Analyzing the Effectiveness of Morphologically Motivated Tokenization on Machine Translation for Low-Resource Languages

Stanford CS224N Custom Project

Abhishek Vangipuram
Department of Computer Science
Stanford University
abhiv@stanford.edu

Emiyare Ikwut-Ukwa
Department of Computer Science
Stanford University
emiyare@stanford.edu

William Huang
Department of Computer Science
Stanford University
willsh@stanford.edu

Abstract

Tokenization, the division of strings into smaller units, say, a sentence into a list of words, is a critical and fundamental step in various modern NLP techniques, including translation. Extremely naive tokenization — for example, simply segmenting units at every space or punctuation mark — ignores the internal structure of words, which can hurt generalization, especially in a scarce-resource context. We developed a morphologically-motivated tokenizer for Obolo, a low-resource language with complex morphology, and compared its effect on performance in downstream tasks (machine translation) to standard tokenizers like BPE. Our findings were inconclusive, with slightly worse performance with our custom tokenizer compared to just using BPE. We also found that training on Obolo specific data alone was vastly superior than fintetuning with a pretrained machine translation model (T5 & mBART).

1 Key Information to include

- Mentor: Moussa Doumbouya
- Team Contributions:
  - Abhishek Vangipuram: Created custom Transformer model and ran experiments with BPE tokenization and custom tokenization.
  - Emiyare Ikwut-Ukwa: Sourced and cleaned data. Conducted literature review. Created custom Obolo tokenizer and detokenizer.
  - William Huang: Queried LLMs for baseline results. Finetuned large NMT models for Obolo usage.
  - Everyone contributed equally to the writing of this report.

2 Introduction

If a tokenizer treats all words as a single unit, downstream tasks using the tokenizer may struggle to predict the behavior of novel words which are composed of already-encountered subunits (e.g., a task utilizing such a tokenizer may not infer, after training on the words “dog,” “dogs,” and “cat,” the behavior of the novel word “cats”). Prevailing tokenization algorithms, including Byte Pair

Encoding (BPE) and WordPiece, leverage unsupervised learning to tokenize at the "subword level," looking for frequently re-occurring substrings. However, these models often slice up words in ways that are incongruent with human intuition and linguistic theory. In particular, these tokenization strategies often separate parts of a word which linguists would describe as belonging to the same morpheme, the fundamental linguistic unit of meaning. For example, while a linguist, or even an ordinary English speaker operating on intuition, would likely segment the word "paratrooper" as {para, troop, er}, BPE yields {par, atro, oper} [1]. Intuitively, this poses a question: will a tokenizer that can effectively break down words into their "true" subcomponents outperform a more naive one?

Our goal is to compare the effectiveness of different tokenization strategies for a morphologically complex and low-resource language. One such language is Obolo, a highly-prefixing Cross River language of southern Nigeria, with less than a million words worth of parallel text readily available. As an example of morphological complexity, the single Obolo word "ikekibonisisi" means "and then they (s) *were* intending to soon be *going*" (these italics represent focus, which is encoded morphologically in Obolo). Given a small dataset, it is impractical, at least intuitively, to infer much about similarly complicated words without decocomposition into meaningful elements (here, i-ke-ki-bo-ni-si-si). With this motivation, we compare existing unsupervised tokenizers with morphologically-motivated methods.

3 Related Work

Jabbar [1] develops an English tokenizer called "MorphPiece", which comes closer to true morphological tokenization and outperforms GPT-2 on several tasks. MorphPiece relies on a large, predetermined morpheme look-up table built from MorphyNet database. This table maps English words to their respective morphological structures (e.g., "batting" to {bat, ing}). MorphPiece divides English words into the morphemes that compose them as prescribed by the table and uses BPE to tokenize the remaining data. MorphPiece is used to develop MorphGPT, a language model with the same architecture as GPT-2 (base), save for the introduction of the MorphPiece tokenizer. Although MorphGPT significantly outperforms GPT-2, as measured by the GLUE Benchmark, it's difficult to generalize: there's no clear way to implement this system without a pre-existing table converting words into lists of morphemes and lists of morphemes into words.

Rehman et al. [2] use a list of approximately 5,700 morphemes to tokenize Urdu. Unlike MorphPiece, this tokenizer doesn't have access to word $\rightarrow$ morpheme list mappings. Instead, it employs a handful of algorithms to divide the input into a sequence of tokens from the provided vocabulary. As before, a large problem with this result is that relies on a body of pre-existing data that does not exist for many low-resource languages. Moreover, without a fail-safe reference like MorphyNet, it's possible to misidentify tokens, incorrectly subdividing words.

