Tokenization Techniques for Optimizing Large Language Models across Tasks

Jonas Endriss and **Mohammad Zain Farooq Gill** and **Yow Siao Kang** and **Orgil Dorj**
Korean Advanced Institute of Sciece and Technology (KAIST)

Abstract

Tokenization is a crucial yet often overlooked step in the field of Natural Language Processing (NLP). This study investigates the impact of three different tokenization techniques, namely Unigram, Byte-Pair-Encoding (BPE), WordPiece. For our small scale project, we have used a scaled-down version of the BERT model, which we have named MicroBERT. Three different NLP tasks were examined, namely Text Generation, Question Answering, and Named Entity Recognition (NER). We have used evaluation metrics such as perplexity, Exact Match, and F1-score to measure performance. Our results have shown that BPE due to its balanced granularity performs consistently well. Unigram excels in Entity Recognition tasks, possibly due to its probabilistic approach. WordPiece struggles with fine-grained tokenization, which resulted in subpar performance for our tasks. These findings illustrate the importance of selecting appropriate tokenization techniques for different NLP tasks for small-scale models. Future research into larger-scale models and more complex tasks is needed.

1 Introduction

Tokenization, or the process of breaking text into smaller pieces called tokens, plays a key role in how computers process and understand language. It's the first step in helping machines make sense of human communication. However, the way text is tokenized can greatly influence the performance of language models in different tasks.

In this study, we explore how three popular tokenization methods, namely Unigram, Byte Pair Encoding (BPE), and WordPiece, impact the performance of our small-scale BERT model called MicroBERT. We tested these methods on three important tasks: generating text, answering questions, and identifying key names or terms in sentences (Named Entity Recognition). These tasks were chosen because they show how tokenization affects different types of real-world applications.

Although researchers have shown that subword-based tokenization methods are often better than character-based ones, there is not much guidance on which method works best for specific tasks, especially when working with smaller models. Our goal is to close this gap by comparing the strengths and weaknesses of these techniques. By doing so, we aim to offer practical advice for improving how computers process language while also deepening our understanding of tokenization.

2 Literature Review

Tokenization is one of the most important preprocessing steps in LLMs, which can affect the performance of the model significantly on a wide range of tasks. This review aims to discuss insights from the literature on how different tokenization techniques impact model efficiency and task-specific optimization. Key studies highlight that tokenization is not a universal process and must be tailored to the needs of specific tasks and data characteristics.

One of the most interesting papers, "Toward a Theory of Tokenization in LLMs"(1), gives a theoretical framework on how tokenization interacts with downstream tasks. It develops a notion of alignment between tokenization strategies and task requirements; some methods that make text summarization more efficient can hurt translation or sentiment analysis. Similarly, the paper "Effects of Tokenization on Transformers for Biological Sequences"(2) investigates tokenization in the biological domain, such as in protein sequences, and establishes that techniques like BPE do reduce the input length but at the cost of introducing sparsity issues, thus demanding domain-specific solutions.

Tokenization impacts domain-specific languages, as discussed in "Impact of Tokenization on Language Models: An Analysis for Turkish"(**?**), ad-

dresses challenges in tokenizing morphologically rich languages. It shows that the subword tokenizers perform better than character-level methods in semantic tasks, again underlining the importance of linguistic considerations during tokenizer design. Other works, such as "An Information Extraction Study: Take in Mind the Tokenization!"(**?**), reveal task-specific improvements like coarse-grained approaches are better for relation extraction.

"What Makes for Good Tokenizers in Vision Transformers?"(**?**) develops general principles of tokenization, focusing on information density and the minimization of redundancy. Though the paper is mainly concerned with computer vision, its principles undergird many tokenization strategies for LLMs. Further work, such as "How BPE Affects Memorization in Transformers"(**?**), points out the dangers of overfitting to frequent n-grams using BPE tokenization and advocates for adaptive methods that balance memorization and generalization.

Innovative approaches include the "LICHEE" framework(**?**), which integrates multi-grained tokenization to enhance LLM representation capabilities, and the "Efficient Domain Adaptation via Adaptive Tokenization"(**?**), which optimizes pretrained models for new domains with domain-specific subword sequences. Moreover, "Flexibly Scaling Contexts Through Extensible Tokenization"(**?**) offers methods to extend LLM context windows without substantial retraining, enabling better handling of long-context tasks.

These collectively confirm the multifaceted impact of tokenization on LLM performance, including task dependencies, domain sensitivities, input length-efficiency trade-offs, and risks of overfitting. While progress is being made, a unified framework for the systematic assessment of tokenization techniques across tasks remains a gap in the literature. This project aims to fill this gap by testing the effectiveness of tokenization strategies and, in turn, increasing knowledge and improving LLMs as a whole.

