Hack the Tokenizer

Luis Pinto

Master’s degree in Computer Science
Department of Computer Sciences
Faculty of Sciences of the University of Porto
2025

Hack the Tokenizer

Luis Pinto
Dissertation carried out as part of the
Master’s degree in Computer Science
Department of Computer Sciences
2025

Supervisor
Alípio Jorge, Professor, Department of Computer Sciences, Faculty of Sciences, University of Porto

External Supervisor
Hugo Sousa, PhD. Student, Department of Computer Sciences, Faculty of Sciences, University of Porto

Acknowledgements

The work presented in this dissertation was only made possible thanks to the support, guidance, and encouragement I received from many people. Although it would be impossible to thank everyone individually, I would like to express my gratitude to several groups and individuals in particular.

First and foremost, I owe my deepest appreciation to my supervisor, Prof. Alípio Mário Guedes Jorge, for his guidance, availability, and encouragement throughout this project. I also wish to sincerely thank my supervisor, MsC. Hugo Sousa, for his invaluable knowledge, advice, and unwavering support whenever I felt stuck – which happened more often than I care to admit. I may not have been the easiest of students to deal with, but both of you remained patient and committed, and without your help this dissertation would not have been possible.

I am profoundly grateful to my family – my parents, my brother, and my grandparents – for their endless love, patience, and support. Even when my time with them was shortened due to the demands of my personal life, they always stood by me, encouraging me and giving me the strength to continue.

To my work colleagues, I want to express my appreciation for the flexibility and understanding you offered, which allowed me to balance both professional responsibilities and the challenges of completing my master's degree. Your support was fundamental in making this possible.

To my friends, I am truly thankful for your patience and encouragement during the difficult moments of this journey. Some of you witnessed my struggles up close, others from afar, but all of you gave me strength. A special thank you goes to Hugo Valdrez, who has been by my side throughout the entire master's journey – your companionship throughout this journey means more than words can say.

Finally, to my partner in crime, Susana: thank you for your kindness, motivation, and unwavering support through the most challenging times. Your advice, patience, and understanding – especially during the countless weekends I could not be by your side – carried me through this work. This dissertation would not have been possible without you.

Resumo

Os modelos de linguagem de grande porte ("Large Language Models") são normalmente treinados em línguas com muitos recursos, deixando as de poucos recursos em desvantagem. Esta dissertação explora estratégias eficientes para adaptar modelos treinados em inglês ao português europeu sem treinar novamente todo o modelo. A ideia central, denominada "hacking the tokenizer", envolve estender seletivamente o vocabulário do tokenizador ("tokenizer") e a camada de *embedding* ("embedding layer") do modelo, permitindo-lhe representar o texto português de forma mais compacta, minimizando o custo computacional.

Propomos e avaliámos várias estratégias de inicialização de incorporação, introduzindo a "Inicialização Ponderada por Posição" ("Position-Weighted Initialization") como um método inovador que superou outros métodos comparados. Para garantir a usabilidade do vocabulário expandido, também desenvolvemos uma estrutura de adaptação de inferência [1] ("Inference Adaptation") que preserva a compatibilidade com pesos pré-treinados ("pre-trained weights"). Além da metodologia, introduzimos novas métricas de avaliação – *"Fertility Boost"*, *"Fertility Output"* e *"Effective Efficiency Gain"* – para capturar melhor os ganhos de eficiência da relação entre tokenização e inferência.

Experiências realizadas em benchmarks de português europeu (*extraGLUE* e *CalamePT*) mostram que a adaptação do tokenizador não tem impacto significativo na qualidade do texto gerado, mas pode melhorar a eficiência da geração ("Generation Efficiency"), reduzindo a contagem de tokens em aproximadamente 18%, com os modelos monolíngua de menor tamanho sendo os mais beneficiados. É importante destacar que a adaptação não prejudica o desempenho em inglês, destacando a viabilidade dessa estratégia leve de expansão multilíngua.

Embora esses resultados sejam encorajadores, eles são modestos em escala: intervenções no nível do tokenizador não são apresentadas como um substituto completo para o ajuste fino ("fine-tuning"). Este trabalho demonstra que a adaptação do tokenizador pode ser um complemento útil para as técnicas de adaptação existentes. Seria necessário um maior desenvolvimento para alcançar a confiabilidade pronta para produção.

Palavras-chave: Ciências de Computadores, Aprendizagem de Máquina, Modelos de Linguagem, Tokenizador, Fertilidade, Processamento Natural de Linguagem

[1] Código Fonte - `https://github.com/LIAAD/hack_the_tokenizer`

Abstract

Large language models (LLMs) are typically trained on high-resource languages, leaving low-resourced ones at a disadvantage. This dissertation explores efficient strategies for adapting English-trained models to European Portuguese without retraining the entire model. The central idea, termed *hacking the tokenizer*, involves selectively extending the tokenizer's vocabulary and embedding layer, enabling models to represent Portuguese text more compactly in terms oftoken count, while minimizing computational cost.

We propose and evaluate multiple embedding initialization strategies, introducing *Position-Weighted Initialization* as a novel method that outperformed other methods we studied. To ensure the usability of the expanded vocabulary, we also design an inference adaptation framework [1] that preserves compatibility with pre-trained weights. In addition to methodology, we introduce new evaluation metrics – *Fertility Boost*, *Fertility Output*, and *Effective Efficiency Gain* – to better capture efficiency gains from the relationship between tokenization and inference.

Experiments conducted on European Portuguese benchmarks (*extraGLUE* and *CalamePT*) show that tokenizer adaptation has no significant impact on effectiveness. However, it can improve generation efficiency by reducing token counts by approximately 18%, with smaller monolingual models benefiting the most. Importantly, adaptation does not degrade English effectiveness, highlighting the viability of this lightweight multilingual expansion strategy.

While these results are encouraging, they are modest in scale. Tokenizer-level interventions are not a full replacement for fine-tuning as they do not increase the effectiveness of the target language. Instead, this work demonstrates that tokenizer adaptation can be a useful, lightweight complement to existing adaptation techniques. Further development, broader evaluations across diverse model architectures and sizes, and integration with parameter-efficient fine-tuning methods would be needed to reach production-ready reliability.

Keywords: Computer Sciences, Machine Learning, Large Language Models, LLM, Tokenizer, Fertility, Natural Language Processing, NLP

[1]Source code - https://github.com/LIAAD/hack_the_tokenizer

Table of Contents

List of Figures

1. Introduction

Since the release of ChatGPT[1] in November 2022, the field of Artificial Intelligence has gained mainstream attention [2] by showcasing impressive performance across a wide range of tasks [3]. These models, trained on large datasets comprising billions to trillions of words [4, 5], demand enormous computational resources.

One of the main challenges these models face is the lack of training data [6, 7] for low-resource languages, which limits their effectiveness in those languages.

The main goal of this dissertation is to explore how to adapt existing models trained primarily in one language (English) to another (European Portuguese) using minimal computational resources.

Current research mainly focuses on fine-tuning, consisting of updating millions of parameters and requiring a multitude of hours using high-quality resources [8]. Researchers have explored freezing some of a model's parameters while training others to accelerate fine-tuning [9]. However, to the best of our knowledge, no prior work has explored adapting LLMs without additional training.

Our approach takes a complementary direction by focusing on training-free adaptation through an often overlooked early stage of LLMs: the *tokenizer*. This stage serves as the first translation layer between raw text and the vector representation the model consumes. Prior work has shown that tokenizer choice and design is correlated with both effectiveness and efficiency [10, 11], making it a promising and lightweight lever for cross-lingual model adaptation.

1.1. Motivation

Developing language models for low-resource languages presents significant challenges due to the limited availability of high-quality data and the high computational cost of training or fine-tuning large-scale models [12]. Only a small fraction of the world's languages are represented in mainstream NLP resources, resulting in chronic data scarcity [13]. Systematic reviews confirm that this scarcity is the dominant bottleneck [14], and that practicable solutions typically rely on transfer learning, data augmentation, or multilingual pretraining rather than training from scratch [15].

European Portuguese illustrates this challenge. Despite being spoken by approximately 10 million native speakers [1], the availability of high-quality language models lags behind that of widely spoken languages. This disparity creates barriers to technological inclusion and limits access to advanced language technologies for Portuguese speakers, such as AI assistants for healthcare or legal domains.

In response, this dissertation explores lightweight and training-free approaches to model adaptation. Rather than relying on costly retraining which would otherwise require access to specialized hardware, we focus on *Tokenizer Adaptation* – a technique focused on modifying and updating the *tokenizer* – the component responsible for mapping raw text into discrete tokens later consumed by the model.

Original Tokenizer: 147 tokens
Os fãs de GTA estão ansiosos pelo lançamento do próximo jogo da série, que pode demorar mais alguns anos para ser lançado. Rumores sugerem que o GTA VI será uma versão moderna de Vice City e contará com um mapa que muda com o passar do tempo. Além disso, há a possibilidade de uma protagonista feminina, o que traz mais expectativa para o jogo. Enquanto aguardamos, resta imaginar o que essa nova aventura nos reserva.
Adapted Tokenizer: 120 tokens
Os fãs de GTA estão ansiosos pelo lançamento do próximo jogo da série, que pode demorar mais alguns anos para ser lançado. Rumores sugerem que o GTA VI será uma versão moderna de Vice City e contará com um mapa que muda com o passar do tempo. Além disso, há a possibilidade de uma protagonista feminina, o que traz mais expectativa para o jogo. Enquanto aguardamos, resta imaginar o que essa nova aventura nos reserva.

Table 1.1: Tokenization before and after vocabulary extension of 2,000 tokens showcases the potential benefits of exploring tokenization approaches/

As illustrated in Table 1.1, adapting an English-centric tokenizer – originally from the `HuggingFaceTB/SmolLM2-135M` model – by incorporating an additional 2,000 tokens reduces the number of tokens required to represent the same text by about 18%. This demonstrates that even modest modifications to a tokenizer can yield substantial efficiency gains, reinforcing the broader motivation for this dissertation: enabling technological inclusion without the prohibitive costs of full-scale retraining.

[1]Number of Portuguese Speakers - https://en.wikipedia.org/wiki/European_Portuguese

1.2. Objectives

The objective of this dissertation is to develop methods that expand language models trained primarily in one source language to a *target* language, while minimizing computational cost. We thoroughly explore this expansion by manipulating the tokenizer and the embedding layer of the model, without the need for intensive additional training.

We focus on addressing the following research questions:

RQ1. Can an English-trained model achieve comparable effectiveness in European Portuguese through strategic modifications of the tokenizer?
RQ2. What is the impact on model generation efficiency of tokenizer adaptation?
RQ3. Does tokenizer adaptation affect all models equally, or are there differences based on model architecture and size?
RQ4. Which embedding initialization strategies are most effective for integrating new tokens into a pre-trained model?
RQ5. Does tokenizer adaptation reduce effectiveness in the model's source language?

By structuring the research around these questions, we provide a systematic evaluation of tokenizer adaptation for multilingual expansion.

1.3. Approach

Our approach centers on modifying the tokenizer and embedding layer of pre-trained models, without editing the model parameters. This strategy reduces training costs while preserving most of the knowledge already present in the model. The methodology consists of four stages:

1. **Token Selection.** Train a new tokenizer specifically on the target language corpora, then identify high-frequency tokens that are not present in the original model's vocabulary.

2. **Vocabulary Expansion.** Extend the model's vocabulary by adding the trained tokens, creating a hybrid tokenizer that maintains compatibility with the source language while expanding coverage for the target language text.

3. **Embedding Initialization.** Develop and compare strategies for initializing the embeddings of the newly added tokens, including *Mean Vector Initialization* and *Position-Weighted Initialization*, which account for the positional importance of constituent tokens.

4. **Inference Adaptation.** Develop an inference framework that enables the model to effectively use the newly added tokens during text generation, maintaining compatibility with the model's pre-trained weights.

This approach enables efficient adaptation at minimal cost, making it particularly suitable for scenarios with limited access to high-performance computing infrastructure.

1.4. Contributions

This dissertation makes several contributions to the field of multilingual model adaptation.

1. Development of a new tokenizer adaptation methodology that enhances model generation efficiency – under sampling methods – for European Portuguese without requiring full model retraining
2. Comparative analysis of multiple embedding initialization strategies, including *Mean Vector Initialization* and *Position-Weighted Initialization*
3. Proposal of an inference adaptation framework that enables efficient utilization of adapted tokenizers while maintaining compatibility with pre-trained model weights
4. Introduction of novel evaluation metrics – *Fertility Boost*, *Fertility Output* and *Effective Efficiency Gain* – that quantify how efficiently adapted tokenizers are used during generation, bridging the gap between tokenization-level analysis and actual inference behavior.
5. Comprehensive experimental evaluation on European Portuguese-specific benchmarks, achieved through the consolidation of two existing evaluation methodologies, offering a practical foundation for future work in low-resource language adaptation

Together, these contributions provide a lightweight, computationally efficient path to extending LLMs to low-resource languages.

1.5. Document Structure

The rest of this work is organized as follows. Chapter 2 establishes the necessary background, starting with an overview of the Transformer architecture and its reliance on tokenization for language representation.

Building on this, Chapter 3 reviews related work, contrasting traditional fine-tuning approaches with more recent tokenizer-level strategies, and introducing European Portuguese benchmarks.

Chapter 4 then introduces the core methodology of tokenizer adaptation. It details how new tokens are selected, explores various strategies for embedding initialization, and presents a controlled evaluation framework.

Chapter 5 provides exploratory analyses and clarifies the design rationale behind each decision. Here, the research questions guiding the work are formalized.

Chapter 6 tackles inference adaptation. It examines the challenges that arise when extending a tokenizer without modifying the model's pre-trained weights and proposes a practical framework to ensure coherent and efficient generation with the newly introduced vocabulary.

Chapter 7 presents the results. Quantitative analyses assess embedding strategies, benchmark effectiveness, and efficiency gains, while qualitative evaluations complement these findings, offering a broader perspective on the implications of tokenizer adaptation.

Finally, Chapter 8 concludes the dissertation, synthesizing the key insights, acknowledging limitations, and outlining directions for future research.

2. Background

This chapter introduces the theoretical background necessary to understand the contributions of this work. We first provide an overview of the Transformer architecture, which underpins most modern LLMs [16]. We then discuss the challenge of representing natural language text as input for these models, tracing the evolution of tokenization algorithms and their role in large-scale language modeling.