Nzeyimana and Rubungo [3] integrate a morphological tokenizer into a BERT model of Kinyarwanda, a low-resource, morphologically-rich language. Unlike the models discussed above, KinyaBERT ultimately keeps one embedding per word: after obtaining embeddings for each morpheme within a word, the model concatenates those embeddings to make the final word embedding. The model also uses an unsupervised part-of-speech tagger, incorporating a part-of-speech tag into each word embedding. KinyaBERT outperforms baselines by 2% in named entity recognition and by 4.3% on average on the GLUE benchmark.

4 Approach

Obolo Tokenizer:
We've developed a preliminary Obolo tokenizer from scratch. Because the vast majority of Obolo's morphological complexity lies in the conjugation of its verbs, our tokenizer aims to split verbs into the morphemes that compose them while avoiding subdivision of other parts of speech. Aaron describes the structure of the Obolo verb as shown in figure 1.

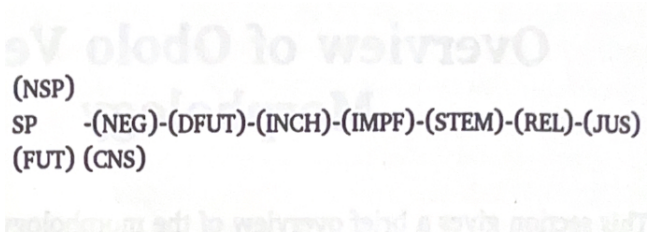

Figure 1: Morpheme Order in the Obolo Verbal Group [4]

That is, a verb consists of a neutral subject prefix, a subject prefix, or a future prefix, or none of the above, optionally followed by a negative prefix or consequential prefix, optionally followed by a definite future prefix, optionally followed by an inchoative prefix, optionally followed by an imperfective prefix, followed by the stem (which we refer to elsewhere as the "root"—the distinction is irrelevant here), optionally followed by a relative suffix, optionally followed by a jussive suffix.

Obolo verbs can also partially reduplicate, which serves several grammatical functions, including focus and nominalization. A partially reduplicated verb in Obolo consists of the root's initial consonant and vowel, followed by the root itself. For example, *sa* $\rightarrow$ *sasa*, and *chieek* $\rightarrow$ *chechieek* (the *i* in *chieek* is technically the consonant /j/, or the English "y-sound").

After dividing at spaces and punctuation marks, our tokenizer attempts to split each word into several verbal morphemes. If a given word belongs to a small manually-created list of "indivisible words" (commonly occurring words that look like verbs, as it were, but are not), they are returned undivided. Otherwise, the tokenizer first extracts prefixes from the left end in the order described by Aaron and then extracts suffixes from the right end. It also checks if the remaining middle portion of the word is partially reduplicated, removing the reduplicative syllable and tokenizing a prefix *REDUP* if reduplication is found. Finally, it tests to see if the remaining portion of the verb belongs to a manually devised list of 464 verb roots, a subset of those which appear in at least 3 different conjugations in the data. If the verb root passes this validation check, the given word is divided into the tokens identified. Otherwise, the process is aborted and the original word is returned. As an example, this entire process breaks the input *mbakitutumu* into the sequence *m ba ki REDUP tumu*.

We also have simple handling for the edge case in which a morphological prefix is also the substring prefix of the root. For example, *si* is a verb prefix and *siik* is a verb root. Given the input *nsiik*, we successfully tokenize into *n siik* rather than attempting *n si ik*, identifying *ik* as an invalid root, and aborting the process.

Obolo Detokenizer:
Our detokenization works with a modified version of our Obolo tokenizer. This modified tokenizer's only change is that it marks prefixes with trailing hyphens (e.g., "ma-") and suffixes with leading hyphens (e.g., "-be"). The actual detokenization process, which operates on an input list of tokens, greedily adds morphemes to words (left-to-right) such that a word has at most one non-affix, the root, and that within a word all prefixes are preceded only by other prefixes, and all suffixes are succeeded only by other suffixes. Whenever these conditions cannot be satisfied, a space is added and a new word begins. If followed by a token that may be reduplicated, the token "REDUP-" is converted into a reduplicative syllable as appropriate. Otherwise, it's erased.

