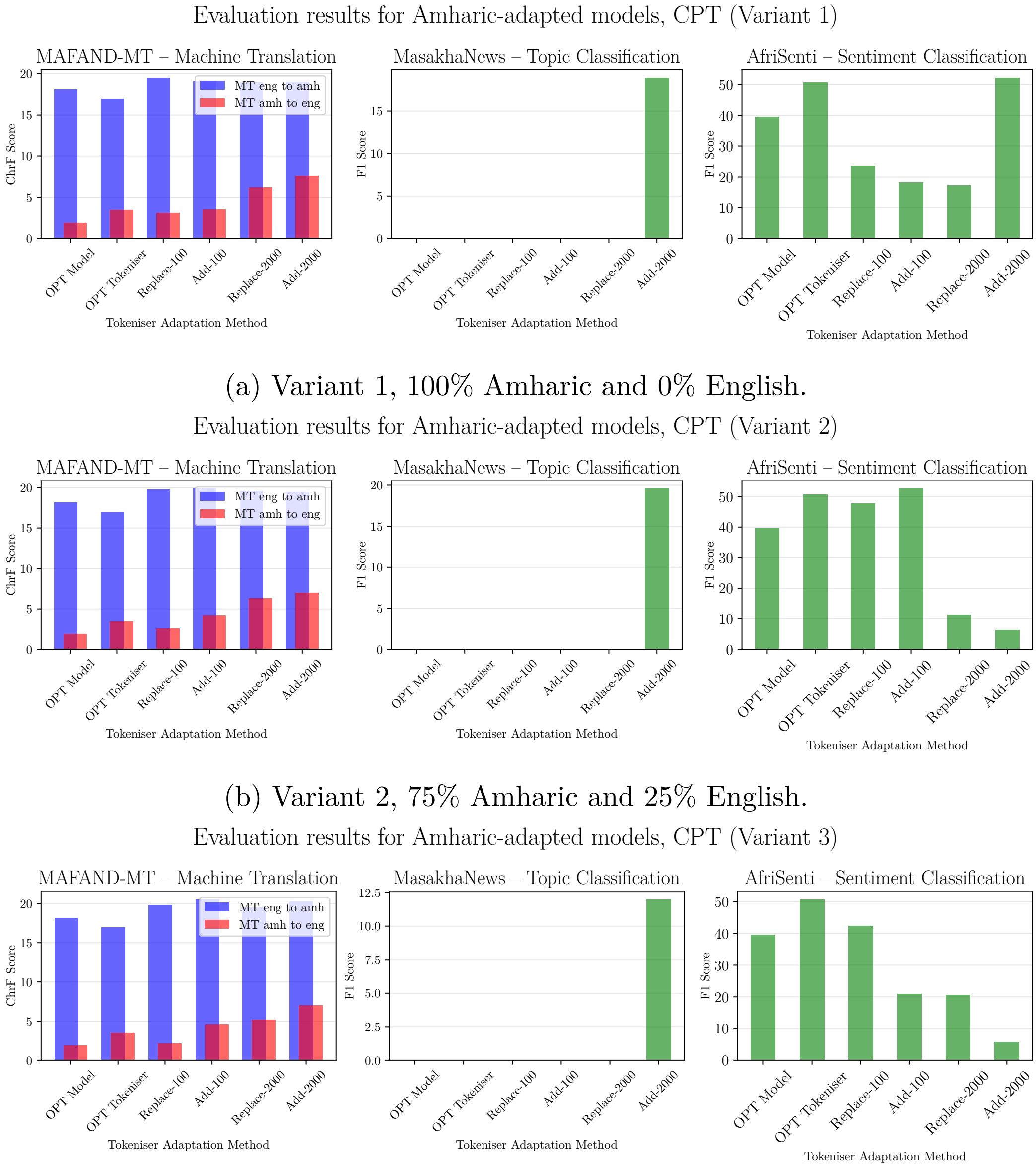
Table B.2: WURA validation set perplexities of models continuously pre-trained on 50% English and 50% target language data.

Tokeniser	**Amharic**	**Hausa**	**Igbo**	**Yoruba**
Replace-100	1.11	1.18	1.24	1.79
Add-100	1.11	1.83	1.25	1.81
Replace-2000	1.11	1.18	1.24	1.77
Add-2000	1.11	1.18	1.24	1.79

Table B.3: Aya validation set perplexities of models continuously pre-trained on 50% English and 50% target language data and instruction-tuned on 25% English and 75% target language.

B.2.2 Complete Downstream Task Evaluation Results

3-shot Evaluation Results of Amharic-adapted Models (CPT only)



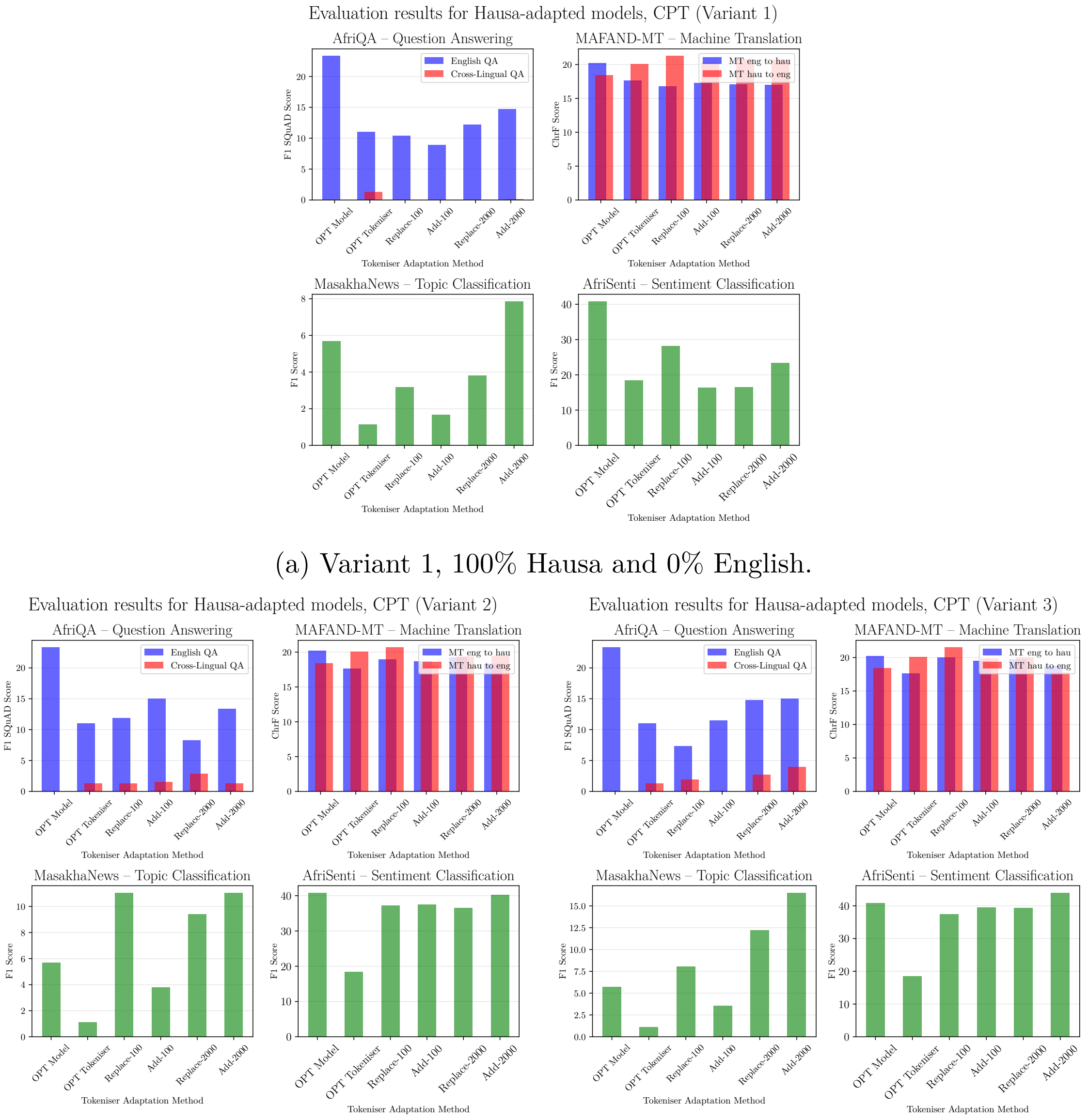
(a) Variant 1, 100% Amharic and 0% English.