2.1. Transformers

The task of enabling machines to process and understand natural language has long been a central challenge in artificial intelligence research [17]. Early approaches were primarily rule-based, relying on hand-crafted grammars and symbolic systems that proved brittle, unable to capture the variability and richness of human language [18]. The introduction of statistical methods and later neural models marked a turning point, allowing researchers to capture patterns of co-occurrence and semantic regularities in a data-driven manner [19].

A key milestone was the development of distributed word representations, motivated by the distributional hypothesis that words appearing in similar contexts tend to have similar meanings [17]. Instead of treating words as discrete symbols, models such as `word2vec` [20] and `GloVe` [21] learned dense vector embeddings that captured semantic similarity. These representations proved powerful in downstream tasks such as text classification [22], sentiment analysis [23], and machine translation [22]. However, these embeddings were *static* – i.e., the word "bank" had the same representation whether it referred to a riverbank or a financial institution – making them insufficient for capturing context-dependent meaning [24].

The search for context-sensitive representations led to the rise of deep neural architectures for sequence modeling. Recurrent Neural Networks (RNNs) [25] and their variants, such as LSTMs [26] and GRUs [27], introduced hidden states that evolved with the sequence, allowing models to adapt word representations based on preceding tokens. This enabled breakthroughs in tasks such as language modeling and machine translation [28]. Yet these models had inherent limitations: they struggled with long contexts due to vanishing gradients, relied on sequential computation that restricted parallelization, and became increasingly inefficient as model sizes grew [29, 30].

Convolutional architectures offered an alternative [31, 32], but they were primarily effective at modeling local patterns in text rather than capturing global semantic relationships [33]. By the late 2010s, the core challenges in NLP still remained [34]: neural networks struggled to balance scalability with robust handling of linguistic ambiguity and discrete symbolic structure.

However, these challenges were mitigated with the introduction of the Transformer architecture by Vaswani et al. [35]. The Transformer replaced recurrence and convolutions with a novel mechanism: *self-attention*. Instead of processing text sequentially, self-attention allowed the model to directly relate each token to every other token in the sequence, regardless of distance. This architecture addressed several bottlenecks simultaneously: it scaled efficiently by enabling parallel training across entire sequences [35], captured long-range dependencies more effectively than RNNs or CNNs [36], and produced contextualized word embeddings, where the representation of a word adapts dynamically to its contextual environment [37]. Together, these innovations reshaped the possibilities in NLP [38].

Despite its adoption not being instantaneous, it gained momentum as subsequent models demonstrated their capabilities at scale. Within just a few years, the architecture became the backbone of nearly all modern large-scale language models [38].The original Transformer paper has since been cited over 190,000 times (as of September 2025), and the architecture underpins influential models such as BERT [39], the GPT series [4, 40], T5 [41], and more recent open-source efforts such as LLaMA [5] and Mistral [42]. These systems collectively demonstrated that scaling Transformer architectures could yield unprecedented performance across virtually every NLP benchmark.

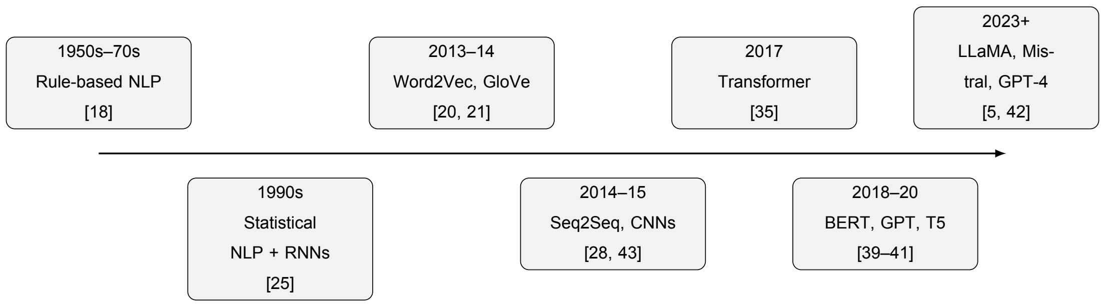

Figure 2.1: Timeline of major milestones in NLP leading to Transformer-based LLMs.

A Transformer model processes data through several conceptual stages. The input data first goes through multiple layers in the **encoder stack**. The first layer that the data goes through is the *Embedding Layer*, followed by multiple stacked layers of self-attention and feedforward networks, producing context-sensitive representations of the sequence. Finally, task-specific output layers allow the model to generate text, classify sequences, or perform other downstream tasks.

As this dissertation is primarily concerned with adapting the *Tokenizer* and *Embedding* layers for cross-lingual use, we provide a more detailed discussion of these components in the following sections.

2.1.1 Embedding

The Transformer requires inputs in the form of numerical vectors of a fixed dimension. Prior to entering the model, inputs are mapped into this vector space through an *Embedding Layer* or equivalent projection step.

Each input element is passed through an embedding matrix – a learnable parameter of the model – which maps it to a dense vector of fixed size. These vectors are referred to as *embeddings*.

They are then augmented with **positional encodings** to provide information about the order of elements in the sequence, as the transformer architecture has no inherent notion of order.

Modifying the Embedding Layer. In this dissertation, only the *Embedding Layer* is trained during the adaptation of a large language model to a new language. This is a form of lightweight fine-tuning, where the embedding matrix is modified while keeping the rest of the model frozen. This strategy is computationally efficient and has demonstrated effectiveness in low-resource or domain adaptation settings [44].

By adjusting only the *Embedding Layer*, we aim to *re-align* the model's input space without requiring full model retraining.

2.1.2 Attention Layer

During the attention layer, the model learns dependencies between input elements by computing how much each element should attend to every other element in the sequence.

Concretely, the inputs (represented as embeddings) are projected into three sets of vectors: queries, keys, and values. Attention weights are obtained by computing the dot product of queries and keys, scaling the result, and applying a softmax function to normalize the weights. These weights are then used to compute a weighted sum of the values, producing context-aware representations for each input element.

This mechanism allows the model to dynamically incorporate information from the entire input sequence at every layer, enabling it to capture complex relationships across long-context windows.

2.1.3 Decoding

Once the input elements have been converted into embeddings and processed through the self-attention and feedforward layers, the decoder generates output representations that capture contextual relationships within the input sequence.

The decoder typically consists of multiple layers, each containing a self-attention mechanism and a feedforward network. These layers iteratively refine the input representations, allowing the model to capture complex dependencies across the entire sequence.

Finally, the decoder produces a probability distribution over the model's output space (e.g., vocabulary in language tasks), which can then be used to select or generate the next element in the output sequence. This process is repeated auto-regressively for sequence generation tasks or directly applied to classification or regression.

2.2. Tokenizers

While the Transformer architecture itself operates on continuous vectors, natural language consists of discrete symbolic units (characters). Thus, to apply Transformers to text, one must first answer the question: *How can we represent text in a form suitable for Transformer models?* This question raises the problem of designing a procedure to convert raw strings into sequences of numerical identifiers.

Early solutions to this problem arose not from NLP, but from data compression research. Algorithms such as Byte Pair Encoding (BPE) [45] were originally introduced as general-purpose compression methods, designed to replace frequent symbol pairs with a more space-effective value (e.g., a single 8-bit number) but were later adapted to construct vocabularies of variable-length subword units, balancing vocabulary size and coverage.

By utilizing algorithms such as BPE, it becomes possible to map character strings into sequences of discrete tokens, which are then mapped into vectors through the *Embedding*

Layer. The list of tokens present in a tokenizer defines the "vocabulary" of the model and serves as the entry point for any language processing task.

Formally:

$$T : \Sigma^* \to V^k, \quad S \mapsto (t_1, t_2, \dots, t_k)$$

where Σ is the alphabet (e.g., Unicode characters), Σ^* is the set of all finite strings over Σ, and $V \subset \Sigma^*$ is the vocabulary.

2.2.1 Tokenization Process

The general tokenization pipeline consists of the following stages:

Algorithm 1 Generic Tokenization Process

```
1: Input: Text string $S$
2: Preprocessing (optional): lowercase, Unicode normalization, whitespace handling
3: Splitting: divide text into basic units (e.g., "cat" → [cat]; "cat" → [c, at]; "cat" → [c, a, t])
4: Mapping: assign token IDs via vocabulary lookup
5: Add special tokens: [CLS], [SEP], [PAD], etc. (if required by the model)
6: Output: Token sequence $(t_1, t_2, \dots, t_n)$
```

Subword tokenization has been the predominant choice for most recent LLMs since it balances vocabulary size and handling of out-of-vocabulary words [10, 46–48].

2.2.2 Byte-Pair-Encoding (BPE)

The BPE algorithm's original purpose – storage optimization – was achieved by iteratively replacing the most frequent pairs of bytes in a sequence with a new symbol of smaller size. This simple frequency-based merging strategy was later adapted to natural language processing by Sennrich et al. [46], where the goal was no longer to compress text, but to efficiently build a vocabulary of subword units for machine translation systems.

The key motivation for BPE in NLP is its balance between two extremes: word-level vocabularies, which suffer from out-of-vocabulary (OOV) issues, and character-level vocabularies, which lead to excessively long sequences. By iteratively merging frequent character pairs into larger units, BPE naturally captures common morphemes, prefixes, and suffixes (e.g., merging "ing" or "pre" as reusable subword units). This property makes it well-suited for handling rare words, inflected forms, and domain-specific terminology.

Illustrative Example. Consider the word "lower". With a character-level vocabulary, it is split as [`l`, `o`, `w`, `e`, `r`]. After training with BPE, frequent merges such as [`lo`] and [`wer`] might appear in the vocabulary, allowing the word to be represented as [`lo`, `wer`]. This reduces sequence length while preserving reconstructability of the original string.

Algorithm. The algorithm can be summarized as follows:

Algorithm 2 Byte-Pair Encoding (BPE)

```
1:  Input: Corpus $C$, size of target vocabulary $size$
2:  $V \leftarrow$ set of all characters in $C$
3:  $R \leftarrow \emptyset$
4:  while $|V| < size$ do
5:      $P \leftarrow$ Frequency Pairs in $C$
6:      $(A, B) \leftarrow$ most frequent pair in $P$
7:      $R \leftarrow R \cup \{(A, B)\}$
8:      $C \leftarrow replace(C, (A, B), Z)$, where $Z = concat(A, B)$
9:      $V \leftarrow V \cup \{AB\}$
10: end while
11: Return: $R$, Merge rules (e.g., ”e” + ”s” $\rightarrow$ ”es”).
```

2.2.3 WordPiece

WordPiece was first introduced in the context of statistical machine translation [49] and later became widely adopted in large-scale pre-trained models such as BERT [39]. Like BPE, it builds a subword vocabulary by iteratively merging symbols. However, instead of relying purely on frequency, WordPiece selects merges that maximize the likelihood of the training corpus under a language model. This makes it more sensitive to semantic and syntactic regularities than raw frequency counts.

The intuition behind WordPiece is to prefer merges that result in subword units useful for predicting text. For instance, while BPE may merge frequent pairs regardless of context, WordPiece evaluates whether a candidate merge improves the probability of the corpus. This often leads to linguistically meaningful units being preserved, enhancing generalization to rare or unseen words.

Illustrative Example. Suppose the corpus contains words like `play`, `playing`, and `player`. WordPiece may merge `play` into a unit because doing so helps model all three words more compactly, while still leaving suffixes like `##ing` or `##er` as separate tokens.

Algorithm. A simplified version of the procedure is shown below:

Algorithm 3 WordPiece

```
1: Input: Corpus $C$, size of target vocabulary $size$
2: $V \leftarrow$ set of all characters in $C$
3: while $|V| < size$ do
4:     $scores = \{\}$
5:     for all candidate merges $(A, B)$ in $C$ do
6:         $scores[(A, B)] \leftarrow \frac{\text{freq}(AB)}{\text{freq}(A) \cdot \text{freq}(B)}$    ▷ Likelihood-based scoring function
7:     end for
8:     $(A, B) \leftarrow$ pair with highest score from $scores$
9:     $C \leftarrow replace(C, (A, B), Z)$, where $Z = concat(A, B)$
10:    $V \leftarrow V \cup \{AB\}$
11: end while
12: Return: $V$, Subword vocabulary
```

WordPiece remains the standard tokenization method in the Transformer family of models built for bidirectional pre-training, including BERT [39], DistilBERT [50], and ALBERT [51], where efficient handling of rare words and morphological variations is critical.

3. Related Work

This chapter explores different methodologies for adapting existing models to new languages, with a focus on the European Portuguese variety of the language. In Section 3.1, we review research that applies fine-tuning and continued pre-training on previously trained models, while in Section 3.2, we review approaches that combine pre-training and fine-tuning with tokenizer adaptation. Section 3.3 examines the available datasets and benchmarks for the evaluation of the European Portuguese language.

3.1. Fine-tuning Approach

Utilizing existing models and adapting them through additional training for more specific tasks can be a faster and less expensive way to obtain competitive results compared to training from scratch. Pre-trained models fine-tuned to European Portuguese have been explored in several recent papers, including GlórIA [52], Sabiá [53], Gervásio-PT [54], and Albertina-PT [55].

3.1.1 Methodology

By using heavily trained models as a starting point, adaptation to new languages or domains can be achieved through additional training on target language data. This approach has been widely explored in various domains, including code generation [56], biomedical applications [57], legal text processing [58], and other specialized domains [59].

The fine-tuning process typically involves setting the pre-trained models to training mode, providing them with domain-specific or language-specific datasets, and training them for a comparatively shorter period than the initial pre-training phase. This approach leverages the general language understanding capabilities already encoded in the model while adapting the parameters to better handle the target language or domain.

Several variations of fine-tuning have been proposed to improve efficiency and effectiveness: **(i)** *Parameter-Efficient Fine-Tuning (PEFT)* with methods such as LoRA [9] and adapters [60] [61], that update only a small subset of parameters; **(ii)** *Instruction Fine-Tuning* by training models on instruction-following datasets to enhance their ability to follow user instructions [62]; and **(iii)** *Continued Pre-training*, with further pre-training on the *target language* data before task-specific fine-tuning [59]

3.1.2 Results

Fine-tuning approaches have shown positive results [8] in various languages and domains; however, they also present challenges such as a reduction in reasoning capabilities [63]. For European Portuguese specifically, GlórIA [52] demonstrated that fine-tuning a multilingual model on Portuguese data achieved performance comparable to models specifically designed for Portuguese. Similarly, Sabiá [53] showed that continued pre-training of existing multilingual models on Portuguese corpora led to significant improvements in Portuguese-specific tasks.