Custom Transformer Model:
To compare the effectiveness of different tokenization schemes on machine translation, we created a custom Transformer model to train on, with the flexibility of using multiple tokenizers.

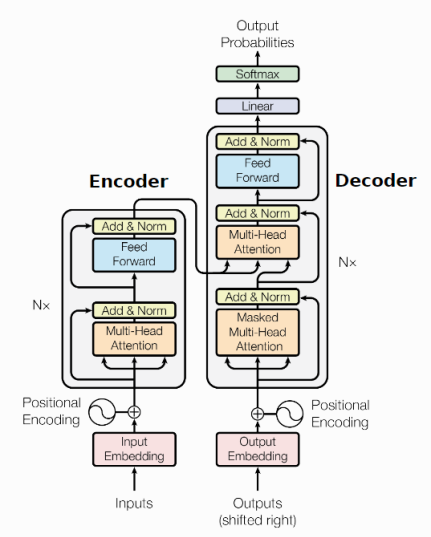

Figure 2: Standard Transformer Architecture

We use the standard Transformer model as shown in Figure 2 with 3 encoder layers and 3 decoder layers, using 8-headed attention. The variations on this model are simply the input and output token embedding steps as we change the tokenizer and detokenizer from BPE to our custom functions.

For Obolo to English translation we use the GPT-2 BPE detokenizer for English and either a BPE tokenizer for Obolo trained using our train set or our custom tokenizer function. For English to Obolo translation we use the GPT-2 BPE tokenizer for English and either a BPE detokenizer for Obolo trained using our train set or our custom detokenizer.

Fine-tuning of Pre-Trained NMT Models

We also fine-tuned T5 and mBART, which are both encoder-decoder architectures. However, these models differ in their pretraining; T5 is pretrained with a variety of tasks, differentiated by a prefix like 'summarize' or 'translate English to German,' but mBART is specifically pretrained on multilingual translation (in particular, multilingual denoising).

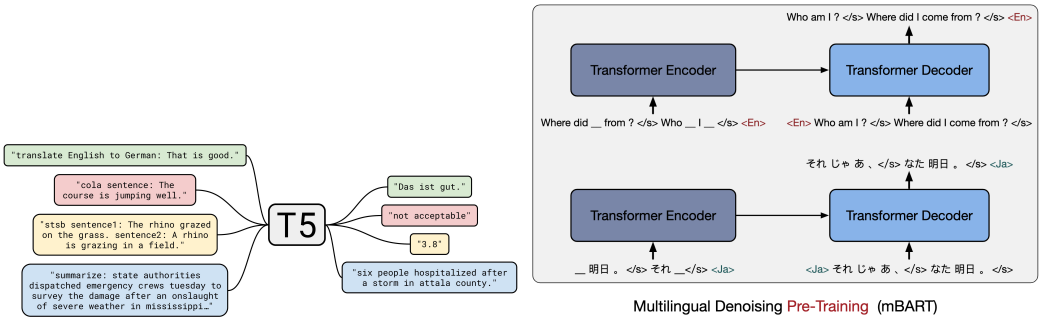

Figure 3: Pretraining of T5 (left) and mBART (right), respectively.

Finetuning these models is pretty straightforward because in both cases, translation tasks were already used to some capacity. Therefore, we simply perform gradient descent starting from the pretrained weights while passing in examples of Obolo $\rightarrow$ English translations in the same format seen in pretraining. In the below figure, you can see this paradigm for mBART.

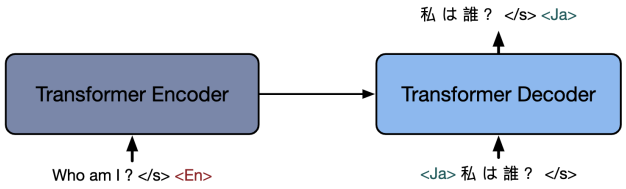

Figure 4: Example of finetuning with mBART.

Baselines:

We have two main baselines to compare to. First, we compute the raw performance of current SoTA LLMs with Obolo to English translation by cleverly querying Llama3-8B-Instruct. We compare this to the same model's performance on French to English translation to compare the differences due to the fact that Obolo is a low-resource language. The second baseline we run is a custom Transformer NMT model trained on our Obolo data for Obolo to English and English to Obolo translation, with only BPE tokenizers. The implementations of these baselines are explained in depth in Section 5.3.