(b) Variant 2, 75% Amharic and 25% English.

(c) Variant 3, 50% Amharic and 50% English.

Figure B.1: 3-shot evaluation of Amharic-adapted models, trained solely through continuous pre-training with different data variants. The empty results for topic classification indicate it achieved F1 scores of 0 for all adapted models but the last *Add-2000*.

3-shot Evaluation Results of Hausa-adapted Models (CPT only)



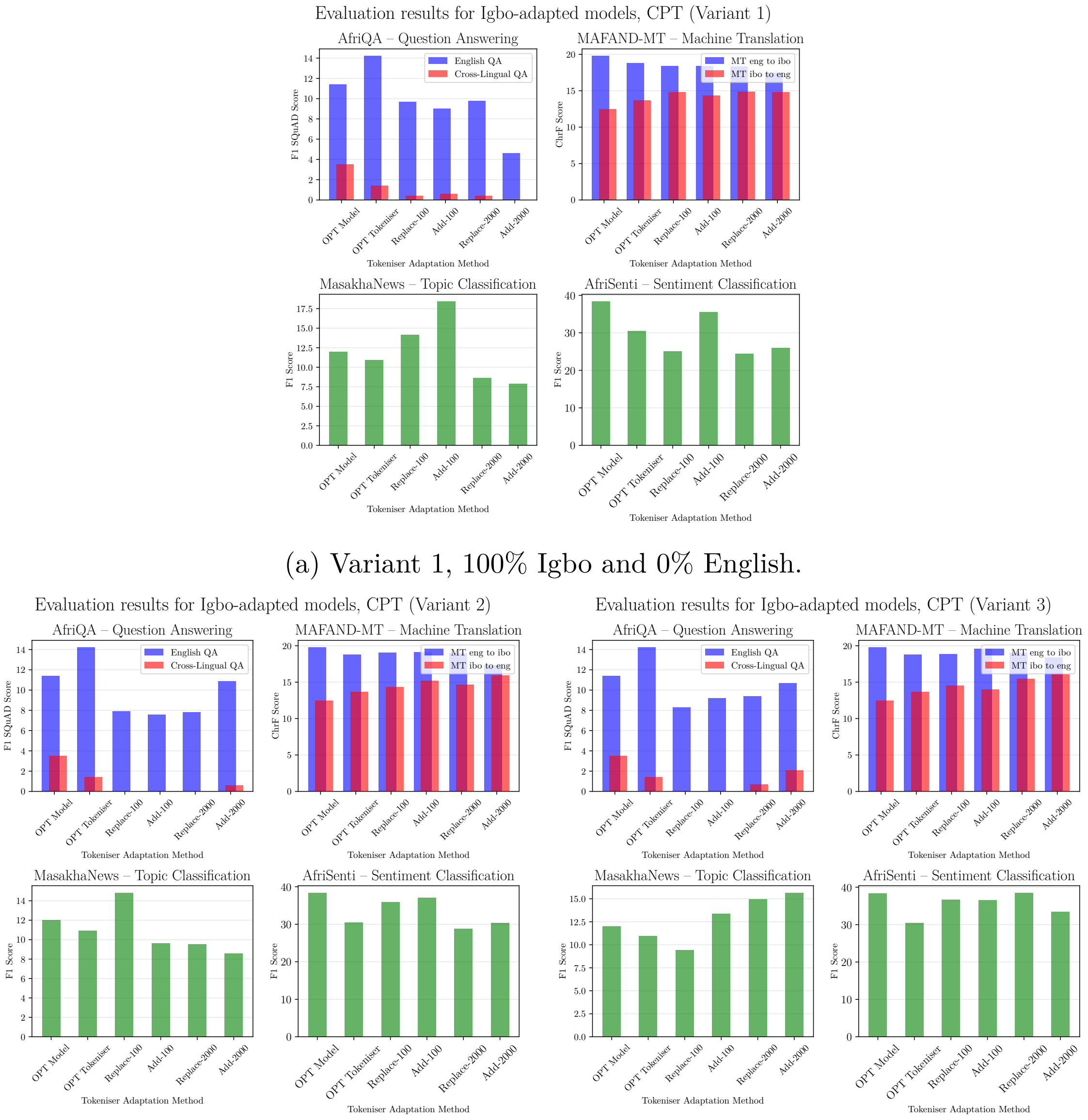
(a) Variant 1, 100% Hausa and 0% English.

(b) Variant 2, 75% Hausa and 25% English.

(c) Variant 3, 50% Hausa and 50% English.

Figure B.2: 3-shot evaluation of Hausa-adapted trained solely through continuous pre-training with different data variants.

3-shot Evaluation Results of Igbo-adapted Models (CPT only)