However, these approaches still require substantial computational resources and large amounts of target language data. The GlórIA model, for example, required training on over 52 billion Portuguese tokens and a runtime of 21 days [52], while Albertina-PT [55] used approximately 15 billion tokens and took 3 days for continued pre-training.

3.2. Exploring Tokenizers

An alternative approach to adapting existing decoder models to new languages is to expand the tokenizer component [64, 65] by either updating or replacing its vocabulary with new tokens. Tokenizers are one of the building blocks of most language models and provide the basis required to train and utilize these models effectively.

This approach focuses on modifying tokenizers and adjusting the corresponding embedding weights to adapt to new languages. One of its primary benefits is the minimal training, or in some cases, none at all.

3.2.1 Methodology

While fine-tuning and continued pre-training have been extensively studied, tokenizer adaptation has received comparatively little attention in the literature. Nonetheless, this methodology offers an appealing alternative due to its potential for efficiency, requiring minimal additional training or, in some cases, none at all. By intervening directly in the tokenizer and embedding layers, models can be adapted to represent target languages more faithfully without the computational demands of large-scale pre-training.

Several strategies for tokenizer adaptation have been proposed. One direction is vocabulary expansion, where additional tokens corresponding to words or subwords of the target language are incorporated into the existing tokenizer while retaining the original vocabulary [66]. Another approach is tokenizer replacement, which discards the original tokenizer and instead employs a new tokenizer trained specifically on the target language, adjusting the model's embedding matrix accordingly [67]. More recent work has also examined hybrid approaches that merge elements of the original tokenizer with language-specific tokens, striking a balance between preserving the model's prior knowledge and improving its coverage of the target language [68].

It is worth noting that existing studies often combine tokenizer adaptation with further fine-tuning. For example, Csaki et al. [68] explored replacing tokens in the tokenizer and re-aligning their embeddings, followed by fine-tuning the adapted models on target-language data. Their results highlight the potential of tokenizer adaptation while underscoring that research focused on tokenizer-only adaptation – without the subsequent training phase – remains largely unexplored. This dissertation contributes to addressing this gap by investigating the viability of tokenizer-based methods as a lightweight alternative for adapting models to new languages.

3.2.2 Embedding Initialization Strategies

A critical aspect of tokenizer adaptation is determining how to initialize the embeddings for newly added tokens. Several strategies have been proposed: **(i)** *Random Initialization* by assigning random values to new token embeddings [69]; **(ii)** *Subword Averaging* by computing new token embeddings as the average of their constituent subwords in the original tokenizer [70]; and **(iii)** *Cross-lingual Mapping* by using bilingual dictionaries to map embeddings from source to target language [71].

Although *Cross-lingual Mapping* has shown promising results, this approach relies on the availability of high-quality bilingual dictionaries linking the source and target languages. As it introduces an additional dependency and pre-processing step compared to the other strategies, this work does not pursue that approach. Instead, we propose a new strategy – *Position-Weighted Initialization* – which accounts for the varying importance of constituent tokens based on their position.

3.2.3 Results

Although still relatively underexplored, studies on tokenizer adaptation have shown encouraging results. Vocabulary expansion approaches have been able to improve effectiveness on downstream tasks in the target language while preserving the model's ability to process the original training languages [66]. Tokenizer replacement methods have also demonstrated that training with a tokenizer better aligned to the target language can reduce tokenization inefficiencies and lead to measurable gains in accuracy [67]. Hybrid methods, which balance the preservation of existing vocabulary with the introduction of new language-specific tokens, have been shown to offer competitive effectiveness while avoiding catastrophic degradation of the original language capabilities [68].

However, these results typically rely on additional fine-tuning or continued pre-training after tokenizer modifications. For example, Csaki et al. [68] reported that models required fine-tuning on target-language corpora to realize the benefits of tokenizer adaptation, particularly for languages with characters distinct from those present in the pre-training data. This reliance on post-adaptation training highlights an open research gap: the effectiveness of tokenizer-only methods without further fine-tuning remains insufficiently explored.

In summary, existing results suggest that tokenizer adaptation is a promising direction, but its full potential – particularly in low-resource and computationally constrained settings – has yet to be systematically explored. This dissertation aims to contribute toward filling this gap by evaluating tokenizer-only adaptation strategies for European Portuguese.

3.3. European Portuguese Benchmarks

One of the main challenges in creating and adapting models for European Portuguese is the limited availability of high-quality datasets. From our literature review, we identified several specifically designed for evaluating Portuguese language models, and focused on two comprehensive benchmarks.

The rationale behind this decision was motivated by the overrepresentation of Brazilian Portuguese in most available resources [72, 73]. As a result, identifying high-quality datasets exclusively in European Portuguese proved challenging. To ensure reliability and linguistic fidelity, we therefore focused on peer-reviewed benchmarks that explicitly focus on European Portuguese as their evaluation data.

3.3.1 extraGLUE

Originally introduced as an extension of the *GLUE* benchmark [74], *SuperGLUE* was designed to provide a more rigorous evaluation of natural language understanding. Its goal was to preserve the accessibility and robustness of GLUE's evaluation framework, while increasing task difficulty in order to ensure that significant progress on the benchmark results would require substantial innovations in language model design.

While *SuperGLUE* represents a strong candidate for evaluating general-purpose English language models, it is less suitable for our research context, given that our target language is European Portuguese rather than English.

To address this, we turned to the European Portuguese adaptation of SuperGLUE: *extraGLUE*[1]. First introduced by Santos et al. [54], this benchmark retains the goals of its parent while adapting the tasks to European Portuguese. The adaptation process relied primarily on automated machine translation using DeepL [2] as well as a filtering of tasks by selectively retaining those more suitable for Portuguese translation. For example, the CoLA dataset was excluded as ungrammatical English expressions are typically rendered as grammatical in Portuguese after translation, undermining the task at hand. This careful curation ensured that only datasets whose linguistic properties could be meaningfully preserved in Portuguese were included.

Composition. Similar to its parent benchmark, *extraGLUE* is composed of multiple tasks, each targeting a specific dimension of language understanding. This multi-task design ensures that evaluation captures broad comprehension rather than specialization. Among the tasks included, we highlight four core components: **(i)** BoolQ-PT, a question-answering dataset requiring binary (yes/no) responses; **(ii)** CB-PT, a textual entailment task where models determine whether a hypothesis entails, contradicts, or is neutral with respect to a premise; **(iii)** COPA-PT, a causal reasoning task evaluating the ability to infer cause-effect relationships; and **(iv)** MultiRC-PT, a reading comprehension task in which multiple answers can be correct.

During preliminary analysis, however, we identified limitations in the translation quality for several tasks, particularly in *MultiRC-PT*, *CB-PT* and *COPA-PT*. Such inconsistencies increase the risk of unreliable evaluation, which would undermine the reliability of our results. For this reason, we decided to concentrate on *BoolQ-PT*, which exhibited the highest translation fidelity.

Evaluation. Each task in *extraGLUE* has its own evaluation metric, typically accuracy or F1 score. The overall benchmark score is computed as the average effectiveness across all tasks, providing a comprehensive measure of a model's language understanding capabilities in European Portuguese.

[1]extraGLUE is available at: https://huggingface.co/datasets/PORTULAN/extraglue
[2]DeepL - https://www.deepl.com

However, since we only focused our benchmark on a single task, our score corresponded to the accuracy achieved on the *BoolQ* task.

Passage: *Da Vinci's Demons – A série estreou nos Estados Unidos no canal Starz a 12 de abril de 2013, e a sua segunda temporada estreou a 22 de março de 2014. A série foi renovada para uma terceira temporada, que estreou a 24 de outubro de 2015. Em 23 de julho de 2015, a Starz anunciou que a terceira temporada seria a última da série. No entanto, Goyer deixou em aberto a possibilidade de um regresso da minissérie.* **Question:** *Haverá uma 4ª temporada de Da Vinci's Demons?* **Answer:** Não
Passage: *Números de telefone gratuitos no Plano de Numeração Norte-Americano – Nos Estados Unidos da América, no Canadá e noutros países que participam no Plano de Numeração Norte-Americano, um número de telefone gratuito tem um dos códigos de área 800, 833, 844, 855, 866, 877 e 888.* **Question:** *O 844 é um número gratuito no Canadá?* **Answer:** Sim

Table 3.1: Representative examples from BoolQ-PT. The system's prediction is compared against the standard answer, with task effectiveness measured in terms of accuracy.

To illustrate the structure of BoolQ-PT, Table 3.1 provides two representative examples of passages, questions, and their standard answers.

3.3.2 CalamePT

CalamePT[1] is a benchmark specifically tailored to evaluate text completion capabilities in European Portuguese. It was originally introduced by Lopes et al. [52] as part of the *GlórIA* project, addressing the lack of evaluation resources tailored for assessing generative abilities in this language.

Composition. The dataset contains 2,476 carefully designed phrases to evaluate contextual prediction. Of these, 406 were manually written by native speakers to ensure linguistic variety and authenticity, while 2,070 were automatically generated with GPT-3.5 [2]. Each phrase is designed such that the final word is highly predictable from the preceding context, providing a controlled test of a model's capacity to track local dependencies and generate coherent completions in Portuguese.

[1]The *CalamePT* dataset is available at HuggingFace.
[2]ChatGPT - `https://openai.com/chatgpt/overview/`

Evaluation. The evaluation protocol is straightforward: for each phrase, the last word is masked, and the model is tasked with predicting it. The predicted token is then compared against the expected last word, with effectiveness scored in binary terms (match / no match). The final benchmark score is calculated as:

$$\text{Score} = \frac{\text{Correct Predictions}}{\text{Total Phrases}}$$

This simple yet effective approach offers a clear measure of a model's ability to process Portuguese context and generate accurate completions.

Prompt: *No mundo da diversidade, nada deveria impedir alguém de amar livremente. O amor não escolhe gênero, apenas se entrega ao* ___ **Expected completion:** coração
Prompt: *Um pequeno asteroide, chamado 1999 JJ72, percorre silenciosamente a cintura principal do nosso sistema* ___ **Expected completion:** solar
Prompt: *As negociações encerraram sem acordo, impossibilitando os peregrinos de realizarem a* ___ **Expected completion:** peregrinação

Table 3.2: Examples of CalamePT completion tasks. Each phrase omits the final word, which the model must predict. Accuracy is computed as the proportion of exact matches.

These examples illustrate the type of predictions required in *CalamePT*, with model effectiveness evaluated in terms of accuracy – the proportion of phrases for which the final word is correctly predicted.

4. Tokenizer Adaptation

This chapter presents the methodological framework for adapting existing language models to new target languages with minimal retraining. The central idea of this dissertation – "Hacking the Tokenizer" – is to exploit the tokenizer as the main adaptation point, allowing pre-trained models to process new languages by extending their vocabulary rather than applying pre-training to the model.

Recent work [75] demonstrates that extending tokenizers with domain-specific tokens can improve fine-tuning efficiency, which motivates our decision to carefully curate the Portuguese vocabulary extension.

Our case study focuses on European Portuguese (PT-PT). This language was selected mainly because it is our native language and the target audience of this work, but also because its orthographic and morphological characteristics pose interesting challenges for tokenizer adaptation. This makes it a strong test case for evaluating the effectiveness of tokenizer-level adaptation.

The overall workflow follows three stages: (§4.2) Selecting which new tokens to add to the vocabulary; (§4.3) Initializing embeddings for these tokens; and (§4.5) Optionally fine-tuning embeddings with lightweight training.

This chapter details each of these stages and prepares the ground for the inference adaptation in Chapter 6 and results in Chapter 7.

4.1. Experimental Setup

All experiments in this chapter were conducted under a consistent configuration, unless otherwise specified:

- **Models:** `HuggingFaceTB/SmolLM2-135M` as a small monolingual baseline, `Qwen/Qwen2.5-1.5B-Instruct` as a strong multilingual reference, and `HuggingFaceTB/SmolLM3-3B` as a larger multilingual variant.
- **Tokenizers:** Byte-Pair Encoding (BPE) with initial vocabulary sizes of approximately $50{,}000$, $150{,}000$, and $130{,}000$, respectively.
- **Portuguese Corpus:** The European Portuguese portion of the OpenSubtitles dataset[1].

[1] https://opus.nlpl.eu/results/en&pt/corpus-result-table

- **Evaluation Benchmarks:** CalamePT (§3.3.2) and *extraGLUE* (§3.3.1), described in Chapter 7.
- **Hardware:** All experiments were executed on a Zephyrus G14 (2023) laptop equipped with an NVIDIA RTX 4060 GPU (8GB VRAM).

4.2. Token Selection Methodology

The first step in adapting a model to a new language is identifying which tokens to add. Tokenizer design choices such as vocabulary size, training corpus, and pre-tokenization regular expressions can significantly impact downstream model effectiveness [76].

To adapt `HuggingFaceTB/SmolLM2-135M` to Portuguese, we expanded the vocabulary by up to 10,000 tokens using the `add_tokens`[1] functionality of the Hugging Face Transformers library.

The selection process of *new_tokens* to expand our tokenizer, was as follows:

1. Train a new BPE tokenizer with a target vocabulary size of $2 \times size$, where $size$ is the number of tokens to be added.
2. Remove all tokens already present in the original vocabulary.
3. Remove tokens that are substrings of existing tokens (e.g., discard *pub* if *público* is already in the vocabulary).
4. Rank the remaining tokens by frequency of occurrence in the Portuguese corpus.[2]
5. Select the top $size$ tokens for integration.

This pipeline ensures that the expansion captures Portuguese-specific tokens with high frequency and utility while minimizing redundancy with the original tokenizer. For example, words like "*chegada*" (arrival), "*trabalhar*" (to work), or "*rapidamente*" (quickly) frequently appear in Portuguese but are fragmented into multiple subtokens by the original tokenizer.

4.3. Embedding Initialization Strategies

When extending the vocabulary of a pre-trained language model, each newly added token requires an embedding vector. The quality of this initialization has a significant impact on

[1]HuggingFace documentation
[2]Here, frequency is defined as the number of times a token appears in the training corpus.

downstream effectiveness: poorly chosen embeddings can disrupt the geometry of the embedding space, leading to semantic inconsistencies and degraded predictions [71, 77].