3 Objectives

The goal of this study is to understand how different tokenization methods affect the performance of language models across various tasks. To achieve this, we focus on three main objectives:

3.1 Compare Tokenization Methods

This study aims to explore how the way we break down text—known as tokenization—affects the ability of language models to perform different tasks. To do this, we have set three clear objectives:

3.2 Compare Tokenization Methods

We're focusing on three popular techniques—Unigram, Byte Pair Encoding (BPE), and WordPiece. By comparing how each one works, we want to understand their strengths and weaknesses.

3.3 Test Performance on Key Tasks

To see how these methods perform, we have chosen three common NLP tasks.

1. Text Generation - Checking how well the model can write natural, meaningful sentences.

2. Question Answering - Measuring how accurately the model can answer questions based on given text.

3. Named Entity Recognition (NER) - Seeing how effectively the model can pick out important names, places, or terms in a sentence.

4 Methodology

4.1 Base Dataset Collection

Publicly available datasets, including BookCorpus and Wikipedia, were used as the basis for training. These datasets provided comprehensive textual data for training our base model and tokenizers.

4.2 Training Tokenizers

We trained three tokenizers, namely Unigram, Byte Pair Encoding (BPE), and WordPiece, using the same base dataset as our BERT model. This ensured consistency between the tokenization process and the model's input requirements.

4.3 Training the Model with Different Tokenizers

Since we had general computational, memory, storage and time constraints, we have used a simple small-scale version of BERT using the Hugging Face library which we have named MicroBERT. This model has 2 transformer layers, 128 hidden dimensions, and 4 attention heads, vocabulary size of 30,000, and other adjustments to sequence length and other parameters to balance computational efficiency and performance.

The MicroBERT model was trained multiple times using the base dataset and each of the three tokenization methods. This allowed us to analyze how tokenization impacts the model's ability to learn during pretraining.

4.4 Fine-Tuning on Task-Specific Datasets

After pretraining, the models were fine-tuned on specific NLP tasks, such as text generation, question answering, and named entity recognition. Task-specific datasets were used during this phase to evaluate real-world applications of the tokenization methods.

4.5 Evaluation

The performance of the model was assessed using established metrics:

1. Text Generation: Perplexity.
2. Question Answering: Exact Match and F1 scores.
3. Named Entity Recognition: Precision, Recall, and F1 scores.

By comparing the results across tasks, we identified the strengths and weaknesses of each tokenization technique.

5 Dataset and Evaluation

5.1 Base Dataset

For our base dataset for pretraining our model, we have used data from 2 different sources.

5.1.1 BookCorpus

BookCorpus, a collection of fictional works, was used to create a smaller subset for our study. Using the Hugging Face datasets library, we randomly selected 60% of sentences while excluding those shorter than 20 characters to maintain quality. The final subset, around 3 GB in size, was saved as a plain text file and suited our storage and memory limits.

5.1.2 Wikipedia

We used the March 2022 English Wikipedia dump to create a subset of structured encyclopedic text. Articles were processed into 128-word chunks, with 40% of these chunks randomly selected for inclusion. Invalid or overly short sentences were excluded, and efficient memory management ensured smooth processing. The final subset was approximately 7 GB and saved as a plain text file.

6 Fine-Tuning

6.1 Text Generation

For text generation, we have used the WritingPrompts dataset, sourced from Hugging Face's euclaise/writingprompts. This dataset contains creative writing prompts paired with human-written responses. This dataset was ideal for evaluating the model's ability to generate coherent and meaningful text, as it required the model to construct contextually relevant and natural-sounding sequences.

For evaluation metric, we have chosen the perplexity score. Perplexity measures how well a language model predicts the next word in a sequence, with lower values indicating better performance. We can see the end result in figure 1.

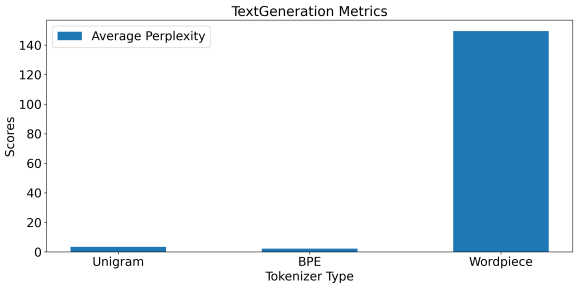

Figure 1

The results were as follows: Unigram Tokenizer: 3.3, BPE Tokenizer: 2.24 (best performance), WordPiece Tokenizer: 149.47.

The results demonstrate that the BPE tokenizer significantly outperformed the others, achieving the lowest perplexity. In contrast, the WordPiece tokenizer struggled with this task. This is likely due to its fine-grained tokenization approach which breaks words into many smaller pieces, leading to loss of context and coherence.

6.2 Question Answering

For question answering, we have used the SQuAD (Stanford Question Answering Dataset) dataset sourced from Hugging Face's squad. This dataset contains passages paired with questions and corresponding answers, designed to test the model's ability to locate answers within a given context.