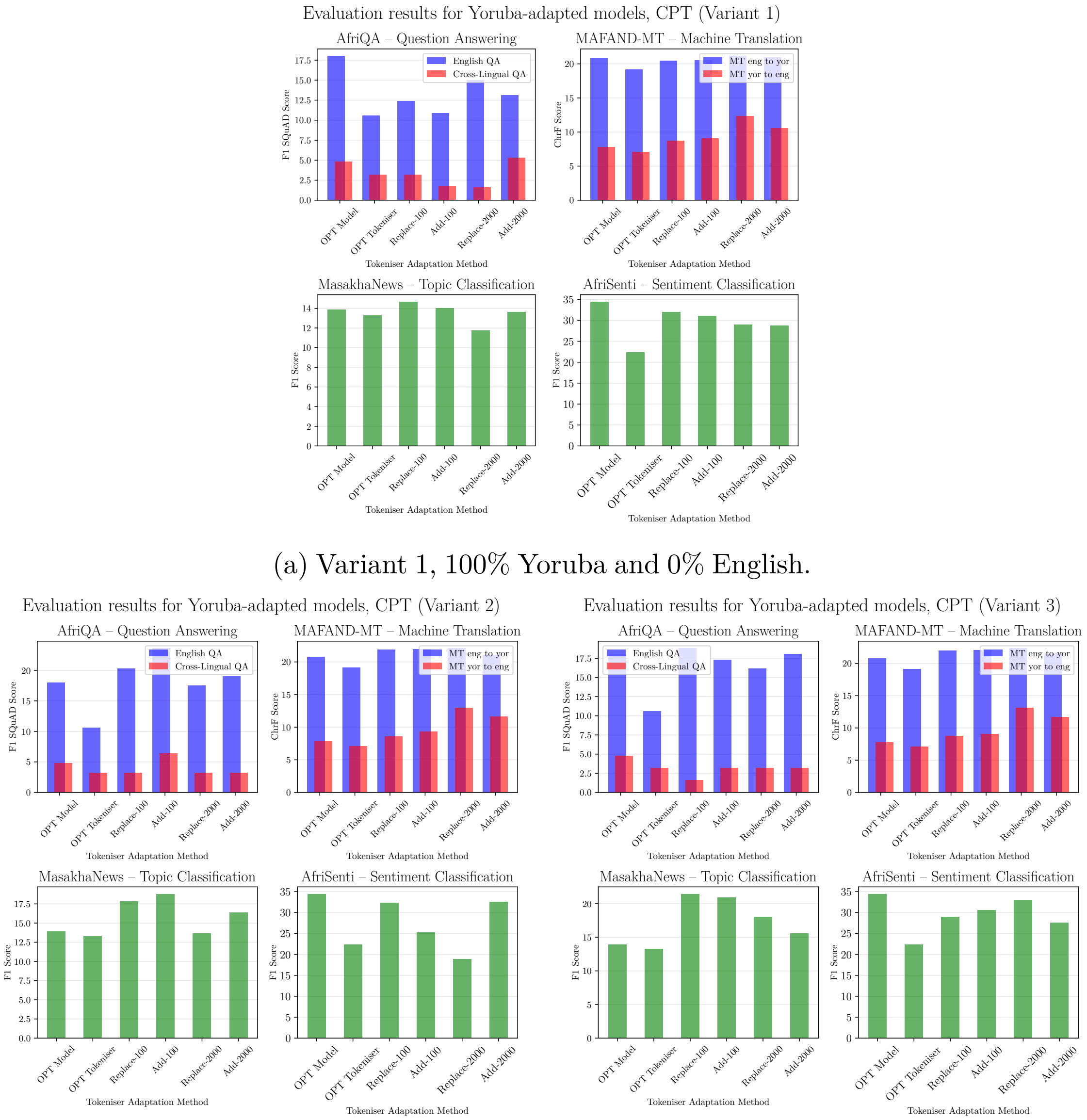
(a) Variant 1, 100% Igbo and 0% English.

(b) Variant 2, 75% Igbo and 25% English.

(c) Variant 3, 50% Igbo and 50% English.

Figure B.3: 3-shot evaluation of Igbo-adapted trained solely through continuous pre-training with different data variants.

3-shot Evaluation Results of Yoruba-adapted Models (CPT only)



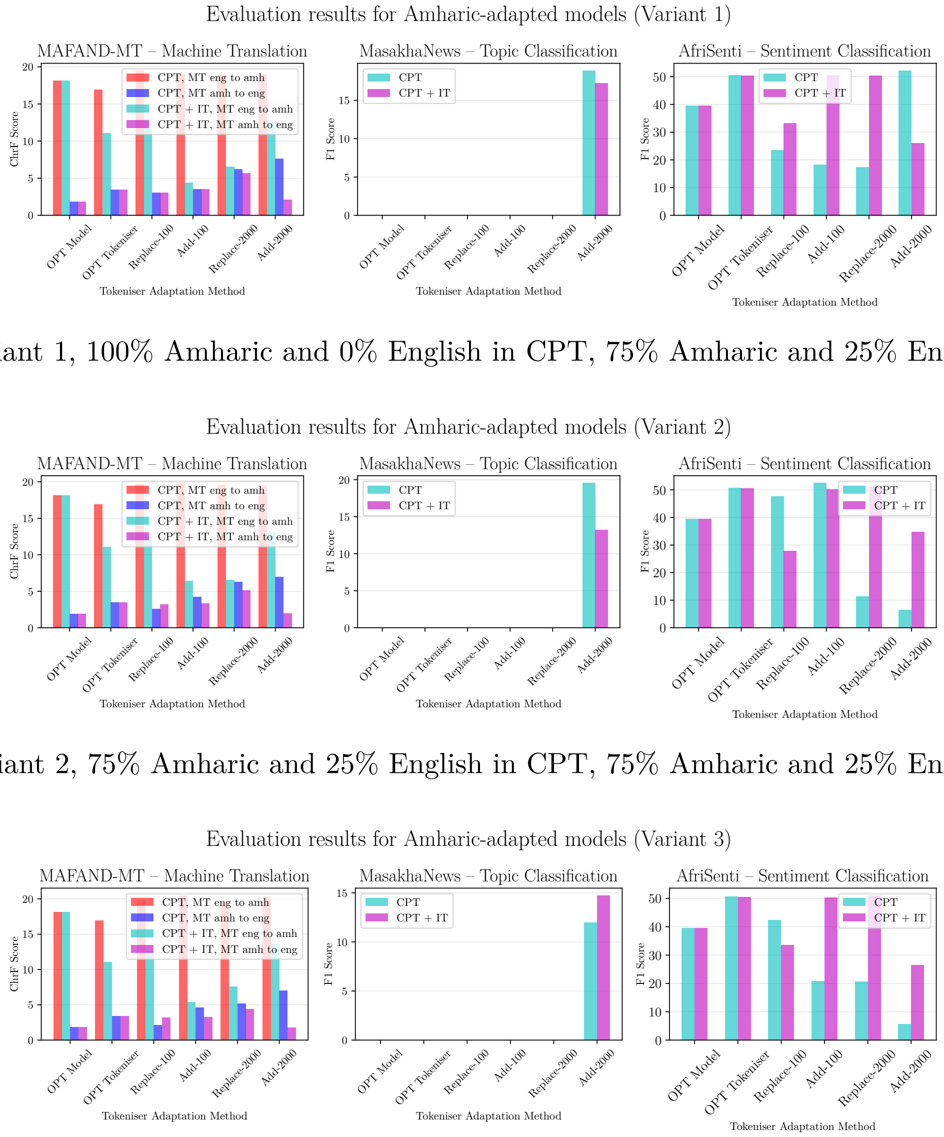
(a) Variant 1, 100% Yoruba and 0% English.

(b) Variant 2, 75% Yoruba and 25% English.

(c) Variant 3, 50% Yoruba and 50% English.

Figure B.4: 3-shot evaluation of Hausa-adapted trained solely through continuous pre-training with different data variants.