5 Experiments

5.1 Data

Our Obolo data consists solely of the Obolo Bible [5], which contains approximately 900,000 words, at least 20,000 distinct words, about 9,000 of which occur only once. Each verse is paired with the King James Bible (an English translation). These are put into 31,097 verses, tagged as Obolo and English pairs, and all set to lowercase. The paired nature of this dataset is designed to perform the task of machine translation. An example pair is the following:

```
Obolo:   "me ibebene, awaji îrom isinyọ̃n mè linyọ̃n."
English: "in the beginning god created the heaven and the earth."
```

We use this source of data in all of our subsequent experiments, including evaluation and future training and fine-tuning. We randomly chose 27987, 500, and 2610 verses for our train, validation, and test sets, respectively. For one of the baseline tests, we also use OPUS Books [6] as a comparable dataset for French to English translation.

5.2 Evaluation method

We use six metrics that are commonly used for machine translation: chrF++, chrF, BLEU, Google BLEU, ROUGE, and METEOR. Implementation details and descriptions of these metrics can be found at [7]. We compute these metrics using translation outputs from the model as predictions and our paired English text as references.

- **chrF++** (character-level F-score): uses character n-grams in addition to word n-grams in order to gather granular and macro translation information.
- **chrF** (character-level F-score): uses character n-grams instead of word n-grams, calculates F-score instead of precision.
- **BLEU** (BiLingual Evaluation Understudy): the classic metric, evaluates precision of n-grams and punishes short translations with a brevity penalty.
- **Google BLEU**: a score derived from BLEU that uses both precision and recall of n-grams.
- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation): a set of metrics that use recall with unigrams (ROUGE-1), bigrams (ROUGE-2) or longest common subsequence (ROUGE-L or ROUGE-Lsum)
- **METEOR** (Metric for Evaluation of Translation with Explicit ORdering): uses more flexible matching with exact words, stems, and synonyms, calculates F-score and penalizes fragmentation.

We provide results with all of these metrics, but we subjectively choose to give special weight to the chrF++ score as we believe it is most fit to take morphology into account for a machine translation task, which is beneficial for our task.

5.3 Experimental details.

To our knowledge, there are no papers that perform machine translation of Obolo to English. Therefore, we conduct an investigation of the effectiveness of our new Obolo tokenizers by performing tests on several architectures and ablations, implementing baselines for comparisons using standard methods and tokenizers.

Experiment 1: LLM. In our first experiment, we perform inference (without fine-tuning) on a popular open-source large language model, Llama3-8B-Instruct, as a baseline for Obolo to English translation.

We provided the following system context for translation to English:

```
"You are an expert translator in LANG and English. You will simply
translate the given line from LANG into English. Your answer will start
with a [BEG] tag and end with a [END] tag. Do not repeat the question or
provide any other text that is not the translation of the provided text."
```

where LANG was set to 'Obolo' and 'French', respectively. We then continued to chat with the model, providing a single line to translate to English and processing the output to retrieve the translation.

Experiment 2: Untrained Transformer. As a second baseline, we also evaluate BPE tokenization and detokenization of Obolo to English translation when combined with a simple Transformer trained solely on our Bible test set. The model has multi-headed attention with 8 heads, and 3 layers for the encoder and decoder. We trained for up to 50 epochs on the training set using Adam with a default set of parameters being a learning rate of 0.0001, betas being 0.9, 0.98, and epsilon of 10^{-9}.

On the train data, our custom tokenizer had a vocab size of 7150, so for many experiments we restricted the vocab size of the traditional tokenizer baseline to 7150 tokens as well prior to training our Transformer model.

Experiments 3 and 4: Pretrained Transformers. In our experimental study, the third and fourth baselines involve the evaluation of two pretrained models: T5 and mbart. Both models belong to the sequence-to-sequence transformer family, designed for complex language processing tasks. T5, or Text-to-Text Transfer Transformer, has a diverse training background, having been trained not only on translation tasks but also on a variety of other natural language processing tasks. Specifically, for translation, it has been trained on three language pairs: English to German, English to French, and English to Romanian. This extensive and varied training regimen equips T5 with a robust ability to handle text transformations across different contexts and languages, making it a versatile tool in our analysis. However, there are also some notable drawbacks to T5, namely, that it is not exclusively trained on machine translation, and the machine translation tasks it is trained on are relatively limited.