For evaluation metrics, we have chosen to use the Exact Match (EM) and F1-Score. The Exact Match (EM) metric demonstrates the percentage of questions where the model's predicted answer matches the ground truth exactly. F1-Score, on the other hand, illustrates the harmonic mean of precision and recall, reflecting partial overlaps between

predicted and true answers. The end results can be seen in figure 2 and were as follows: Unigram Tokenizer: EM = 0.0136, F1 = 0.0145, BPE Tokenizer: EM = 0.0185, F1 = 0.0209 (best), WordPiece Tokenizer: EM = 0.0, F1 = 0.0.

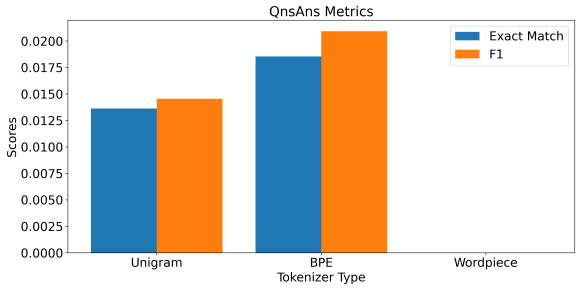

Figure 2

Here the BPE tokenizer slightly outperforms the Unigram tokenizer. Meanwhile, Wordpiece tokenizer failed to perform this task. This is likely due to its overly fine-grained approach which leads to a lack of coherence. The BPE tokenizer's slightly better performance in question answering could be due to its ability to balance between capturing subword information and maintaining context, which is crucial for understanding and answering questions accurately.

6.3 Named Entity Recognition (NER)

For named entity recognition, we have used the CoNLL-2003 dataset sourced from Hugging Face's conll2003. This is a standard dataset for such tasks, containing labeled examples for recognizing entities such as people, organizations, locations, and more.

For evaluation metrics, we have used three different metrics: Precision, Recall, and F1-Score. Precision measures the percentage of correctly predicted entities out of all predicted entities, indicating how many of the model's predictions were accurate. Recall measures the percentage of correctly predicted entities out of all true entities in the dataset, showing how well the model identified all relevant entities. F1-Score provides a balanced measure by combining precision and recall, offering a harmonic mean that accounts for both false positives and false negatives. The results of these evaluations can be seen in figure 3.

The results were as follows: Unigram Tokenizer: Precision = 0.3924, Recall = 0.4720, F1 = 0.4285 (best), BPE Tokenizer: Precision = 0.3498, Recall = 0.4579, F1 = 0.3966, WordPiece Tokenizer: Precision = 0.3124, Recall = 0.3703, F1 = 0.3389.

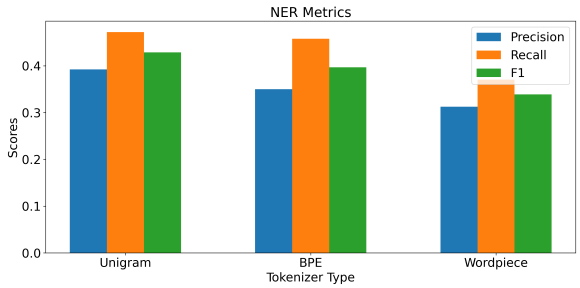

Figure 3

The Unigram tokenizer's superior performance in NER might be attributed to its probabilistic approach, which can better capture the variability in entity boundaries compared to the more rigid splitting of the Wordpiece tokenizer.

7 Conclusion and Future Work

In summary, the differences in performance across tasks can be explained by their inherent characteristics of each tokenizer. The BPE tokenizer has a good balance between character and word-level tokenization which makes it versatile, while the Unigram tokenizer's flexibility helps in tasks like NER. The Wordpiece tokenizer's overly fine-grained approach seems to hinder its performance across the board.

Here, it is important to keep in mind that small model size limits the ability to understand context, capture complex patterns, create detailed representations, generalize to new data, handle long sequences, and efficiently use the attention mechanism. These limitations collectively contribute to the poor performance of our models in all tasks.

Future work could investigate the impacts of tokenization techniques on larger-scale models and extend the analysis to more complex NLP tasks, which would help refine the findings and explore the broader applications of different tokenization methods.

References

[1] Toward a Theory of Tokenization in LLMs

[2] Effects of Tokenization on Transformers for Biological Sequences

[3] Impact of Tokenization on Language Models: An Analysis for Turkish

[4] An Information Extraction Study: Take in Mind the Tokenization!

[5] What Makes for Good Tokenizers in Vision Transformers?

[6] How BPE Affects Memorization in Transformers

[7] LICHEE

[8] Efficient Domain Adaptation via Adaptive Tokenization

[9] Flexibly Scaling Contexts Through Extensible Tokenization