3-shot Evaluation Results of Amharic-adapted Models (CPT + IT)



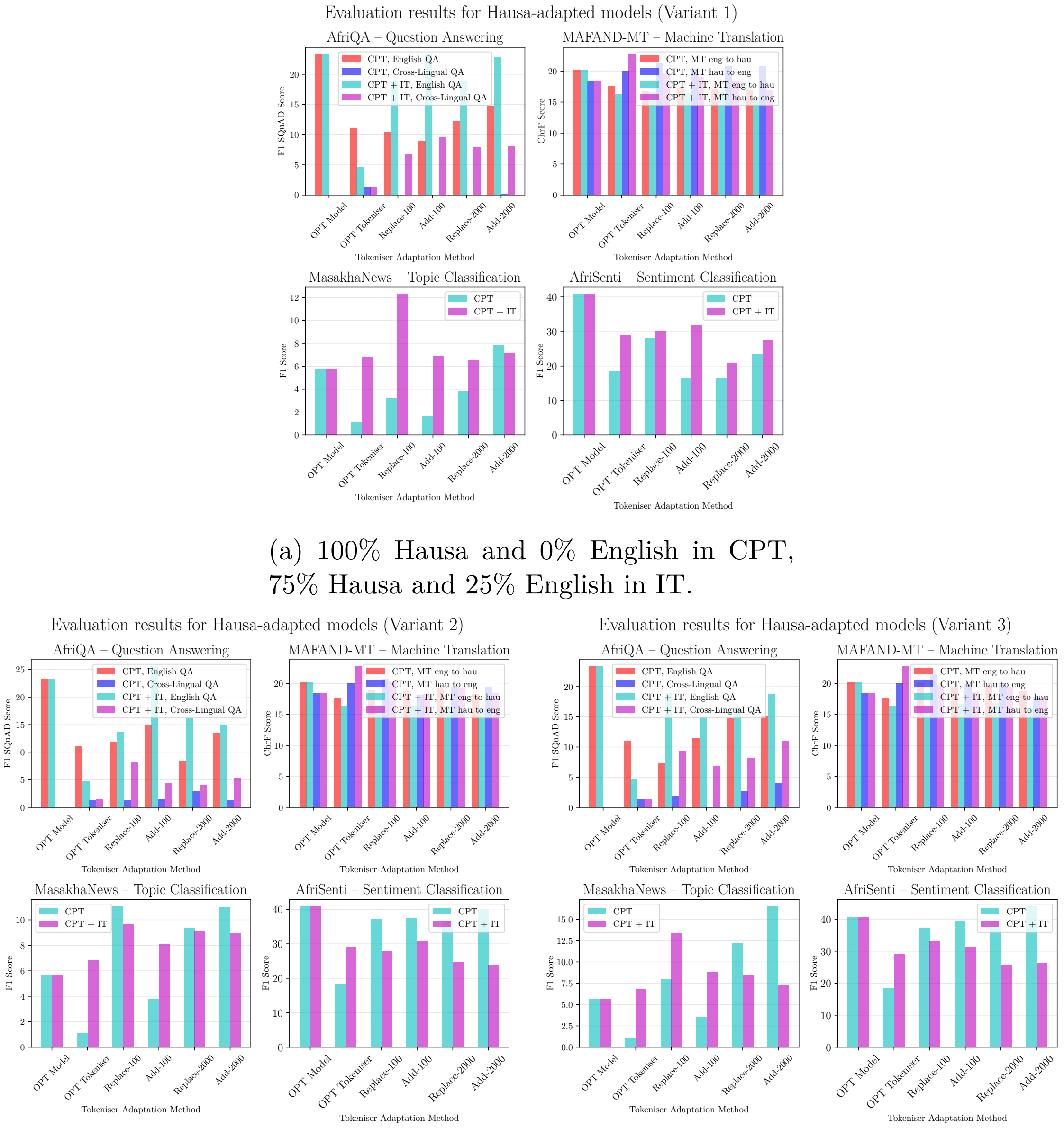
(a) Variant 1, 100% Amharic and 0% English in CPT, 75% Amharic and 25% English in IT.

(b) Variant 2, 75% Amharic and 25% English in CPT, 75% Amharic and 25% English in IT.

(c) Variant 3, 50% Amharic and 50% English in CPT, 75% Amharic and 25% English in IT.

Figure B.5: 3-shot evaluation of Amharic-adapted models, trained solely through continuous pre-training with different data variants. The empty results for topic classification indicate it achieved F1 scores of 0 for all adapted models but the last *Add-2000*.

3-shot Evaluation Results of Hausa-adapted Models (CPT + IT)



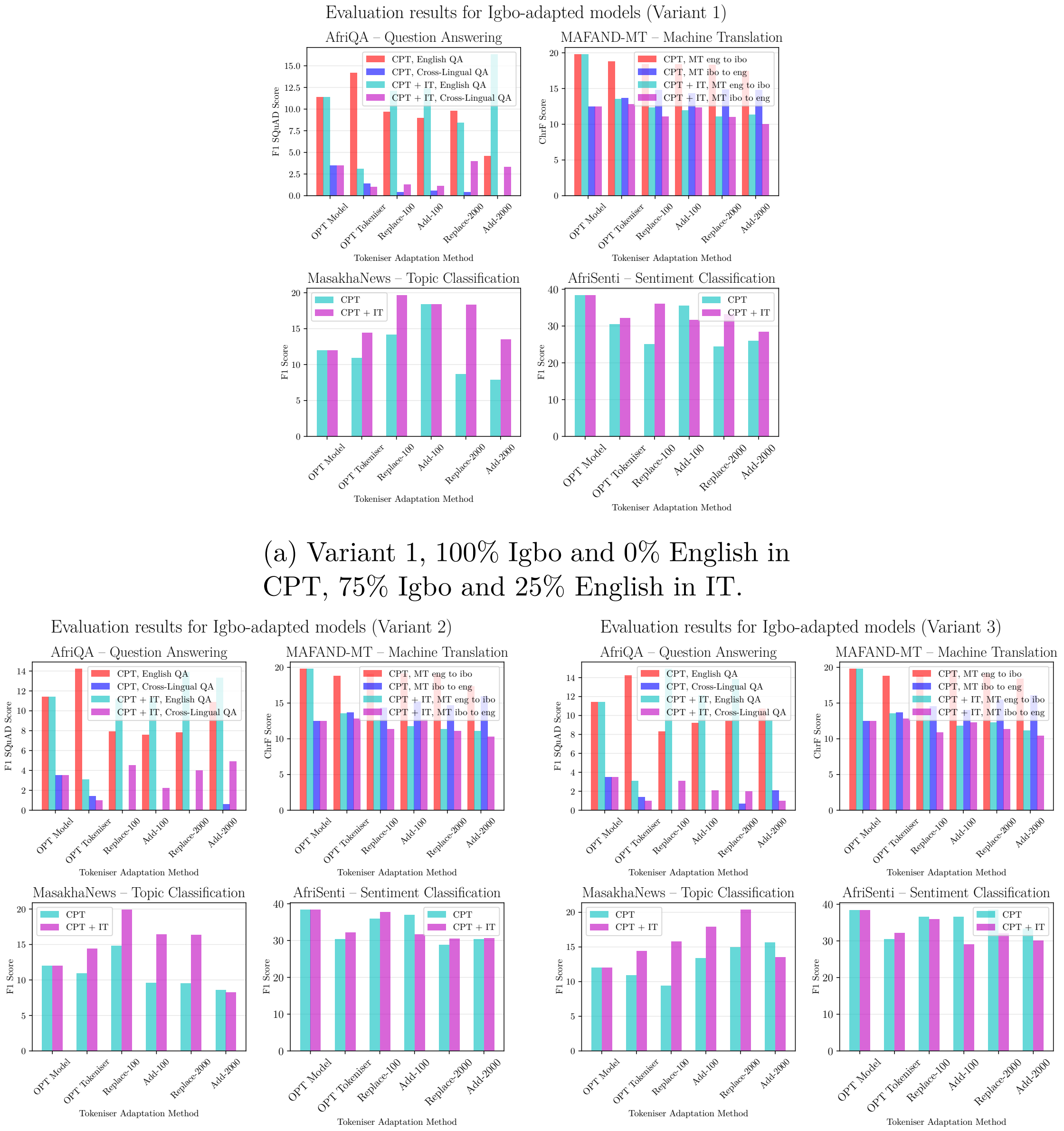
(a) 100% Hausa and 0% English in CPT, 75% Hausa and 25% English in IT.