Conversely, mbart is more specialized compared to T5, focusing exclusively on machine translation tasks. It is pretrained on a broad spectrum of 25 languages, demonstrating its capacity for handling multilingual translation scenarios. Unlike T5, which utilizes the sentencepiece tokenizer as its default method for handling textual inputs, mbart is designed to accommodate different tokenizers tailored to the linguistic structures of various languages included in its training. This feature allows mbart to adapt more precisely to the syntactic and morphological nuances of each language it processes, potentially offering more accurate translations in a multilingual context. The distinctions in training focus and tokenization strategies between T5 and mbart highlight their differing approaches to solving translation tasks, providing a rich basis for comparison in our study.

Ablations with Obolo Tokenizer. With all baselines (except the first experiment, with which finetuning required too much memory and compute for the scope of our project), we constructed a version of each model that utilized our custom built Obolo tokenizer to see if training with a specially designed tokenizer might have a positive effect on the quality of translation. Specifically, for the simple Transformer model, we swapped out the BPE tokenizer for Obolo input to our custom tokenizer

for the Obolo input text for Obolo to English translation, and swapped out BPE detokenization for Obolo custom detokenization when dealing with English to Obolo translation.

5.4 Results

Detailed metric results are shown in the tables in the Appendix A. The baseline Llama3-8B-Instruct results shown in Table 3 verify that state-of-the-art LLMs have very little knowledge of Obolo and have trouble translating Obolo to English, much more than translation with high-resource languages such as French to English.

Obolo → English Models	**chrF++**
BPE Tokenizer Large (vocab size=17596)	39.473
BPE Tokenizer Small (vocab size=7150)	39.838
Wordpiece Tokenizer (vocab size=7150)	39.548
Custom Obolo Tokenizer (vocab size=7150)	38.499

English → Obolo Models	**chrF++**
BPE Detokenizer (vocab size=7150)	36.190
Custom Obolo Detokenizer	35.147

Table 1: chrF++ scores for Obolo to English and English to Obolo Transformer NMT models, trained using different tokenization or detokenizaiton schemes

Moving on to our tokenizer comparison, we tested our Transformer model on 2610 verse pairs at several different checkpoints of training. We report the best test result for each model over these checkpoints in terms of chrF++ score in Table 1. Results of many experiments and variations can be found in Appendix A. The main result, comparing our custom tokenizer and detokenizer implementations to simply using a BPE tokenizer trained on our Obolo training data, show that our implementations are worse than BPE. For Obolo to English translation our approach has a chrF++ score of 38.499 compared to the baseline of 39.838, and for English to Obolo translation our approach has a chrF++ score of 35.147 compared to the baseline of 36.190. An interesting observation to note We see that the restriction of BPE tokenization to 7150 tokens rather than letting the BPE tokenizer train with an unrestricted vocab size (17596) does not affect chrF++ score significantly, as seen in Table 4 versus Table 5. This could mean that the unrestricted BPE tokenizer creates a lot of unnecessary tokens and there may be further use cases to be explored showing that tokenizer with reduced vocab sizes can recreate if not exceed results of tokenizers with unrestricted vocab sizes. Another surprising result was when we trained a larger Transformer model, with 6 encoder and decoder layers and 16 attention heads instead of 8, we had significantly worse performance. This could be an example of an overly large model overfitting due to our extremely small dataset.

These results are worse than expected as we believed that our custom Obolo tokenization approach that has specific knowledge about the morphology of the language would allow for significantly superior machine translation compared to using BPE which does not have this detailed morphological information. Possible reasons for this unexpected result include the folloowing: (1) tokenization does not have that much of an impact given a task with such low amounts of data and/or (2) BPE, through its automated supervised learning approach, is actually quite good at creating tokens for the machine translation task. These are good questions to explore in the future.

Unfortunately, the pretraining-finetuning paradigm that is often seen in machine learning today did not yield substantial improvements. In fact, these models did worse than the transformers we trained from scratch on the English-Obolo paired dataset. This could be due to a number of reasons. First, the pretraining might have actually hurt the model. Upon introducing a large number of Obolo tokens into the default tokenizer, the way English was tokenized was substantially changed, resulting in essentially useless embeddings for each token. On the other hand, the custom tokenizer essentially could not utilize the pretrained weights well. Additionally, another factor is that we did not perform substantial hyperparameter tuning of either T5 or mBART. Considering that we performed many iterations on our default transformer and tuned the architecture and training hyperparameters for Obolo machine translation, there may have been substantial performance gains in those changes that may be absent in the below numbers.