To address this, prior work has explored a range of strategies for initializing new embeddings, from simple random assignment to cross-lingual alignment and heuristic methods that exploit the structure of the existing embedding space. Below we review the most common baselines used in vocabulary extension and introduce our own methodology. Additionally, we introduce a dedicated evaluation framework for comparing initialization strategies. This allowed us to empirically select which approach best aligned with our work.

4.3.1 Baseline Methods

As reviewed in Chapter 4.3, prior work has proposed several strategies for initializing new token embeddings. In our experiments, we consider representative approaches spanning the main categories:

- **Random Initialization:** Embeddings are sampled from the distribution used during model training [78].
- **Heuristic Initialization:** Methods that construct new embeddings from existing ones. For example, FOCUS [79] combines embeddings using weighted sums, while OFA [80] incorporates external multilingual word vectors.

Cross-lingual approaches such as WECHSEL [81] have demonstrated strong results, but we exclude them here due to their reliance on high-quality parallel corpora or bilingual lexicons. Our focus remains on lightweight, resource-agnostic strategies that can be deployed under minimal assumptions.

4.3.2 Mean Vector Initialization

The first approach, termed “Mean Vector Initialization”, computes the embedding for each new token by averaging the embeddings of its constituent subtokens as determined by the original tokenizer.

This method provides a straightforward way to capture the average semantic content of the constituent tokens. For example, the Portuguese word “chegada” might be tokenized as “che” + “gada”; the embedding for the new single token “chegada” would be the average of the embeddings for “che” and “gada”.

Mean Vector Initialization is computationally efficient and intuitive, but it treats all subtokens as equally important, which may not always hold in morphologically rich languages.

4.3.3 Position-Weighted Initialization

The second approach, termed “Position-Weighted Initialization”, assigns differential importance to constituent tokens based on their position within the sequence. This method is predicated on the hypothesis that initial tokens in a sequence often carry greater semantic weight in autoregressive language models.

This method can be implemented as follows:

Algorithm 4 Position-Weighted Initialization

```
1: Input: New token $t$, represented by subtoken IDs $(s_1, s_2, \ldots, s_n)$; embedding matrix $E$; decay parameter $\alpha$
2: $M \leftarrow [E[s_1], E[s_2], \ldots, E[s_n]]$, embeddings for all subtokens
3: $i \leftarrow 0$
4: while $i < n$ do
5:     $w_i = \alpha^{(n-i+1)}$
6:     $w_i = \frac{w_i}{\sum_j w_j}$, Normalize weights
7: end while
8: $e_t = \sum_{i=1}^{n} w_i \cdot M[i]$, Compute weighted embedding
9: Output: Initialized embedding $e_t$ for token $t$
```

By weighting the subtokens according to position, this strategy preserves more semantic information learned from the original model while still incorporating contributions from later subtokens.

4.4. Comparative Methodology of Initialization Methods

We conducted extensive experiments to compare the effectiveness of these initialization strategies, including various values of the weighting parameter α for the Position-Weighted Initialization method.

To ensure a fair comparison across initialization strategies, we designed a controlled evaluation pipeline that focuses on the predictive ability of new tokens introduced into the vocabulary. The procedure is detailed below:

1. A list of Portuguese-specific tokens (`new_tokens`) was generated from the BPE tokenizer trained on the `OpenSubtitlesPT` subset (see §4.2).
2. For each initialization method, the base model was modified to include the `new_tokens` and corresponding embeddings.
3. From the `OpenSubtitlesPT` corpus, we sampled 1,000 lines containing at least **one** of the `new_tokens`.
4. For each line:
   (a) Randomly select **one** `new_token` present in the line.
   (b) Split the line at that token, dropping the final piece (i.e., *phrase.split(token)[:-1]*).
   (c) Use `model.generate` to continue the text from the remaining fragments.
   (d) Record the model's predicted *score* and *rank* of the correct `new_token` among all vocabulary candidates.
5. Repeat until all sampled lines have been processed.

To enable a direct comparison across initialization strategies, we additionally recorded, for each sampled line, which method assigned the lowest rank to the correct token. We then aggregated these results by counting the number of "wins" – instances where a method achieved the best rank among all compared strategies. This win-rate metric complements the per-method rank distributions by providing a simple yet robust comparative measure of effectiveness.

This methodology directly evaluates how well each initialization strategy positions new embeddings in the model's vector space, as measured by the model's ability to correctly predict unseen tokens in realistic contexts. It also allows us to compare methods under consistent conditions without additional fine-tuning or external resources.

The outcomes of this evaluation are presented and analyzed in Chapter 7.

4.5. Embedding Fine-tuning

In addition to initialization, we implemented a lightweight, context-aware fine-tuning procedure to further refine the embeddings of the new tokens. The key idea is to update only the embeddings of the newly added tokens while leaving the rest of the model frozen. This ensures minimal interference with the pre-trained knowledge.

For each new token, similarly to the initialization methodology comparison, we extract phrases from the training dataset that precede the token and discard the trailing fragment. We then run the model in generation mode to compute logits (i.e., scores) and hidden states for these phrases. The embedding for the target token is updated proportionally to the difference between the maximum predicted logit and the logit corresponding to the token, using the last hidden state of the generation as a directional signal.

This procedure can be formalized as follows:

Algorithm 5 Context-Aware Embedding Fine-Tuning

```
1: **Input:** Pre-trained model $M$, tokenizer $T$, new token set $\mathcal{N}$, training dataset $\mathcal{D}$, learning rate $\eta$
2: **for all** new token $t \in \mathcal{N}$ **do**
3:     $phrases \leftarrow \{phrase : phrase \in \mathcal{D} \land t \in phrase\}$, Collect all phrases containing $t$
4:     **for all** $phrase \in phrases$ **do**
5:         $phrases_{split} \leftarrow phrases.split(t)[: -1]$, Keep text that expects $t$
6:         **for all** $p \in phrases_{split}$ **do**
7:             $g \leftarrow M.generate(p)$, Pass the text through the model
8:             $logits \leftarrow g.logits$, Compute $logits = [l_1, l_2 \ldots l_n]$
9:             $hidden \leftarrow g.hidden\_states$, Obtain the hidden states of the generation
10:            $\Delta l = \max(\text{logits}) - \text{logits}[t]$
11:            $h \leftarrow hidden.last()$ Extract last hidden state vector from $hidden$
12:            $h \leftarrow h.norm()$, Normalize $h$ to unit length
13:            $E[t] \leftarrow E[t] + \eta \cdot \Delta l \cdot h$, Update embedding for new token $t$
14:        **end for**
15:    **end for**
16: **end for**
17: **Output:** Updated embedding matrix for new tokens
```

This approach nudges the new embeddings toward vector representations that better align with the model's predictions in realistic contexts while preserving the pre-trained model's original knowledge. Importantly, empirical evaluation showed no significant bias, as measured by downstream effectiveness metrics.

4.6. Chapter Summary and Rationale for Method Selection

In this chapter, we presented a systematic approach for adapting pre-trained language models to a new target language by "hacking" the tokenizer: A resource-agnostic approach that avoids costly large-scale retraining. The process involved three stages: (1) selecting new tokens, (2) initializing their embeddings, and (3) optionally fine-tuning them with lightweight updates.

We also introduced a comparative evaluation pipeline designed to assess the effectiveness of different embedding initialization strategies in a controlled setting. This framework enables us to quantify how well each strategy positions new tokens within the embedding space and how reliably the model can generate them in realistic contexts.

The detailed experimental results of this comparative analysis are presented in Chapter 7. Based on these findings, we identify a default initialization strategy that will serve as the foundation for all subsequent evaluations of tokenizer adaptation throughout this dissertation.

5. Exploratory Analysis and Design Rationale

As introduced in Chapter 1, this dissertation is guided by a set of research questions focused on tokenizer adaptation and evaluation for European Portuguese. This chapter presents exploratory analyses addressing these questions and outlines the rationale behind our methodological decisions.

In Chapter 4, we examined several tokenizer adaptation strategies, including vocabulary expansion and replacement. Given the wide overlap between tokens trained on European Portuguese corpora and English, expansion proved especially well-suited, allowing us to add only the missing tokens, simplifying the adaptation process while preserving the efficiency of the original tokenizer.

A second area we explored concerned model embedding initialization techniques. Our exploratory analysis suggested that position-aware weighting schemes could produce stable results. This motivated us to include such approaches in our systematic comparison.

With these design choices established, we proceed to explore how to best answer our initial questions presented in Chapter 1.

It is worth mentioning that, unless otherwise noted, experiments follow the configuration outlined in Section 4.1 and are conducted using the `SmolLM2-135M` model for efficiency.

Terminology Note: Throughout this chapter, the term *baseline* is used to refer to the original pre-trained model without any modifications. All comparisons are made relative to this unmodified baseline unless otherwise specified.

5.1. Objectives

The main goal of this research is to develop efficient and cost-effective strategies for adapting language models to new languages. To address this objective, we identified several potential points of interest in the adaptation pipeline. After careful consideration, we selected the tokenizer as the focal point of study. Tokenizers are a relatively under-explored component of language models [82], despite being among the earliest steps in the processing pipeline and having demonstrated an impact on model effectiveness [83].

After deciding on the tokenizer as the central point of research, we conducted early exploration, including direct edits and full replacements, without a clearly defined methodology.

From these preliminary trials, several recurring questions stood out, gainning traction in the formulation of our research objectives and were formalized as the guiding research questions (RQs) for this dissertation.

RQ1: Impact on Model Effectiveness The first and most central question of our research is: *"Can an English-trained model achieve comparable effectiveness in European Portuguese through strategic modifications of the tokenizer?"*

This question defines what "successful" adaptation means within our framework. A positive outcome would indicate that tokenizer adaptation alone is sufficient to bring Portuguese effectiveness closer to English, thereby validating the approach as a viable strategy. Conversely, a negative outcome would not invalidate tokenizer adaptation entirely but would highlight its limitations and point toward the need for complementary techniques.

RQ2: Token Efficiency – Fertility Metrics During our exploratory phase, we observed that the adapted tokenizers consistently produced shorter tokenizations for the same inputs. This was expected, given the increased vocabulary coverage for the target language. However, this effect had only been reflected at the tokenization step prior to model inference. A natural question then followed: *"What is the impact on model generation efficiency of tokenizer adaptation?"*

Here the focus shifts from accuracy to efficiency. While shorter tokenizations suggest reduced computational load, the real impact must be measured during inference, which is why we focused on this topic as a focal research question.

RQ3: Model Architecture Impact on Adaptation Although conducted at a smaller scale than the final experiments, initial test results suggest that different models may be affected differently by same tokenizer modifications. This observation prompted our third research question: *"Does tokenizer adaptation affect all models equally, or are there differences based on model architecture and size?"*

Because language models vary widely in scale and design, it is unclear whether adaptation strategies generalize, revealing the need to answer this question.

RQ4: Embedding Initialization Strategies This question became unavoidable once we realized that expanding the tokenizer unavoidably requires extending the embedding layer. Unsurprisingly, our fourth research question emerged: *"Which embedding initialization strategies are most effective for integrating new tokens into a pre-trained model?"*

RQ5: Impact on English Effectiveness In spite of surfacing later in the research process, it is still important to address our fifth and final question. Naturally, after adapting our models, preservation became a point of discussion for our research and, as such, this question presented itself: *"Does tokenizer adaptation reduce effectiveness in the model's source language?"*

5.2. RQ1: Impact on Model Effectiveness

Question: *Can an English-trained model achieve comparable effectiveness in European Portuguese through strategic modifications of the tokenizer?*

A simple and effective way to measure the impact of tokenizer adaptation on effectiveness in a target language is to benchmark the model before and after adaptation. However, this alone does not fully address our first research question, as our aim is to determine whether tokenizer adaptation can bring effectiveness in the target language closer to that of the source language.

Evaluating this question requires benchmarking model effectiveness in both languages using comparable datasets. While direct one-to-one comparisons are not always possible, we leverage the `extraGLUE` benchmark (for European Portuguese) as a parallel to the widely used `SuperGLUE` benchmark (for English). By comparing effectiveness across these benchmarks before and after adaptation, we can assess the extent to which adaptation narrows the effectiveness gap between the two languages.

For this purpose, we evaluate three stages of model adaptation:

- **Baseline:** Model with original tokenizer.
- **Post-Adaptation:** Model with new tokens integrated into the vocabulary.
- **Post-Training:** Model after lightweight fine-tuning of the new embeddings.

These comparisons allow us to quantify the effect of tokenizer adaptation on task-specific effectiveness in European Portuguese. The results of this evaluation are presented in Chapter 7.

5.3. RQ2: Token Efficiency – Fertility Metrics

Question: *What is the impact on model generation efficiency of tokenizer adaptation?*

During early experiments, we observed that adapted tokenizers consistently produced shorter tokenizations for identical inputs. This was expected given the increased coverage of Portuguese-specific tokens. However, this observation reflects only the tokenization process and does not guarantee improved efficiency during model inference. This raised the question of whether efficiency gains at the tokenizer level translate into real gains during model generation.

The *fertility* metric [68, 82], defined as the average number of tokens required to represent a word, initially appeared to be a suitable measure. Yet, its usefulness is limited to tokenization analysis and does not directly reflect model effectiveness. To bridge this gap, we introduced two complementary metrics designed to capture efficiency during inference: **Fertility Output** and **Fertility Boost**.

Fertility Output

This metric measures the number of tokens a model generates to complete a full word during inference. It presented a straightforward implementation that consists of two simple steps: **(i)** provide a fixed prefix (e.g., `"Em português, a palavra para"`); **(ii)** iteratively extend the prefix until the model generates a complete word.
This process is then repeated across multiple phrases, and the average number of tokens generated per word is calculated. An illustrative implementation is shown below:

Example Implementation:

```
num_tokens = []
for phrase in phrases:
    tokens = tokenizer.encode(phrase)
    for n in range(3, len(tokens)):
        generation = []
        while not is_full_word(generation):
            next_token = model.generate(tokens[:n])
            generation.append(next_token)
        num_tokens.append(len(generation))
fertility_output = average(num_tokens)
```

Limitations

Due to the iterative nature of this procedure, we were limited to using only a small set of data samples (four Portuguese phrases across different contexts). While this restricts generalization, the results provide a useful proof-of-concept for the metric.