(b) 75% Hausa and 25% English in CPT, 75% Hausa and 25% English in IT.

(c) 50% Hausa and 50% English in CPT, 75% Hausa and 25% English in IT.

Figure B.6: 3-shot evaluation of Hausa-adapted trained solely through continuous pre-training with different data variants.

3-shot Evaluation Results of Igbo-adapted Models (CPT + IT)



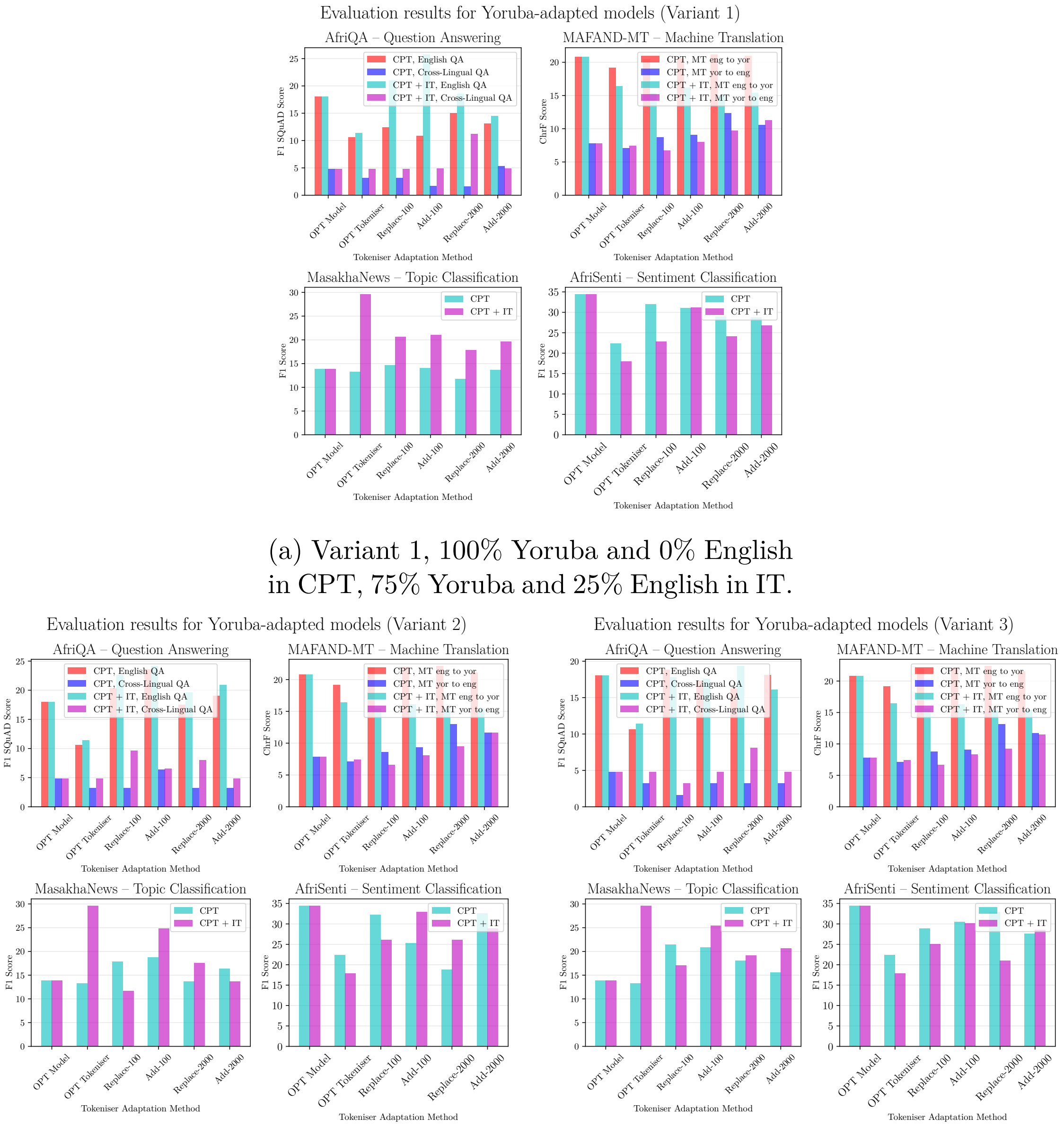
(a) Variant 1, 100% Igbo and 0% English in CPT, 75% Igbo and 25% English in IT.

(b) Variant 2, 75% Igbo and 25% English in CPT, 75% Igbo and 25% English in IT.

(c) Variant 3, 50% Igbo and 50% English in CPT, 75% Igbo and 25% English in IT.

Figure B.7: 3-shot evaluation of Igbo-adapted trained solely through continuous pre-training with different data variants.

3-shot Evaluation Results of Yoruba-adapted Models (CPT + IT)



(a) Variant 1, 100% Yoruba and 0% English in CPT, 75% Yoruba and 25% English in IT.

(b) Variant 2, 75% Yoruba and 25% English in CPT, 75% Yoruba and 25% English in IT.

(c) Variant 3, 50% Yoruba and 50% English in CPT, 75% Yoruba and 25% English in IT.

Figure B.8: 3-shot evaluation of Hausa-adapted trained solely through continuous pre-training with different data variants.

Aggregated 3-shot Evaluation Results per Tokeniser (CPT only)

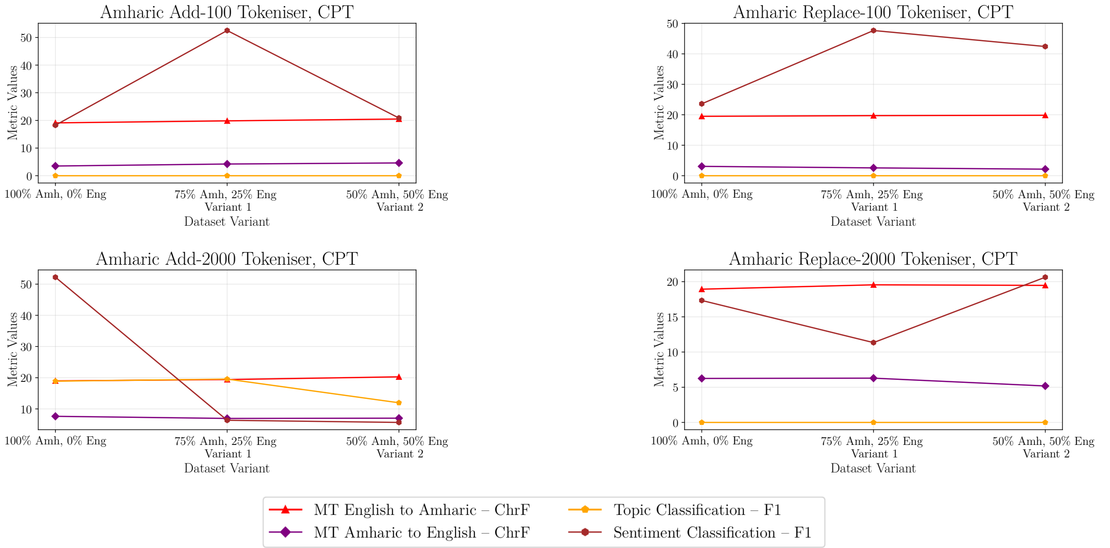


Figure B.9: 3-shot evaluation of Amharic-adapted models, trained solely through continuous pre-training on different dataset variants.