6 Analysis

Given these results, we propose a few hypotheses to explain the superior performance of BPE in comparison to our custom tokenizer. One possible explanation is that it's more effective to

Obolo → English Models	chrF++
T5, default tokenizer	20.245
T5, Custom Obolo Tokenizer	19.853
mBART, default tokenizer	22.754
mBART, Custom Obolo Tokenizer	21.124

Table 2: chrF++ scores for Obolo to English models using different pretrained schemes.

understand the most frequent forms of common verbs well than to generalize to rare forms of less common roots. Although our tokenization should allow for greater generalization, it can obscure simple translations of common conjugations by, for example, turning what could've been a one-to-one relationship (e.g., Obolo token to English token), into a many-to-one relationship. In other words, the added verbal information and the flexibility it allows for may be inconsequential compared to more "rote" reasoning it obstructs. Consider that if a single conjugation of a verbal root accounts for a very high percentage of the occurrences of that verb in the data, learning only that form could be at least as effective—if not more—than accounting for a diverse set of rare conjugations.

This problem could be compounded by the small size of the dataset and the specific distribution of verb forms found in the Bible.

In Appendix A we include a single pair of Obolo and English sentences, the Obolo sentence's BPE tokenization and custom tokenization, and the English translation output by the two corresponding models. Here, we see that the custom tokenizer still divides some non-verbs into verbs (e.g., "ema", meaning "they"), but also successfully breaks down some verbs (e.g., "iwut" into "i-" and "wut"). Both translations are poor, but the translation using the custom tokenizer is more repetitive and uses more generic words. 5

7 Conclusion

In this paper, we created a custom, morphologically-based tokenizer and detokenizer for an extremely low-resource language, Obolo, from scratch. We compared the effectiveness at machine translation with a small Transformer model and large pretrained NMT models using BPE tokenizers only and those same models using our custom tokenizer. We found that BPE tokenization and our custom tokenization provide very similar results, with BPE slightly edging out. We also found that our approach to finetuning large NMT models provided significantly worse results compared to training a new Transformer from scratch. There are many limitations that we did not have the time or ability to fully delve deep in for this paper. One limitation is that our dataset was quite small due to the low-resource nature of Obolo. A morphologically-motivated tokenization scheme may have more promising results when given larger amounts of data, as has already been seen with other language modeling tasks in [1]. Additionally, there were some more features and edge cases that we could have added to our custom tokenizer but did not have the time for. For example, the most recently tested version of the tokenizer greedily pulls the prefix "m-" from the word "matumu," which should in fact be tokenized as "ma-"+"tumu".

Avenues for future work in this area include, given the success of other morphological tokenizers, attempting similar strategies on a language with more data, or, given the specific architecture of KinyaBERT, building two-layered encoding with part-of-speech tagging.

It may also help to decompose morphemes into their semantic components: a morpheme may encode multiple, segmentable pieces of meaning, (e.g., 3rd-person plural future), and a single morpheme may occur in many forms, called "allomorphs" (e.g., the definite future prefix "bV-" may take the form "ba," "bo," or "be," depending on the subject's person and number). The tokenizer could potentially be improved by decomposing affixes into their pieces of meaning and removing the non-meaningful distinction between allomorphs. This, however, would require a more careful analysis of Obolo verb forms, and a significantly more complicated detokenization process.

8 Ethics Statement

One ethical concern this project faces is the bias that results from training on Biblical data alone. Because the Bible is a relatively small resource and restricted to a specific domain, it is inevitably a biased sample of language use. For example, the Bible is filled with Christian ideas, traditions, and philosophies, is primarily concerned with a small area of the world, and mentions men much more often than women. Training on the Bible bakes all of these biases into the model's knowledge base, and thus most likely spreads them to its users. Without other parallel text in Obolo, it's hard to think of a neat solution to this problem. Certainly, if more parallel text becomes available, it would be desirable to incorporate it into training. A second approach is to use a pre-trained model with broader sources of information rather than transformer trained only on Obolo/English. However, in our experiment, a simple transformer significantly outperformed a pre-trained model with fine-tuning.

Another concern is simply that we are attempting to introduce language models—which are tied to a number of ethical controversies—to a new community. Some particularly relevant concerns include interference with the work of human translators (people spent decades working on the Obolo Bible!) and outputting a high volume of low-quality Obolo text, resulting in misinformation and misrepresentation of the language. The former can be addressed by contacting organizations involved in translation (like the Obolo Language Bible Translation Organization) and asking for insight, and the latter mitigated by clearly labelling translated text.