Fertility Boost

While *Fertility Output* (§5.3) quantifies number of tokens per word generated, this new metric — **Fertility Boost** — estimates the frequency of newly added tokens during generation. This provides insight into how well the model has incorporated the new vocabulary and whether these tokens accelerate inference.

The intuition behind this metric is that each new token corresponds to a sequence of two or more original subtokens. If the model generates the new token directly, it achieves a speed gain relative to the baseline tokenizer. For example, suppose `N1` represents `t1+t2`, and `N2` represents `t3+t4+t5`. A baseline model would need five decoding steps, while the adapted model completes the same output in only two:

Timeline	Time1	Time2	Time3	Time4	Time5
Baseline	t1	t2	t3	t4	t5
Adapted	N1	N2			

Both models produce the identical output sequence (`t1t2t3t4t5 = N1N2`), but the adapted model completes the output sooner, reflecting an efficiency gain.

Methodology. In order to avoid the computational cost limitations observed with *Fertility Output*, we measure *Fertility Boost* by sampling from the model's logits to approximate realistic generation behavior:

1. Encode texts known to contain added tokens.
2. Run a forward pass through the model (without decoding).
3. Apply temperature scaling ($T = 0.8$) to the logits.
4. Sample from the probability distribution to obtain predicted tokens.
5. Count the proportion of sampled tokens that belong to the new vocabulary.

6. Repeat this process multiple times (10 runs) and report the mean and standard deviation.

Algorithm 6 Fertility Boost Estimation

```
1:  Input: Adapted model $M$, tokenizer before adaptation $T$, dataset $\mathcal{D}$, new tokens $\mathcal{N}$,
    temperature $\tau$, repetitions $R$
2:  $boosts \leftarrow []$
3:  for $r = 1 \ldots R$ do
4:      for all $phrase \in \mathcal{D}$ do
5:          $input \leftarrow T.encode(phrase)$
6:          $logits \leftarrow M.forward(input)$
7:          $scaled \leftarrow logits/\tau$
8:          $pred \leftarrow$ Sample$(softmax(scaled))$
9:          $ratio \leftarrow \#\{p \in pred : p \in \mathcal{N}\}/|pred|$
10:         Append $ratio$ to $boosts$
11:     end for
12: end for
13: Output: Fertility Boost $= |boosts|^{-1} \sum boosts$
```

By directly measuring generation-time efficiency, *Fertility Boost* captures how effectively new tokens accelerate output compared to the baseline tokenizer, but this field by itself does not.

Effective Efficiency Gain

While *Fertility Boost* quantifies the extent to which new tokens are used during generation, it does not by itself indicate how much efficiency is gained. To estimate the actual improvement, we combine it with the classic notion of *fertility*, evaluated on the tokenizer and defined as the average number of tokens required per word. Comparing fertility before and after adaptation yields a **Fertility Gain**, which represents the theoretical compression benefit of the new vocabulary.

By combining both measures, we obtain the **Effective Efficiency Gain (EEG)**:

$$\text{EEG} = \text{Fertility Gain} \times \text{Fertility Boost}$$

This metric is key because it bridges theoretical compression (*Fertility Gain*) with actual usage during generation (*Fertility Boost*). Without combining both, either aspect can be misleading: a tokenizer may look efficient in theory but rarely used in practice, or vice versa. For example, if the adapted tokenizer achieves a Fertility Gain of 0.25 (25% fewer tokens per word) but the model uses new tokens only 60% of the time (`Fertility Boost` = 0.6), the effective speedup is $0.25 \times 0.6 = 0.15$, corresponding to a 15% real-world improvement.

5.4. RQ3: Model Architecture Impact on Adaptation

Question: *Does tokenizer adaptation affect all models equally, or are there differences based on model architecture and size?*

Our early-stage experiments mainly focused on a single model – *Hugging Face/SmolLM2-135m* – which made us unaware of how adaptation would differ depending on model architecture. However, once we started exploring with the model *Qwen2.5-1.5B-Instruct*, we noticed model architecture may impact tokenizer adaptation. The main difference we noticed between the two models that may affect tokenizer adaptation was the original tokenizer's size. While our smaller model had a vocabulary containing approximately 50,000 tokens, our slightly bigger model has approximately 150,000 tokens. This raised the hypothesis that the benefits of tokenizer adaptation may depend not only on the adaptation procedure itself but also on the underlying model architecture – especially the size of its original vocabulary.

Fortunately, this question can be addressed by simply comparing the same tokenizer adaptations across models of varying sizes and configurations, and checking whether the results remain consistent across all architectures. For that, we offer results for the following models:

- `SmolLM2-135M`[1] – a monolingual model with 135 Million parameters.
- `Qwen2.5-1.5B-Instruct`[2] – a multilingual model with 1.5 Billion parameters.
- `SmolLM3-3B`[3] – a multilingual model with 3 Billion parameters.

[1]Hugging Face/SmolLM2-135m - https://huggingface.co/HuggingFaceTB/SmolLM2-135M
[2]Qwen/Qwen2.5-1.5B-Instruct - https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct
[3]Hugging Face/SmolLM3-3B - https://huggingface.co/HuggingFaceTB/SmolLM3-3B

5.5. RQ4: Embedding Initialization Strategies

Question: *Which embedding initialization strategies are most effective for integrating new tokens into a pre-trained model?*

Embedding initialization determines how smoothly new tokens integrate into the existing representation space. Poor initialization risks destabilizing training, while effective methods can accelerate adaptation and preserve learned knowledge. We therefore systematically compare initialization strategies – ranging from random assignment to sub-token averaging to weighted approaches such as `weighted_drop` – to identify the method that best balances stability, convergence, and effectiveness.

To address this question, we systematically compare a range of initialization strategies, including:

- Zero initialization
- Random uniform sampling
- Mean of existing token embeddings
- Dimension-wise min/max of token embeddings
- Weighted dropout of similar embeddings (e.g., `weighted_drop(2)`)

The methodology for evaluating these strategies was introduced in Chapter 4 (§4.4), where we outlined a controlled setup to measure how effectively each method positions new tokens in the embedding space. The corresponding results are presented in Chapter 7, where we identify the initialization approach that provides the best trade-off between stability, convergence, and downstream effectiveness.

5.6. RQ5: Impact on English Effectiveness

Question: *Does tokenizer adaptation reduce effectiveness in the model's source language?*

While not the central focus of our study, it is nonetheless interesting to understand how adapting a model's embedding layer affects its original effectiveness. To investigate this, we compare English benchmark results before and after adaptation, quantifying any effectiveness drop relative to the baseline. By monitoring both languages in parallel, we determine whether tokenizer adaptation produces multilingual benefits or instead introduces interference in language understanding.

This question ensures that our methods do not sacrifice source language performance while seeking target-language gains – a critical requirement for practical multilingual adaptation.

6. Inference Adaptation

Extending the tokenizer with new tokens enables more efficient representation of Portuguese text but also introduces challenges, since the model has never encountered these tokens during pre-training, which may result in incoherent or repetitive outputs.

This chapter introduces an inference adaptation framework designed to mitigate this problem.

6.1. Motivation

To understand the need for a specialized inference procedure, we performed stepwise experiments illustrating the evolution of model behavior.

All experiments in this chapter were conducted using the following model configuration:

- **Model:** HuggingFaceTB/SmolLM2-135M
- **Tokenizer (BPE) Dataset:** OpenSubtitles[1], sample of 1000 datapoints
- **Number of Added Tokens:** 10,000
- **Embeddings Initialization Method:** `weighted_drop(1.5)`
- **Maximum New Tokens:** 20

6.1.1 Baseline Generation

We first tested the model using its original tokenizer and weights (Table 6.1) without adding any new tokens. This baseline demonstrates how the pre-trained model performs on Portuguese prompts prior to any modifications.

Prompt	**Generated Output (Original Tokenizer)**
Olá, podes contar-me um poema com as palavras: Sol, Lua, Céu e Nuvens?	Ao fazer isso, o poeta deve ter uma palavra
Bacalhau é um dos peixes mais utilizados na culinária	Ao entender a criatura dos peixes, o uso de

Table 6.1: Baseline generation

[1]OpenSubtitles Corpus: https://opus.nlpl.eu/results/en&pt/corpus-result-table

6.1.2 Default Inference Generation After Tokenizer Adaptation

Next, we added 10,000 Portuguese-specific tokens to the tokenizer. Using the default inference procedure with the expanded tokenizer, the model produced incoherent generations, as shown in Table 6.2. This highlights the limitations of simply adding tokens without modifying inference.

Prompt	**Generated Output (Default Inference)**
Olá, podes contar-me um poema com as palavras: Sol, Lua, Céu e Nuvens?	Ao que o que o que o que o que o que o que o que o
Bacalhau é um dos peixes mais utilizados na culinária	de alimentos. Ao entender a dieta dos alimentos, o

Table 6.2: Adapted model default inference generation

6.2. Proposed Inference Framework

To address the problems observed in the previous step, we developed a two-phase inference adaptation procedure:

1. **Input Processing**: Input text is first tokenized using the original tokenizer, allowing the model to leverage its pre-trained distribution.
2. **Output Processing**: If the model generates a newly added token, it is dynamically replaced with its constituent original tokens before being fed back into the model.

This ensures compatibility with the pre-trained distribution while preserving token efficiency gains. An algorithmic proposal for this approach follows:

Proposed Algorithm

```
def inference(model, tokenizer, encoding_tokenizer, prompt):
    outputs = []
    for i in range(max_new_tokens):
        if len(outputs) > 0:
            inputs = encoding_tokenizer.encode(prompt + tokenizer.decode(outputs))
        new_token = model.generate(input_ids, max_new_tokens=1)
        outputs.append(new_token)
    generation = tokenizer.decode(outputs)
    return generation
```

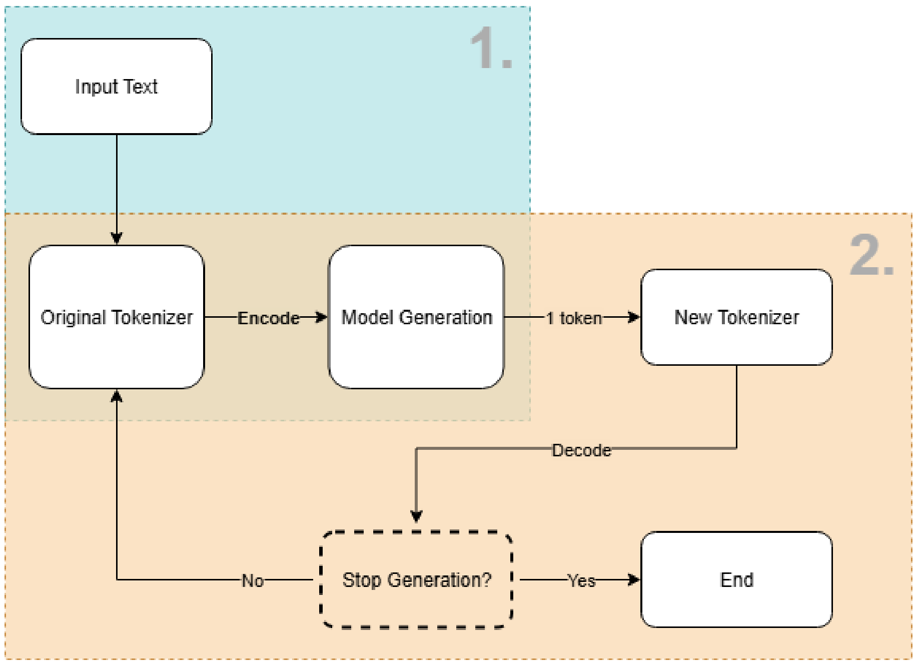

Figure 6.1: Inference Adaptation Framework

6.3. Implementation Considerations

The current implementation was developed at the Python level using the Hugging Face Transformers library. For production-level deployment, future work would require:

- Optimization of the inference process to minimize computational overhead.
- Development of custom extensions within `PyTorch` to natively handle dynamic token mapping.

6.4. Potential Benefits

If implemented at a lower level, inference adaptation could yield several benefits:

- **Improved Generation Coherence**: Reduce inconsistencies when generating text with new tokens. (See §6.5)
- **Enhanced Token Efficiency**: The approach would preserve the token efficiency gains of extending the vocabulary, while mitigating impacts on model effectiveness.

6.5. Illustrative Results

After applying the inference framework, coherence for the same prompts is now restored, as shown in Table 6.3:

Prompt	Generated Output (With Inference Adaptation)
Olá, podes contar-me um poema com as palavras: Sol, Lua, Céu e Nuvens?	Ao fazer isso, o poeta deve ter uma palavra
Bacalhau é um dos peixes mais utilizados na culinária	Ao entender a criatura dos peixes, o uso de

Table 6.3: Preliminary test outputs using the proposed inference adaptation with expanded tokenizer

These results confirm that the framework preserves coherence and aligns generation with Portuguese context, validating the proposed method in this chapter.

6.6. Summary

This chapter presented a step-by-step demonstration of why inference adaptation is necessary when extending a tokenizer with new tokens, followed by the development of a specialized framework. Preliminary tests showed that the naive use of new tokens leads to incoherent generation, whereas the proposed two-phase procedure ensures coherent and consistent generation.

7. Results

This chapter presents the experimental results of our tokenizer adaptation method. We first report quantitative evaluations, covering initialization strategies, benchmark accuracy, tokenization efficiency, rank distributions, and retention of source language effectiveness. These findings are later complemented by a qualitative evaluation of model completions and, finally, the chapter concludes with a discussion that synthesizes the implications of both quantitative and qualitative results, while noting encountered limitations.

7.1. Quantitative Results

Throughout this section, we provide evidence to support our conclusions on the effects of vocabulary expansion and embedding initialization. Each subsection reports results with a reflection on their meaning for the research questions guiding this work, as proposed in Section 1.2

7.1.1 Embedding Initialization Strategies

By focusing on embedding initialization strategies, we directly explore our RQ4, which concerns the impact of embedding initialization strategies on the stability and usefulness of adapted vocabularies.