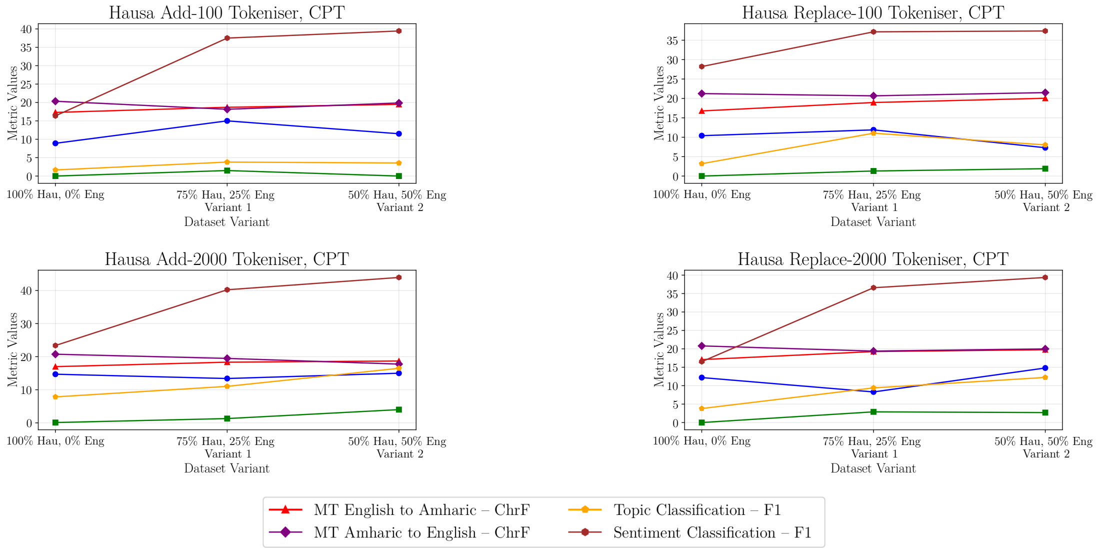


Figure B.10: 3-shot evaluation of Hausa-adapted models, trained solely through continuous pre-training on different dataset variants.

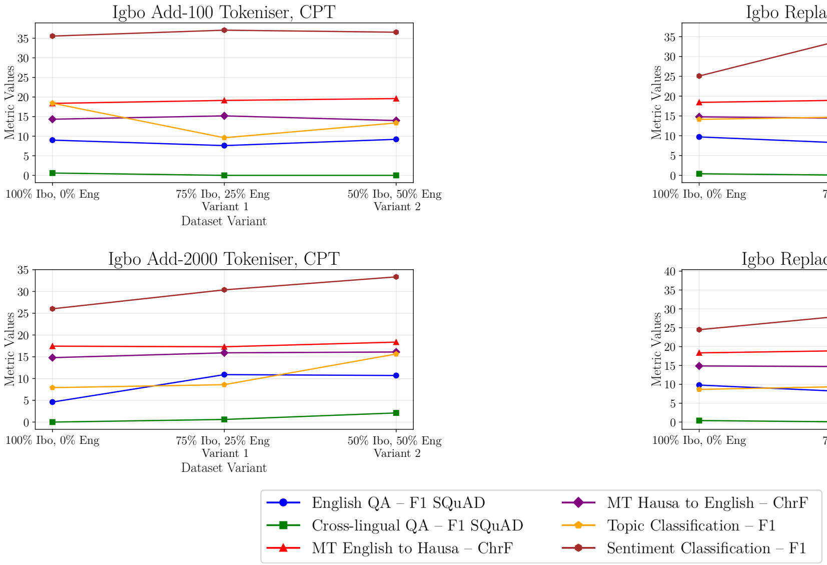


Figure B.11: 3-shot evaluation of Igbo-adapted models, trained solely through continuous pre-training on different dataset variants.

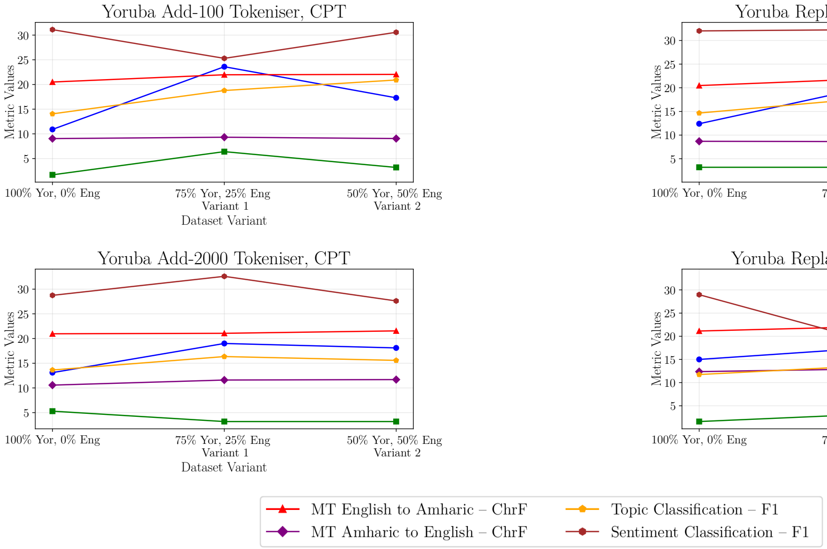


Figure B.12: 3-shot evaluation of Yoruba-adapted models, trained solely through continuous pre-training on different dataset variants.

Aggregated 3-shot Evaluation Results per Tokeniser (CPT + IT)

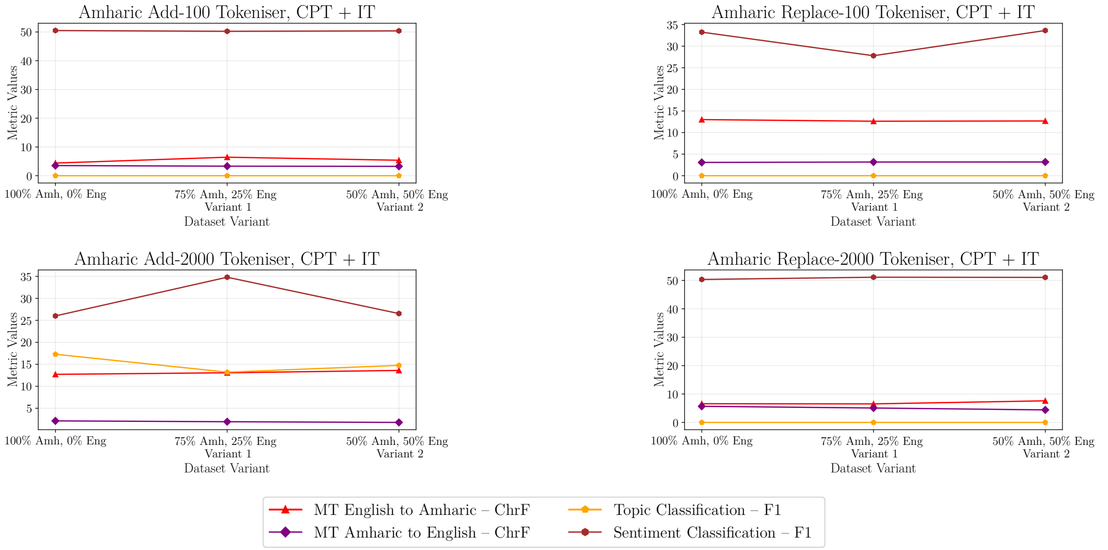


Figure B.13: 3-shot evaluation of Amharic-adapted models, trained through both continuous pre-training and instruction-tuning on different dataset variants.