References

[1] Haris Jabbar. Morphpiece : A linguistic tokenizer for large language models, 2024.

[2] Zobia Rehman, Waqas Anwar, Usama Ijaz Bajwa, Wang Xuan, and Zhou Chaoying. Morpheme matching based text tokenization for a scarce resourced language. *PLoS ONE*, 8(8):e68178, August 2013.

[3] Antoine Nzeyimana and Andre Niyongabo Rubungo. Kinyabert: a morphology-aware kinyarwanda language model. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022.

[4] Uche Aaron. *Tense and Aspect in Obolo Grammar and Discourse*. SIL International, 1999.

[5] Obolo Language Bible Translation Organization. Ikpa mbuban (obolo). `https://www.bible.com/bible/1587`, 2014. [Accessed 24-05-2024].

[6] Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[7] evaluate-metric (Evaluate Metric) — huggingface.co. `https://huggingface.co/evaluate-metric`. [Accessed 24-05-2024].

A Appendix

Metrics	Obolo → English	French → English
BLEU	0.000	0.236
Google BLEU	0.031	0.271
chrf	16.11	48.68
ROUGE-1	0.101	0.565
ROUGE-2	0.008	0.337
ROUGE-L	0.083	0.531
ROUGE-Lsum	0.083	0.532
METEOR	0.108	0.528

Table 3: Evaluation of Llama3-8B-Instruct translation on 200 bible verses for Obolo versus 200 lines from the OPUS Books dataset for French.

Metrics	10 Epochs	20 Epochs	30 Epochs	40 Epochs	50 Epochs
chrF++	33.980	37.445	37.881	39.344	39.473
chrF	34.774	38.380	38.926	40.417	40.722
BLEU	0.110	0.150	0.170	0.166	0.159
Google BLEU	0.161	0.195	0.205	0.207	0.202
ROUGE-1	0.389	0.434	0.450	0.449	0.444
ROUGE-2	0.152	0.188	0.201	0.205	0.200
ROUGE-L	0.328	0.366	0.378	0.378	0.370
ROUGE-Lsum	0.328	0.365	0.378	0.378	0.370
METEOR	0.335	0.372	0.374	0.386	0.381

Table 4: Evaluation of **Obolo** → **English** translation using Transformer model with Obolo BPE tokenizer with unrestricted vocab size (vocab size = 17596) with varying epochs of training

Metrics	10 Epochs	20 Epochs	30 Epochs	40 Epochs	50 Epochs
chrF++	33.716	36.647	39.146	39.572	39.838
chrF	34.476	37.503	40.205	40.648	41.042
BLEU	0.110	0.154	0.156	0.164	0.160
Google BLEU	0.161	0.195	0.200	0.206	0.203
ROUGE-1	0.386	0.433	0.445	0.452	0.446
ROUGE-2	0.150	0.188	0.200	0.207	0.203
ROUGE-L	0.325	0.367	0.374	0.381	0.374
ROUGE-Lsum	0.325	0.367	0.374	0.381	0.374
METEOR	0.334	0.368	0.384	0.388	0.387

Table 5: Evaluation of **Obolo** → **English** translation using Transformer model with Obolo BPE tokenizer with restricted vocab size (vocab size = 7150) with varying epochs of training

Metrics	10 Epochs	20 Epochs	30 Epochs	40 Epochs	50 Epochs
chrF++	33.149	36.586	38.439	38.499	37.874
chrF	33.945	37.602	39.597	39.708	39.216
BLEU	0.106	0.138	0.140	0.138	0.137
Google BLEU	0.156	0.183	0.186	0.184	0.182
ROUGE-1	0.373	0.412	0.421	0.419	0.412
ROUGE-2	0.143	0.175	0.0.181	0.182	0.176
ROUGE-L	0.316	0.350	0.353	0.350	0.345
ROUGE-Lsum	0.316	0.349	0.353	0.350	0.344
METEOR	0.323	0.356	0.371	0.366	0.353

Table 6: Evaluation of **Obolo** → **English** translation using Transformer model with our custom tokenizer with varying epochs of training

Metrics	10 Epochs	20 Epochs	30 Epochs	40 Epochs	50 Epochs
chrF++	33.808	36.904	38.909	39.548	39.505
chrF	34.578	37.768	39.981	40.630	40.732
BLEU	0.121	0.164	0.166	0.173	0.171
Google BLEU	0.172	0.204	0.208	0.214	0.211
ROUGE-1	0.402	0.447	0.450	0.458	0.454
ROUGE-2	0.158	0.199	0.205	0.214	0.208
ROUGE-L	0.338	0.378	0.380	0.388	0.381
ROUGE-Lsum	0.338	0.378	0.380	0.388	0.381
METEOR	0.332	0.370	0.379	0.385	0.378