Initialization Method	Number of Wins
Random Initialization	36
Mean Vector Initialization	39
Weighted Drop ($\alpha = 0.5$)	26
Weighted Drop ($\alpha = 1.0$)	37
Weighted Drop ($\alpha = 1.5$)	47
Weighted Drop ($\alpha = 2.0$)	53
Weighted Drop ($\alpha = 2.5$)	81
Weighted Drop ($\alpha = 3.0$)	89
Weighted Drop ($\alpha = 3.5$)	79
Weighted Drop ($\alpha = 4.0$)	99
Weighted Drop ($\alpha = 4.5$)	126
Weighted Drop ($\alpha = 5.0$)	127

Table 7.1: Frequency of best prediction (lowest rank) across 1,000 sampled lines for each initialization strategy.

Using the methodology described in Section 4.4, we compared initialization strategies by counting, for each method, how often it produced the best prediction (lowest score) for a new token across 1,000 sampled lines. This metric reflects the model's ability to correctly predict unseen tokens given partial context.

Table 7.1 presents results across models. Random initialization often led to unstable behavior, while both mean and position-weighted approaches were more reliable. Among these, the position-weighted strategy produced embeddings that better reflected token context, and so it was selected for the remaining experiments.

These results form the empirical foundation for discussing RQ4. A more comprehensive evaluation of their significance will follow later.

7.1.2 Benchmark Accuracy

In this section we make use of the benchmarks introduced in Section 3.3, so as to reinforce our strategy. Additionally, we explore questions **RQ1**, which focuses on the effectiveness of the adapted model in the target language.

The main benchmarks we focus on are: *CalamePT*, built to evaluate comprehension on Portuguese text by providing our models with incomplete phrases where the last word is easily obtainable through the context; *extraGLUE*, built to capture data on reasoning and text comprehension of a model, by providing it with questions where the answer does not directly emerge from it; and *SuperGLUE*, which is the original English version of *extraGLUE* making it useful for comparing a model's Portuguese effectiveness with its English counterpart.

Since the benchmark evaluation was designed to test whether adapted vocabularies improve accuracy on downstream tasks, each baseline model was therefore compared with two variants: one containing only the new tokens, and another in which those tokens were also given lightweight embedding training (§4.5).

The results, shown in Table 7.2, cover three models of different sizes: `SmolLM2-135M`, `SmolLM3-3B`, and `Qwen2.5-1.5B-Instruct`.

New Tokens	**Type**	**CalamePT**	**extraGLUE**[1]	**SuperGLUE**[2]
		Model: *HuggingFaceTB/SmolLM2-135M*		
0	Baseline	13.54 %	1.47 %	41.02 %
1000	No Training	13.54 %	1.47 %	-
1000	With Training	13.54 %	1.47 %	-
5000	No Training	13.54 %	1.51 %	-
5000	With Training	13.54 %	1.51 %	-
7500	No Training	13.54 %	1.51 %	-
7500	With Training	13.54 %	1.51 %	-
		Model: *HuggingFaceTB/SmolLM3-3B*		
0	Baseline	58.53 %	49.69 %	38.45 %
1000	No Training	58.53 %	49.69 %	-
1000	With Training	58.53 %	49.69 %	-
5000	No Training	58.53 %	49.69 %	-
5000	With Training	58.53 %	49.69 %	-
7500	No Training	58.53 %	49.69 %	-
7500	With Training	58.53 %	49.69 %	-
		Model: *Qwen/Qwen2.5-1.5B-Instruct*		
0	Baseline	49.61 %	40.24 %	60.88 %
1000	No Training	49.61 %	39.86 %	-
1000	With Training	49.61 %	39.79 %	-
5000	No Training	49.61 %	40.28 %	-
5000	With Training	49.61 %	40.21 %	-
7500	No Training	49.57 %	40.14 %	-
7500	With Training	49.57 %	40.06 %	-

Table 7.2: Results on Portuguese Benchmarks (*CalamePT* and *extraGLUE*)

Overall, benchmark accuracy remains stable across most configurations. Minor fluctuations (≤0.5%) suggest that vocabulary expansion has little effect on high-level reasoning (*extraGLUE*) or text completion (CalamePT) without more extensive training. Unsurprisingly, model effectiveness for English remained superior to that of Portuguese.

7.1.3 Generation Efficiency

Whereas benchmark accuracy reflects task correctness, generation efficiency captures how economically a model represents text in tokens, directly addressing **RQ2**, which asks whether tokenizer adaptation improves efficiency. More efficient tokenization reduces sequence length, speeds up inference, and lowers computational cost. We also address **RQ3**, although not as directly as *RQ2*, which questions how different model architectures influence tokenizer adaptation.

To examine this, we report two principal results in the main table: *FertilityOutput*, an empirical estimate of the average number of tokens the model produces per Portuguese word on a small held-out set of example generations; and *FertilityBoost*, the generation-side improvement measured under the sampling regime described in Section 5.3.

The third quantity, the *Effective Efficiency Gain (EEG)*, is derived from tokenizer-level *Fertility* statistics and combines them with generation-side improvements:

$$EEG = FertilityGains \times FertilityBoost \tag{7.1}$$

$$\text{where } FertilityGains = \frac{Fertility_{Original}}{Fertility_{Adapted}}.$$

Ideally, *FertilityOutput* would serve as the primary measure of generation efficiency, since it directly reflects how many tokens a model requires to generate full words in practice. However, computing *FertilityOutput* at scale proved computationally prohibitive. For this reason, we instead relied on *EEG* as a scalable proxy: it extrapolates expected efficiency by combining tokenizer-side *Fertility* statistics with the observed *FertilityBoost*.

Table 7.3 presents results for all evaluated models and adaptation strategies. We do not repeat the raw *Fertility* values in the main table because any two of the triplet {Fertility, FertilityBoost, EEG} determine the third, and presenting all three would be redundant; the full *Fertility* measures are available in Annex A.1

New Tokens	**Type**	**FertilityOutput**	**FertilityBoost**[1]	**EEG**[2]
		Model: *HuggingFaceTB/SmolLM2-135M*		
0	Baseline	2.47	-	-
1000	No Training	1.94	$5.07\% \pm 0.05\%$	$1.41\% \pm 0.01\%$
1000	With Training	1.94	$5.05\% \pm 0.06\%$	$1.40\% \pm 0.02\%$
5000	No Training	1.76	$12.88\% \pm 0.04\%$	$5.24\% \pm 0.02\%$
5000	With Training	1.76	$12.94\% \pm 0.11\%$	$5.27\% \pm 0.04\%$
7500	No Training	1.75	$16.03\% \pm 0.11\%$	$6.58\% \pm 0.05\%$
7500	With Training	1.75	$15.99\% \pm 0.12\%$	$6.56\% \pm 0.05\%$
		Model: *HuggingFaceTB/SmolLM3-3B*		
0	Baseline	2.00	-	-
1000	No Training	2.11	$0.22\% \pm 0.01\%$	$0.02\% \pm 0.00\%$
1000	With Training	2.11	$0.22\% \pm 0.01\%$	$0.02\% \pm 0.00\%$
5000	No Training	2.05	$0.87\% \pm 0.04\%$	$0.15\% \pm 0.01\%$
5000	With Training	2.04	$0.84\% \pm 0.06\%$	$0.15\% \pm 0.01\%$
7500	No Training	2.02	$1.05\% \pm 0.04\%$	$0.18\% \pm 0.01\%$
7500	With Training	2.03	$1.06\% \pm 0.04\%$	$0.18\% \pm 0.01\%$
		Model: *Qwen/Qwen2.5-1.5B-Instruct*		
0	Baseline	1.74	-	-
1000	No Training	2.19	$0.63\% \pm 0.02\%$	$0.06\% \pm 0.00\%$
1000	With Training	2.21	$0.62\% \pm 0.02\%$	$0.06\% \pm 0.00\%$
5000	No Training	2.19	$2.09\% \pm 0.04\%$	$0.34\% \pm 0.01\%$
5000	With Training	2.26	$2.12\% \pm 0.05\%$	$0.34\% \pm 0.01\%$
7500	No Training	2.19	$2.56\% \pm 0.04\%$	$0.41\% \pm 0.01\%$
7500	With Training	2.17	$2.52\% \pm 0.04\%$	$0.40\% \pm 0.01\%$

Table 7.3: Tokenization Efficiency

Interestingly, the smaller multilingual model sees considerable improvements in *EEG*, while the bigger multilingual models, to a lesser extent. It remains unclear whether the discrepancy in results stems from the multilingual characteristic of our two larger models, or if it's related to model size.

7.1.4 Retained Effectiveness on Source Language

While our primary interest is in Portuguese adaptation, it is equally important to verify that modifications do not degrade effectiveness in the model's original training language(s). To this end, we evaluated all variants on the **Massive Multitask Language Understanding (MMLU)** benchmark, which covers 57 academic subjects and serves as a broad proxy for general knowledge and reasoning ability.

Table 7.4 reports MMLU accuracy for each configuration.

Model	New Tokens	Type	MMLU (%)
SmolLM2-135M	0	Baseline	23.25
SmolLM2-135M	1000	No Training	23.25
SmolLM2-135M	1000	With Training	23.25
SmolLM2-135M	5000	No Training	23.25
SmolLM2-135M	5000	With Training	23.25
SmolLM2-135M	7500	No Training	23.25
SmolLM2-135M	7500	With Training	23.25
SmolLM3-3B	0	Baseline	56.67
SmolLM3-3B	1000	No Training	56.67
SmolLM3-3B	1000	With Training	56.67
SmolLM3-3B	5000	No Training	56.67
SmolLM3-3B	5000	With Training	56.67
SmolLM3-3B	7500	No Training	56.67
SmolLM3-3B	7500	With Training	56.67
Qwen2.5-1.5B-Instruct	0	Baseline	58.97
Qwen2.5-1.5B-Instruct	1000	No Training	58.97
Qwen2.5-1.5B-Instruct	1000	With Training	58.97
Qwen2.5-1.5B-Instruct	5000	No Training	58.97
Qwen2.5-1.5B-Instruct	5000	With Training	58.97
Qwen2.5-1.5B-Instruct	7500	No Training	58.97
Qwen2.5-1.5B-Instruct	7500	With Training	58.97

Table 7.4: MMLU accuracy (baseline vs. adapted tokenizers).

Results demonstrate that MMLU results are **completely stable** across all settings. This indicates that tokenizer adaptation does not interfere with the model's original linguistic or reasoning capabilities. In other words, improvements for Portuguese are not obtained at the expense of effectiveness in the source language. We further explore this theory in the qualitative tests.

7.1.5 New Token Rank Distribution

In this section, we examine whether newly introduced tokens were preferred over their decomposed forms during generation. This analysis provides insight into whether the adapted model assigns a higher probability (i.e., lower rank) to the new tokens compared to the original decomposition.

To make this comparison, we first constructed a dedicated evaluation dataset and pipeline:

1. Identify **7,500** new tokens to add to the vocabulary using the procedure described in Section 4.2, trained on the `OpenSubtitlesPT` dataset.
2. Select datapoints from the **CalamePT** benchmark containing at least one of these tokens.

3. Run baseline generations with the original tokenizer, recording the score of the first subtoken in the decomposition of each new token.

4. Adapt the model by adding the new tokens without training their embeddings, and repeat the evaluation while recording the score of the new token directly.

5. Apply the proposed embedding fine-tuning method (§4.5) and repeat the same score extraction as in step 4.

This pipeline ensured a consistent comparison between the original decomposed forms and the newly added tokens. Unless otherwise specified, all results in this section and in the Annex were obtained using the **SmolLM2-135M** model, adapted with 7,500 tokens from OpenSubtitlesPT and evaluated on the CalamePT dataset. While additional runs were performed with different vocabulary sizes, results were broadly consistent, so we report this representative setting.

Figure 7.1 summarizes the scores obtained across all evaluated generations, while Figure 7.2 showcases the distribution of the difference $new\_token\_rank - old\_token\_rank$. We observe that, in many cases, the new token achieves a higher score than the original sequence, suggesting that the model internally recognizes the utility of these merged units. This effect is especially relevant under stochastic decoding strategies (*temperature* > 0), where alternative tokens can be sampled.

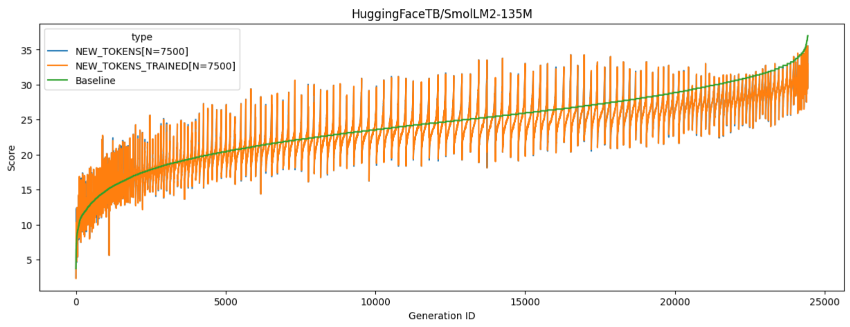

Figure 7.1: Score comparison between original sub-tokens and new tokens[1]

[1]We ran this comparison with an additional 7500 tokens.

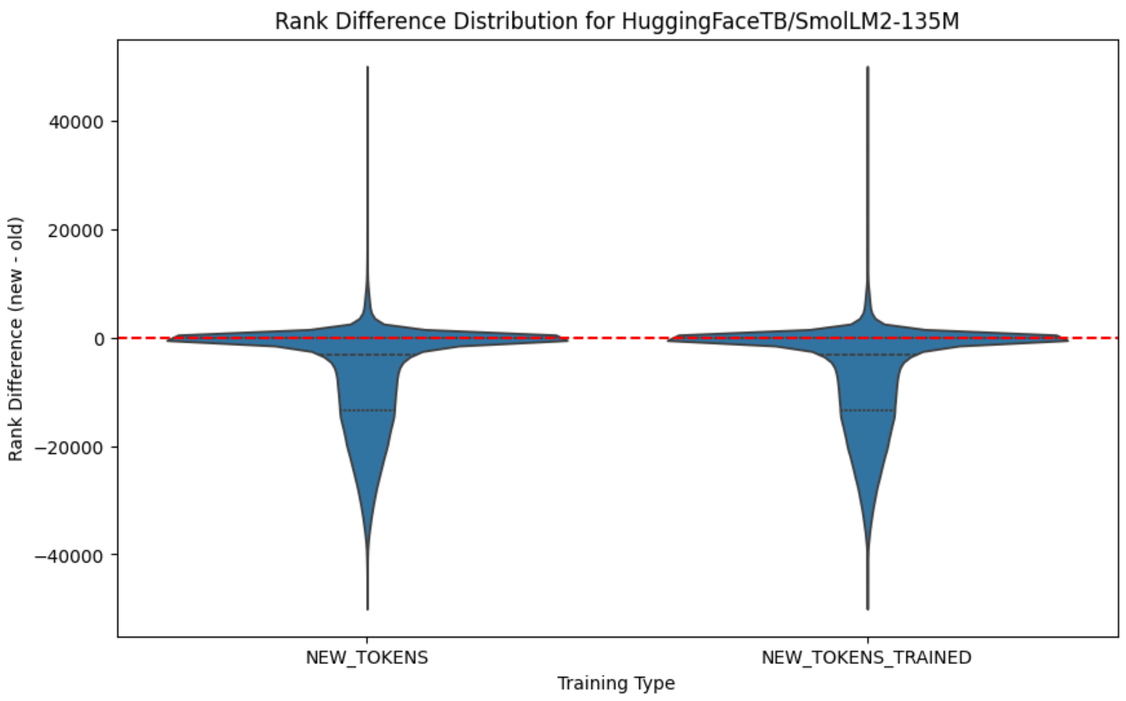

Figure 7.2: Distribution of rank differences across the entire results data, shown as violin plots. The x-axis indicates the training type, while the y-axis shows the rank difference $new\_token\_rank - old\_token\_rank$. Each “violin” depicts the distribution of values, where its width reflects the density of observations and the internal lines mark quartiles. Negative values indicate preference towards `new_token`.