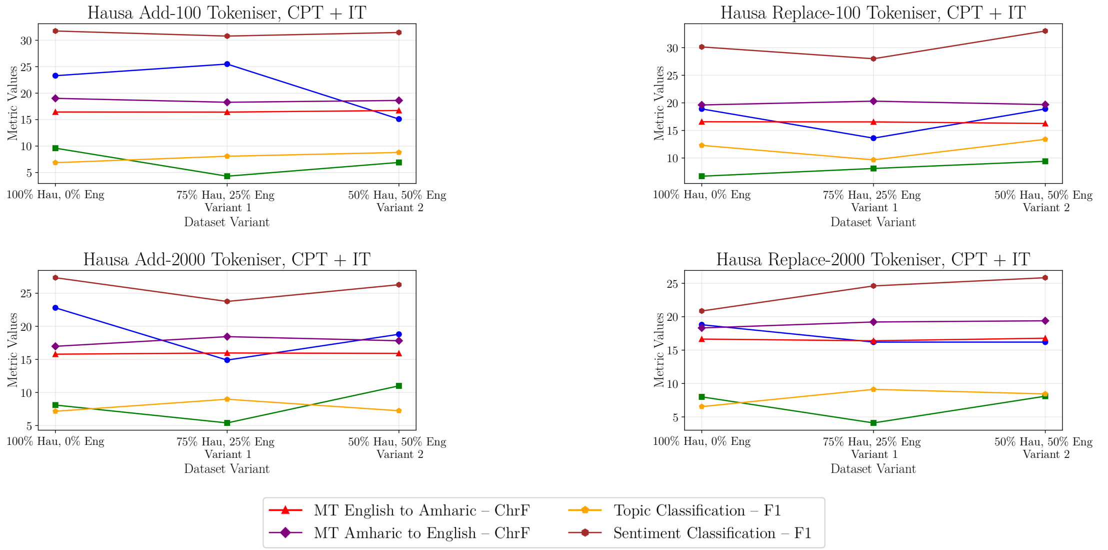


Figure B.14: 3-shot evaluation of Hausa-adapted models, trained through both continuous pre-training and instruction-tuning on different dataset variants.

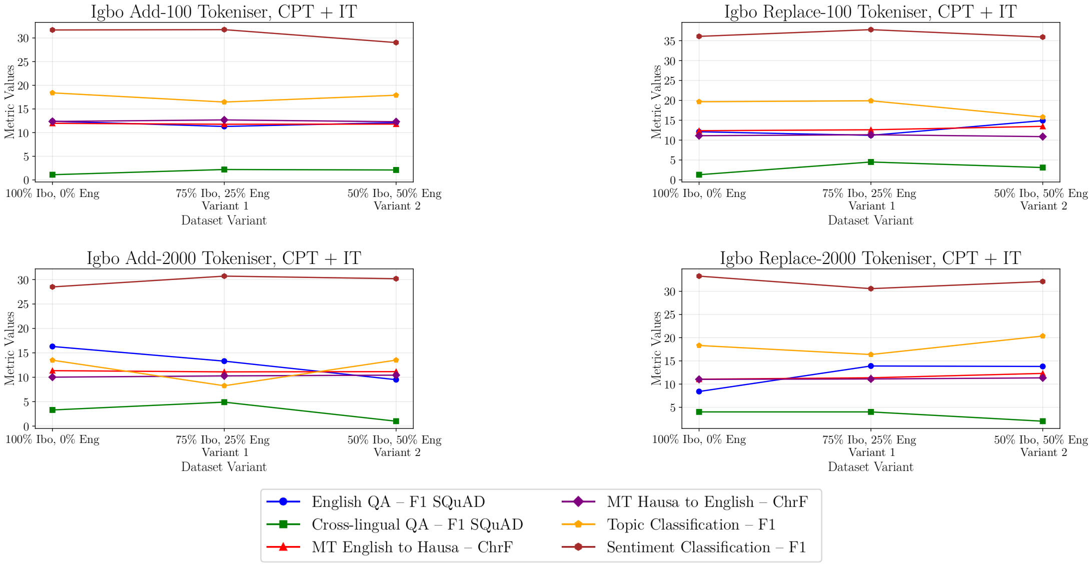


Figure B.15: 3-shot evaluation of Igbo-adapted models, trained through both continuous pre-training and instruction-tuning on different dataset variants.

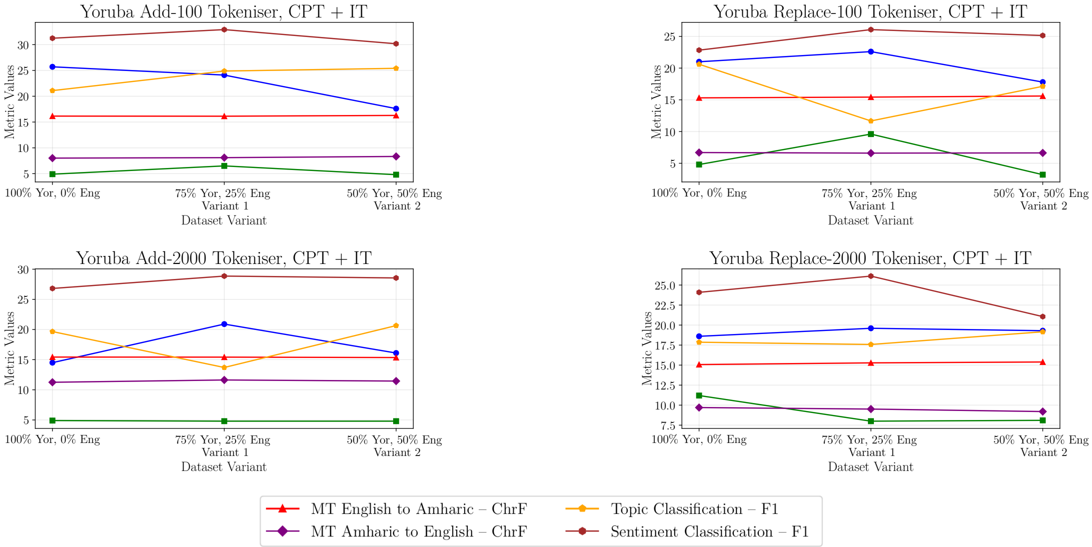


Figure B.16: 3-shot evaluation of Yoruba-adapted models, trained through both continuous pre-training and instruction-tuning on different dataset variants.

Appendix C

Additional Project Deliverables

C.1 Project Plan

Project Overview

The recent developments of Large Language Models (LLMs) capable of In-Context Learning revolutionised natural language processing. The further emergence of LLMs designed for following user-specified instructions wrapped in user-friendly web interfaces made cutting-edge technology available to the general public. The AI-powered tools are used for entertainment and industrial applications, such as software engineering, consulting and education. Nevertheless, depending on the choice of prompting language, the performance of LLMs and, therefore, tools based on them can differ significantly. Not optimised for use with languages poorly represented in training datasets, models perform particularly badly on low-resource languages, effectively disallowing citizens of African, Asian and Eastern European countries from accessing ChatGPT-like programs in their native languages.