Table 7: Evaluation of **Obolo** → **English** translation using Transformer model with Obolo Wordpiece tokenizer with restricted vocab size (vocab size = 7150) and English BPE detokenizer, with varying epochs of training

Metrics	10 Epochs	20 Epochs	30 Epochs	40 Epochs	50 Epochs
chrF++	31.459	34.609	35.865	36.190	35.646
chrF	33.118	36.453	37.871	38.377	37.757
BLEU	0.097	0.124	0.131	0.130	0.126
Google BLEU	0.149	0.170	0.176	0.175	0.172
ROUGE-1	0.402	0.426	0.433	0.440	0.431
ROUGE-2	0.169	0.197	0.203	0.208	0.200
ROUGE-L	0.330	0.351	0.356	0.364	0.354
ROUGE-Lsum	0.330	0.351	0.356	0.364	0.355
METEOR	0.275	0.299	0.306	0.307	0.201

Table 8: Evaluation of **English** → **Obolo** translation using Transformer model with English BPE tokenizer and restricted Obolo BPE detokenizer (vocab size = 7150), with varying epochs of training

Metrics	10 Epochs	20 Epochs	30 Epochs	40 Epochs	50 Epochs
chrF++	30.516	33.707	34.767	35.147	35.114

```
obolo_sentence = "otutuuk ebi ìkikpukpo owu kubọk arọọñ cha, efet môben ema otutuuk iwut ije,
                  ebi òma kwuñ mêniñ usuñ akọñ; mgbọ ya, esip mè ochak môre owu lek, bak me lek otutuuk inu ikpak kwuñ."

english_sentence = "the wind shall eat up all thy pastors, and thy lovers shall go into captivity:
                    surely then shalt thou be ashamed and confounded for all thy wickedness."


BPE Tokenizer Tokenized Input:
['ġotutuuk', 'ġebi', 'ġi', 'ỊÇ', 'kikpukpo', 'ġowu', 'ġkubá»ịk', 'ġará»ịá»ịn', 'ỊĦ', 'ġcha', ',', 'ġefet',
 'ġmo', 'ỊÂ', 'ben', 'ġema', 'ġotutuuk', 'ġiwut', 'ġije', ',', 'ġebi', 'ġo', 'ỊÇ', 'ma', 'ġkwun', 'ỊĦ', 'ġme',
 'ỊÂ', 'nin', 'ỊĦ', 'ġusun', 'ỊĦ', 'ġaká»ịn', 'ỊĦ;', 'ġmgbá»ị', 'ġya', ',', 'ġesip', 'ġme', 'ỊÇ', 'ġochak',
 'ġmo', 'ỊÂ', 're', 'ġowu', 'ġlek', ',', 'ġbak', 'ġme', 'ġlek', 'ġotutuuk', 'ġinu', 'ġikpak', 'ġkwun', 'ỊĦ.']


BPE Tokenizer Model Translation:
"they that [are] all thy pasture shall go over against thee, [and] all the wind shall go, and they
shall be brought to thee; they shall fall by thy wickedness, because of thy wickedness [shall be] for all thine iniquities."


Custom Tokenizer Tokenized Input:
['otutuuk', 'e', 'bi', 'ìkikpukpo', 'owu', 'kubọk', 'arọọñ', 'cha', ',', 'e', 'fet',
 'môben', 'e', 'ma', 'otutuuk', 'i', 'wut', 'i', 'je', ',', 'e', 'bi', 'o', '', 'ma', 'kwuñ',
 'mêniñ', 'usuñ', 'akọñ', ';', 'mgbọ', 'ya', ',', 'e', 'sip', 'mè', 'o', 'chak',
'môre', 'owu', 'lek', ',', 'bak', 'me', 'lek', 'otutuuk', 'inu', 'ikpak', 'kwuñ', '.']

Custom Tokenizer Model Translation:
"and all the feed them that feed thee do feed [them] among them that are set [them] among them that are set [them]
 are come over against thee, [and] they that bew with thee shall be put to shame for all thy wickedness:"
```

Figure 5: Tokenization and translation of example Obolo sentence with BPE and custom tokenizer Transformer models