Distribution of rank differences across the entire results data, shown as violin plots. The x-axis indicates the training type, while the y-axis shows the rank difference $new\_token\_rank - old\_token\_rank$. Each “violin” depicts the distribution of values, where its width reflects the density of observations and the internal lines mark quartiles. Negative values indicate preference towards `new_token`.

The rank analysis confirms that many new tokens were indeed assigned higher probabilities than their decomposed counterparts. This indicates that models learn to prefer merged units after adaptation. Interestingly, there doesn’t seem to exist a significant impact from applying further pre-training – adding new tokens with the right initialization strategy is enough. Extended plots for additional models are presented in Appendix A.2 (score comparison) and Appendix A.3 (rank differences).

7.2. Qualitative Evaluation

Benchmark scores and fertility metrics only partially capture the interactive experience of using a model. To complement the quantitative analysis, we conducted a qualitative comparison of completions. This evaluation was carried out for both European Portuguese

(Table 7.5) and English (Table 7.6), using equivalent prompts in each language.

Generations were constrained to a maximum of 50 new tokens, which increases the likelihood of truncated phrases. Sampling was applied with the following parameters: *temperature* (0.8), *top-p* (0.9), and *top-k* (100).[1]

Baseline	Adapted w/ 5000 tokens	Adapted and Trained
Prompt: "Para calcular a raiz quadrada de um número manualmente,"		

"retorna uma posição.
```
python
x = 10
y = 20
print(f"A raiz quadrada de {x} e {y} é {x ** "
```
	"utilizando o
	tipo de cálculo de fatorial. Por exemplo, se uma quantidade de pemares a pouco mais de um alcance, seja 110000"
		"
		e construa um número quadrado em duas partes aixa e a linda,
		A matoê estra:
		A matoê do primeiro num méênico da raiz"

Prompt: "Nos dias de hoje, democracia é o sistema politico"		
". Ao demorar por 6 dias para que seja o seu emissar, a dor da elemência eleitora atual aumentará e o próprio p"	"que ele, na quais seriam a diferença, de forma fatoriva, de uma média de 013020 00010000000010..."	", o que não se encontra nis. A que também é o democracia. Eluão de Hic e da Cria de lo Paulo. Hic"
Prompt: "O poema seguinte contém várias palavras-chave: azul, borboleta e sol:"		
"e sí, e também, e física e próximo. O poema também tem como seu último último, o estudante de"	"céu de la tierra, y vrgosos deixa. Vídeo como nome: «Cicio-dei alcoba e sol», com aumos dos dcimentoimens"	"Este estará por esqueço da esquina do lado do que a leitura seja frente ao largo do espaço". Para outras palavras-chave,"

Table 7.5: Illustrative Examples of Model Completions (Baseline vs. Adapted vs. Adapted and Trained) for Portuguese

The results reveal clear differences across conditions. Baseline generations are often syntactically incomplete, emphasizing the weak effectiveness in European Portuguese. Adapted models produce somewhat longer and more structured sentences, though improvements are modest. Nonetheless, they frequently invent morphology or generate ungrammatical constructions. Semantic grounding remains fragile: numerical reasoning prompts yield incoherent results, while open-ended prompts (e.g., political or poetic) elicit more topically appropriate responses in the Adapted and Trained variant.

[1]Temperature scales the logits before sampling, controlling randomness; lower values yield more deterministic generations. Top-p (nucleus sampling) restricts choices to the smallest set of tokens whose cumulative probability exceeds p. Top-k limits sampling to the k most probable tokens.

Prompt: "To calculate the square root of a number by hand,"

Baseline	Adapted w/ 5000 tokens	Adapted and Trained
" you would divide the number into equal parts and then subtract the results to get the square root of the result. In Python, we can use the built-in 'sqrt()' function to calculate the square root of a number. ###"	" the digits of the number can be marked on a piece of paper, or a digital calculator will do the job for you. Step 14: Choose a suitable scale Use a calculator or a digital scale to scale the numbers"	" the rule is to take the negative of the first digit, then multiply the result by itself, and then subtract the result ### Why Square Root Calculator? The square root of 1833195073330500"

Prompt: "Nowadays, democracy is the political system"

Baseline	Adapted w/ 5000 tokens	Adapted and Trained
" that allows citizens to choose the leaders who will guide them in their chosen path. They are also known as the people. DEMOCRACY - The people who rule a country or an organization.DEPARTMENT"	" of a country or country region where the people, including elected representatives, are able to choose the leaders of the country. In other words, the people are free to govern and choose their leaders and policies for their country. In the"	" that allows citizens to vote for their representatives, so that they can elect a government. In a democracy, people vote for politicians to represent them. This is similar to a democracy where people elect representatives who vote for those politicians. A"

Prompt: "The following poem contains several keywords: Blue, Butterfly, and Sun:"

Baseline	Adapted w/ 5000 tokens	Adapted and Trained
"Here is the beginning of the poem: Blue, Butterfly, and Sun Blue, Butterfly, and Sun Blue, Butterfly, and Sun Blue, Butterfly, and Sun Blue, Butterfly"	"Blue Butterfly I am blue, The butterfly of my mind I feel my mind Throbbing in my breast, And I cannot tell what to do. I cannot tell what to do,"	"Blue Butterfly Blue Butterfly I feel blue A butterfly has flown Across the blue sky And I feel sad I feel sad and angry Ia like the butterfly"

Table 7.6: Illustrative Examples of Model Completions (Baseline vs. Adapted vs. Adapted and Trained) for English.

Unsurprisingly, qualitative results mirror the quantitative findings. These results suggest that tokenizer adaptation to a new target language has no observable effect on effectiveness on the model's source language. The observed variations are attributed to sampling parameters rather than substantive changes in generation quality.

These qualitative results are presented only for the monolingual model, `SmolLM2-135M`, to better isolate the effects of adapting a model trained exclusively in one language to a different linguistic context.

7.3. Limitations and Discussion

The results reported in this chapter should be interpreted with caution. Many of the experiments rely on small-scale models such as `SmolLM2-135M`, which exhibit severe hallucination effects even before adaptation. This constrains the extent to which improvements can be generalized. Evaluation metrics introduce further challenges: benchmark scores, while robust for task accuracy, are coarse-grained and may fail to capture subtler shifts in fluency or style, whereas fertility metrics, though informative, are sensitive to corpus size and the heuristics used for token selection. As a result, the impact of adaptation may be underestimated in our current setup. Broader evaluations across larger architectures and human-judged fluency assessments would strengthen our findings.

Despite these caveats, the evidence points to a cautiously optimistic picture. Vocabulary expansion does not alter benchmark accuracy, but neither does it degrade it. Efficiency gains are substantial in small monolingual models but negligible in mid-scale multilingual ones. Rank analyses reveal that models learn to prefer newly added tokens, suggesting an internal mechanism by which adaptation affects fluency even when benchmarks remain stable. Qualitative analysis confirms that generations can become longer and more syntactically structured, though their semantic reliability remains weak. Notably, source language effectiveness remains fully stable, ensuring that Portuguese adaptation does not compromise the model's original competencies in English.

Taken together, these findings suggest that tokenizer adaptation is most promising in scenarios where baseline tokenization is inefficient and resources are limited. For lightweight models, especially those not initially optimized for Portuguese, the benefits are concrete and measurable. For larger multilingual systems, gains are less pronounced. In this sense, the results confirm the central hypothesis of this dissertation: token-level interventions may provide a low-cost pathway to make English-trained models more suitable for underrepresented languages such as European Portuguese.

8. Conclusions

The research presented in this dissertation set out to address the problem of adapting large language models to low-resource languages, with a particular focus on European Portuguese. While existing approaches largely rely on computationally expensive fine-tuning or continued pre-training, this work explored whether lightweight, training-free interventions at the tokenizer level could offer a viable alternative. By modifying the tokenizer vocabulary and embedding initialization strategies, the goal was to improve efficiency and effectiveness without requiring access to large-scale computational resources.

The developments carried out in this work converged into three central contributions. First, a systematic methodology for tokenizer adaptation was designed, encompassing vocabulary expansion, embedding initialization, and an inference adaptation framework. Second, a new strategy for embedding initialization was proposed and evaluated, *Position-Weighted Initialization*, which was identified as the most effective among those tested. Third, these methods were applied and validated on European Portuguese benchmarks, offering a practical demonstration of their potential for extending model capabilities to underrepresented languages.

The overall conclusion is that tokenizer adaptation represents a computationally efficient pathway for enabling cross-lingual use. The approach yielded measurable efficiency improvements, especially for smaller monolingual models, while maintaining effectiveness on benchmark tasks and preserving the original source language capabilities. At the same time, the findings revealed that the benefits are model-dependent, underscoring the importance of tailoring adaptation strategies to specific architectures. With these overarching results established, the remainder of this chapter turns to the answering of the research questions directly, before reflecting on the limitations of the work and outlining promising directions for future research.

Research Questions. The first research question asked whether an English-trained model could achieve comparable effectiveness in European Portuguese through tokenizer adaptation. The findings suggest that tokenizer-level modifications alone were insufficient to close the gap between English and Portuguese effectiveness. Results on *extraGLUE* consistently lagged behind those on *SuperGLUE*, providing a negative answer to this question. Nonetheless, this outcome is valuable, as it clarifies the boundaries of what tokenizer adaptation can and cannot achieve without complementary techniques.

The second research question concerned the impact on generation efficiency. The results obtained through *Effective Efficiency Gains* (Table 7.3) showed improvements in the smaller monolingual model, which required fewer tokens per generated word. Larger multilingual models, however, benefited only marginally, and deterministic settings still favored the original tokens. Efficiency improvements were therefore real, but mostly evident when sampling was enabled, and primarily in resource-constrained, smaller-scale settings.

The third research question asked whether adaptation affects all models equally, or if differences depend on architecture and size. Across all metrics in Table 7.3 and the *Rank Differences Distribution* illustrated in Figure **??**, architecture emerged as a key factor. Smaller monolingual models benefited substantially in terms of efficiency, while larger multilingual ones showed limited gains. Effectiveness itself remained broadly consistent, with differences mainly reflected in token distribution. These findings indicate that model architecture plays a meaningful role in determining the effectiveness of tokenizer adaptation.

The fourth research question focused on embedding initialization strategies. Among the tested methods, the newly proposed *Position-Weighted Initialization* consistently produced the most coherent embeddings and stable results (Table 7.1), confirming that positional context plays a critical role in effective embedding construction.

Finally, the fifth research question asked whether adapting the tokenizer reduces effectiveness in the model's source language, English. The findings from both quantitative evaluations using *MMLU* (Table 7.4) and qualitative completions (Table 7.6) showed no significant loss of effectiveness. This indicates that adaptation to Portuguese did not compromise English capabilities.

Taken together, these answers outline both the potential and the limitations of tokenizer adaptation as a strategy for cross-lingual transfer.

Limitations. Despite these contributions, several limitations must be acknowledged. The most important is the model-dependent nature of the observed gains: efficiency improvements were substantial in smaller monolingual models but negligible in larger multilingual ones. This suggests that the approach cannot yet be considered universally effective across all architectures. Another limitation lies in inference adaptation, which was only introduced conceptually and not implemented in a production-ready manner. While embedding initialization strategies improved semantic alignment, they cannot guarantee optimal placement without additional fine-tuning. The evaluation itself was also restricted to text completion and binary classification, leaving out more complex tasks such as translation, summarization, and creative generation. Finally, the study focused solely on European Portuguese, which prevents broader generalization and leaves open the possibility of trade-offs in multilingual effectiveness.

Future Work. Building on these findings, several avenues for further research emerge. A deeper analysis of how model size and architecture shape the effectiveness of tokenizer adaptation could help design strategies suited to specific classes of models. The development of a production-ready inference adaptation framework would ensure coherence and efficiency in real-world deployments. Lightweight fine-tuning approaches that optimize only the embeddings of new tokens may provide a middle ground between computational efficiency and improved effectiveness. The exploration of hybrid methods that combine tokenizer adaptation with parameter-efficient fine-tuning, such as *LoRA*, *adapters*, or prompt tuning, also present a promising direction. Extending the methodology to other low-resource languages, particularly those with complex morphology, would test its broader applicability. Finally, a more rigorous theoretical treatment of the relationship between tokenization, model efficiency, and effectiveness could provide deeper insight into the mechanisms underlying the empirical results.

In conclusion, this dissertation demonstrates that tokenizer-level interventions constitute a practical and computationally efficient pathway for extending large language models to underrepresented languages. By reducing resource requirements while maintaining effectiveness, and by offering concrete strategies for embedding initialization and inference adaptation, the work contributes to the broader effort of making advanced language technologies more inclusive and accessible.

References

[1] OpenAI, accessed: 2025-08-26, Online chat-bot that first shared LLM architecture with the general public. [Online]. Available: https://chatgpt.com/ [Cited on page 1.]