The project aims to investigate the area of Large Language Models and methods of increasing their performance on low-resource languages. In particular, it will look into African languages and low-resource language adaption of models following the decoder-only gpt-like architecture.

Aims & Objectives

Aim 1: To develop a technique for a low-compute multi-lingual adaptation of a pre-trained decoder-only transformer language model.

Objectives:

1. Review the design of decoder-only transformer language model architectures, popular Large Language Model implementations and datasets used for training and evaluation.

2. Review fine-tuning and in-context learning techniques used in multilingual Large Language Models.

3. Develop a data pre-processing tool for target African languages (e.g. Yoruba and Amharic), making them a suitable input for further training.

4. Perform a language adaption of a language model through modifications of its embedding layers.

5. Compare the resulting model to its un-adapted version through evaluation on common tasks and available benchmarks (e.g. Topic classification – *MasakhaNEWS*, Named Entity Recognition – *MasakhaNER*, Question Answering – *AfriQA*).

Aim 2: To learn about the state-of-the-art techniques used for language modelling of low-resource African languages.

1. Review the basics of linguistic (e.g. language classification) and tokenisation methods for various scripts.
2. Review the available models, benchmarks and datasets in African languages.

Aim 3: To learn about the technological stack used for deep learning model pre-training.

1. Review the implementation details of Large Language Model neural network architectures in PyTorch and accompanying frameworks (e.g. *DeepSpeed*, *Accelerate*, *Megatron-LM*).
2. Develop a single-gpu training program for a gpt-like model.
3. Develop a multi-node, multi-gpu training program for a gpt-like model, supporting model parallelism and necessary hardware and software configuration of the distributed environment.

Expected Deliverables

- A literature review of state-of-the-art language models and methods used in multilingual and low-resource settings.
- A design specification for the proposed algorithm and a description of conducted experiments.
- Documented and functional software for data processing, deep learning model training, and evaluation.
- Trained model parameters provided in a form of checkpoints.
- Results obtained through the evaluation of developed language models.

Work Plan

A work plan is presented in Figure C.1.

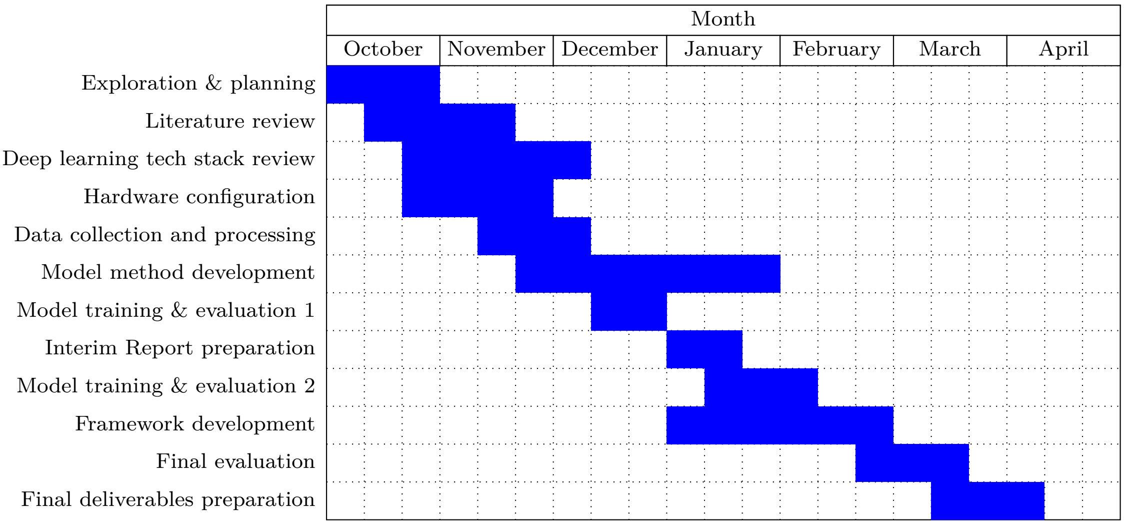


Figure C.1: The expected timeline of the project.

C.2 Interim Report

Project's goal

Given a pre-trained large language model, which was not trained on data in some languages $\{l_1, l_2, \ldots, l_n\} = L$, improve the performance of the language model on classic downstream tasks (e.g. news classification, machine translation, question answering) in all languages in A *simultaneously.* Assuming a common academic setting with no access to significant computational resources and a large amount of high-quality data for languages in A, this project aims to investigate adaptations to African languages, in particular.

Progress made to date

Extensive literature review – The landscape of neural Natural Language Processing is dynamically evolving and it is crucial to understand its dynamics. The literature review has been split into several categories, including the overview of the problem of low-resource language modelling, specific neural architectures and their improvements (e.g. various implementations of the Transformer), and available datasets for training and evaluating models in low-resource African languages. Moreover, it includes a review of existing language model adaptation techniques and mentions concepts from the area of linguistics, helpful for navigating through the theory of linguistics.

Collection of training and evaluation datasets – This project uses language data in two different ways: 1) data used to fine-tune particular elements of pre-trained models such as embeddings layers, and 2) datasets used for evaluation of the model's performance on common downstream tasks. The fine-tuning dataset for African languages has been constructed using WURA – a recently published work extending the contents of the Common Crawl data. To compare the adaptation quality to other languages, a dataset for German has been generated by sampling the multilingual, cleaned version of the Common Crawl dataset. Moreover, a small Polish dataset has been gathered from available sources such as Polish literature and Wikipedia pages. On the other hand, a number of evaluation datasets have been processed. Currently considered evaluation tasks are news classification, sentiment classification, named entity recognition, question answering and machine translation.

Fine-tuning and the development of architectural changes to a model – The English-based GPT2 model has been successfully modified by relearning its embedding layers on a Polish dataset and combining them in a Mixture of Experts component. The router for the MoE components has been further trained on a mix of pre-training English and fine-tuning Polish data using a parameter-efficient approach. The model has not been evaluated using the In-Context Learning evaluation setup, however, it displays multilingual capabilities.

Remaining work

Evaluation setup development – This step is of paramount importance to the project since it will allow conducting experiments showing the effect of the proposed adaptation technique. Furthermore, it will allow experiments with adaptation hyper-parameters and ablation studies. Preliminary experiments and existing work show that considered model sizes (up to 7B parameters) do not work well in a zero-shot in-context learning setting. Instead, a few-shot approach produces more reliable evaluation results and, therefore will be pursued. The evaluation should cover a number of downstream tasks for which the data has already been gathered and compare the original and adapted versions of considered language models.

Generalisation of architectural changes to a family of decoder-only models – A framework to perform a parameter efficient fine-tuning (PEFT), relearning the embedding layers and combining them into a Mixture of Experts component will allow for automatic application of the proposed method to several language model implementations. One of the most commonly used implementations of language models is the Python *transformers* library. The proposed adaptation framework should be locally incorporated into the *transformers* abstraction mechanism.

Experiments with varying parameters and ablation studies – Having both the evaluation setup and the framework implementation, the experiments need to be conducted. Most importantly, the adaptation quality should be evaluated, including the potential loss in the quality of a model on languages used in the pre-training dataset (e.g. English). Furthermore, given the research nature of this project, an ablation study should be considered to explore the possibility of multi-language adaptation without the additional step of Mixture of Expert Embeddings.