[2] C. Leiter, R. Zhang, Y. Chen, J. Belouadi, D. Larionov, V. Fresen, and S. Eger, "Chatgpt: A meta-analysis after 2.5 months," *Machine Learning with Applications*, vol. 16, p. 100541, 2024. [Cited on page 1.]

[3] M. U. Haque, I. Dharmadasa, Z. T. Sworna, R. N. Rajapakse, and H. Ahmad, ""i think this is the most disruptive technology": Exploring sentiments of chatgpt early adopters using twitter data," 2022. [Online]. Available: https://arxiv.org/abs/2212.05856 [Cited on page 1.]

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020. [Cited on pages 1 and 7.]

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023. [Cited on pages 1 and 7.]

[6] H. Sousa, R. Almeida, P. Silvano, I. Cantante, R. Campos, and A. Jorge, "Enhancing portuguese variety identification with cross-domain approaches," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 24, 2025, pp. 25 192–25 200. [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=d-14Qx4AAAAJ&citation_for_view=d-14Qx4AAAAJ:W7OEmFMy1HYC [Cited on page 1.]

[7] H. Sousa, S. Almasian, R. Campos, and A. Jorge, "Tradutor: Building a variety specific translation model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 24, 2025, pp. 25 183–25 191. [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=d-14Qx4AAAAJ&citation_for_view=d-14Qx4AAAAJ:Y0pCki6q_DkC [Cited on page 1.]

[8] Y. Xia, J. Kim, Y. Chen, H. Ye, S. Kundu, C. C. Hao, and N. Talati, "Understanding the performance and estimating the cost of llm fine-tuning," in *2024 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2024, pp. 210–223. [Cited on pages 1 and 14.]

[9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022. [Cited on pages 1 and 13.]

[10] K. Bostrom and G. Durrett, "Byte pair encoding is suboptimal for language model pretraining," *arXiv preprint arXiv:2004.03720*, 2020. [Cited on pages 1 and 10.]

[11] J. F. Lotz, A. V. Lopes, S. Peitz, H. Setiawan, and L. Emili, "Beyond text compression: Evaluating tokenizers across scales," *arXiv preprint arXiv:2506.03101*, 2025. [Cited on page 1.]

[12] K. Yassin Hussen, W. Tewabe Sewunetie, A. A. Ayele, S. Hafiz Imam, S. H. Muhammad, and S. Muhie Yimam, "The state of large language models for african languages: Progress and challenges," *arXiv e-prints*, pp. arXiv–2506, 2025. [Cited on page 1.]

[13] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury, "The state and fate of linguistic diversity and inclusion in the nlp world," *arXiv preprint arXiv:2004.09095*, 2020. [Cited on page 1.]

[14] J. McGiff and N. S. Nikolov, "Overcoming data scarcity in generative language modelling for low-resource languages: A systematic review," *arXiv preprint arXiv:2505.04531*, 2025. [Cited on page 1.]

[15] K. Huang, F. Mo, X. Zhang, H. Li, Y. Li, Y. Zhang, W. Yi, Y. Mao, J. Liu, Y. Xu *et al.*, "A survey on large language models with multilingualism: Recent advances and new frontiers," *arXiv preprint arXiv:2405.10936*, 2024. [Cited on page 1.]

[16] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," *arXiv preprint arXiv:2402.06196*, 2024. [Cited on page 6.]

[17] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954. [Cited on page 6.]

[18] T. Winograd, “Understanding natural language,” *Cognitive psychology*, vol. 3, no. 1, pp. 1–191, 1972. [Online]. Available: https://doi.org/10.1016/0010-0285(72)90002-3 [Cited on pages 6 and 7.]

[19] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003. [Cited on page 6.]

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013. [Cited on pages 6 and 7.]

[21] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543. [Cited on pages 6 and 7.]

[22] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” *Advances in neural information processing systems*, vol. 28, 2015. [Cited on page 6.]

[23] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642. [Cited on page 6.]

[24] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” 2019. [Online]. Available: https://arxiv.org/abs/1901.02860 [Cited on page 6.]

[25] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990. [Cited on pages 6 and 7.]

[26] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Cited on page 6.]

[27] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014. [Cited on page 6.]

[28] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014. [Cited on pages 6 and 7.]

[29] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994. [Cited on page 6.]

[30] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013. [Cited on page 6.]

[31] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014. [Cited on page 7.]

[32] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International conference on machine learning*. PMLR, 2017, pp. 1243–1252. [Cited on page 7.]

[33] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 562–570. [Cited on page 7.]

[34] Y. Goldberg, "Neural network methods for natural language processing," *Computational Linguistics*, vol. 44, no. 1, pp. 194–195, 2018. [Cited on page 7.]

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. [Cited on page 7.]

[36] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin, "Distilling task-specific knowledge from bert into simple neural networks," *arXiv preprint arXiv:1903.12136*, 2019. [Cited on page 7.]

[37] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202/ [Cited on page 7.]

[38] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, “Transformers: State-of-the-art natural language processing,” in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45. [Cited on page 7.]

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “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)*, 2019, pp. 4171–4186. [Cited on pages 7, 11, and 12.]

[40] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, “Improving language understanding by generative pre-training,” 2018. [Cited on page 7.]

[41] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020. [Cited on page 7.]

[42] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, “Mistral 7b,” 2023. [Online]. Available: https://arxiv.org/abs/2310.06825 [Cited on page 7.]

[43] Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014. [Cited on page 7.]

[44] M. Artetxe, S. Ruder, and D. Yogatama, “On the cross-lingual transferability of monolingual representations,” *arXiv preprint arXiv:1910.11856*, 2019. [Cited on page 8.]

[45] P. Gage, “A new algorithm for data compression,” *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994. [Cited on page 9.]

[46] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” *arXiv preprint arXiv:1508.07909*, 2015. [Cited on page 10.]

[47] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging

the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[48] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018. [Cited on page 10.]

[49] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 5149–5152. [Cited on page 11.]

[50] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019. [Cited on page 12.]

[51] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019. [Cited on page 12.]

[52] R. Lopes, J. Magalhães, and D. Semedo, "Glória – a generative and open large language model for portuguese," 2024. [Online]. Available: https://arxiv.org/abs/2402.12969 [Cited on pages 13, 14, and 18.]

[53] R. Pires, H. Abonizio, T. S. Almeida, and R. Nogueira, "Sabiá: Portuguese large language models," in *Intelligent Systems*, M. C. Naldi and R. A. C. Bianchi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 226–240. [Cited on pages 13 and 14.]

[54] R. Santos, J. Silva, L. Gomes, J. Rodrigues, and A. Branco, "Advancing generative ai for portuguese with open decoder gerv\'asio pt," *arXiv preprint arXiv:2402.18766*, 2024. [Cited on pages 13 and 17.]

[55] J. Rodrigues, L. Gomes, J. Silva, A. Branco, R. Santos, H. L. Cardoso, and T. Osório, "Advancing neural encoding of portuguese with transformer albertina pt," in *EPIA Conference on Artificial Intelligence*. Springer, 2023, pp. 441–453. [Cited on pages 13 and 14.]

[56] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021. [Cited on page 13.]

[57] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020. [Cited on page 13.]

[58] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "Legal-bert: The muppets straight out of law school," *arXiv preprint arXiv:2010.02559*, 2020. [Cited on page 13.]

[59] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," *arXiv preprint arXiv:2004.10964*, 2020. [Cited on page 13.]

[60] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International conference on machine learning*. PMLR, 2019, pp. 2790–2799. [Cited on page 13.]

[61] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee, "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models," *arXiv preprint arXiv:2304.01933*, 2023. [Cited on page 13.]

[62] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021. [Cited on page 13.]

[63] O. Khade, S. Jagdale, A. Phaltankar, G. Takalikar, and R. Joshi, "Challenges in adapting multilingual llms to low-resource languages using lora peft tuning," *arXiv e-prints*, pp. arXiv–2411, 2024. [Cited on page 14.]

[64] S. Kim, S. Choi, and M. Jeong, "Efficient and effective vocabulary expansion towards multilingual large language models," *arXiv preprint arXiv:2402.14714*, 2024. [Cited on page 14.]

[65] I. Nakash, N. Calderon, E. B. David, E. Hoffer, and R. Reichart, "Adaptivocab: Enhancing llm efficiency in focused domains through lightweight vocabulary adaptation," *arXiv preprint arXiv:2503.19693*, 2025. [Cited on page 14.]

[66] F. Fernandez-Martinez, C. Luna-Jiménez, R. Kleinlein, D. Griol, Z. Callejas, and J. M. Montero, "Fine-tuning bert models for intent recognition using a frequency cut-off strategy for domain-specific vocabulary extension," *Applied Sciences*, vol. 12, no. 3, p. 1610, 2022. [Cited on pages 14 and 15.]

[67] P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych, "How good is your tokenizer? on the monolingual performance of multilingual language models," *arXiv preprint arXiv:2012.15613*, 2020. [Cited on pages 15 and 16.]

[68] Z. Csaki, P. Pawakapan, U. Thakker, and Q. Xu, "Efficiently adapting pretrained language models to new languages," *arXiv preprint arXiv:2311.05741*, 2023. [Cited on pages 15, 16, and 30.]

[69] F. Koto, J. H. Lau, and T. Baldwin, "Indobertweet: A pretrained language model for indonesian twitter with effective domain-specific vocabulary initialization," *arXiv preprint arXiv:2109.04607*, 2021. [Cited on page 15.]

[70] S. Gu, M. Zhao, B. Zhang, L. Wang, J. Li, and G. Liu, "Retok: Replacing tokenizer to enhance representation efficiency in large language model," *arXiv preprint arXiv:2410.04335*, 2024. [Cited on page 15.]

[71] M. Artetxe, G. Labaka, and E. Agirre, "A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings," *arXiv preprint arXiv:1805.06297*, 2018. [Cited on pages 15 and 22.]

[72] J. Sanches, R. Ribeiro, and L. Coheur, "From brazilian portuguese to european portuguese," *arXiv preprint arXiv:2408.07457*, 2024. [Cited on page 16.]

[73] C. Carvalho, F. Teixeira, C. Botelho, A. Pompili, R. Solera-Ureña, S. Paulo, M. Julião, T. Rolland, J. Mendonça, D. Pereira *et al.*, "Cam\~ oes: A comprehensive automatic speech recognition benchmark for european portuguese," *arXiv preprint arXiv:2508.19721*, 2025. [Cited on page 16.]

[74] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018. [Cited on page 16.]

[75] □□□, "Exploring the effects of tokenizer extension on domain-specific fine-tuning of language models," Ph.D. dissertation, □□□□□ □□□, 2025. [Cited on page 20.]

[76] G. Dagan, G. Synnaeve, and B. Roziere, “Getting the most out of your tokenizer for pre-training and domain adaptation,” *arXiv preprint arXiv:2402.01035*, 2024. [Cited on page 21.]

[77] A. Chronopoulou, D. Stojanovski, and A. Fraser, “Reusing a pretrained language model on languages with limited corpora for unsupervised nmt,” *arXiv preprint arXiv:2009.07610*, 2020. [Cited on page 22.]

[78] T. Kocmi and O. Bojar, “An exploration of word embedding initialization in deep-learning tasks,” *arXiv preprint arXiv:1711.09160*, 2017. [Cited on page 22.]

[79] K. Dobler and G. De Melo, “Focus: Effective embedding initialization for monolingual specialization of multilingual models,” *arXiv preprint arXiv:2305.14481*, 2023. [Cited on page 22.]

[80] Y. Liu, P. Lin, M. Wang, and H. Schütze, “Ofa: A framework of initializing unseen subword embeddings for efficient large-scale multilingual continued pretraining,” *arXiv preprint arXiv:2311.08849*, 2023. [Cited on page 22.]

[81] B. Minixhofer, F. Paischer, and N. Rekabsaz, “Wechsel: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models,” *arXiv preprint arXiv:2112.06598*, 2021. [Cited on page 22.]

[82] M. Ali, M. Fromm, K. Thellmann, R. Rutmann, M. Lübbering, J. Leveling, K. Klug, J. Ebert, N. Doll, J. Buschhoff *et al.*, “Tokenizer choice for llm training: Negligible or crucial?” in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 3907–3924. [Cited on pages 27 and 30.]

[83] C. Toraman, E. H. Yilmaz, F. Şahi□nuç, and O. Ozcelik, “Impact of tokenization on language models: An analysis for turkish,” *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, no. 4, p. 1–21, Mar. 2023. [Online]. Available: http://dx.doi.org/10.1145/3578707 [Cited on page 27.]

A. Appendix

A.1. Fertility

Referenced on [7.1.3]

New Tokens	Model	Type	Fertility
0	HuggingFaceTB/SmolLM2-135M	Baseline	2.47
1000	HuggingFaceTB/SmolLM2-135M	No Training	1.94
1000	HuggingFaceTB/SmolLM2-135M	With Training	1.94
5000	HuggingFaceTB/SmolLM2-135M	No Training	1.76
5000	HuggingFaceTB/SmolLM2-135M	With Training	1.76
7500	HuggingFaceTB/SmolLM2-135M	No Training	1.75
7500	HuggingFaceTB/SmolLM2-135M	With Training	1.75
0	HuggingFaceTB/SmolLM3-3B	Baseline	1.94
1000	HuggingFaceTB/SmolLM3-3B	No Training	1.77
1000	HuggingFaceTB/SmolLM3-3B	With Training	1.77
5000	HuggingFaceTB/SmolLM3-3B	No Training	1.65
5000	HuggingFaceTB/SmolLM3-3B	With Training	1.65
7500	HuggingFaceTB/SmolLM3-3B	No Training	1.65
7500	HuggingFaceTB/SmolLM3-3B	With Training	1.65
0	Qwen/Qwen2.5-1.5B-Instruct	Baseline	1.93
1000	Qwen/Qwen2.5-1.5B-Instruct	No Training	1.76
1000	Qwen/Qwen2.5-1.5B-Instruct	With Training	1.76
5000	Qwen/Qwen2.5-1.5B-Instruct	No Training	1.66
5000	Qwen/Qwen2.5-1.5B-Instruct	With Training	1.66
7500	Qwen/Qwen2.5-1.5B-Instruct	No Training	1.67
7500	Qwen/Qwen2.5-1.5B-Instruct	With Training	1.67

A.2. Token Ranks Comparison

Referenced on [7.1.5]

A.2.1 Model *HuggingFaceTB/SmolLM2-135M*

A.2.1.1 Number New Tokens = 1000

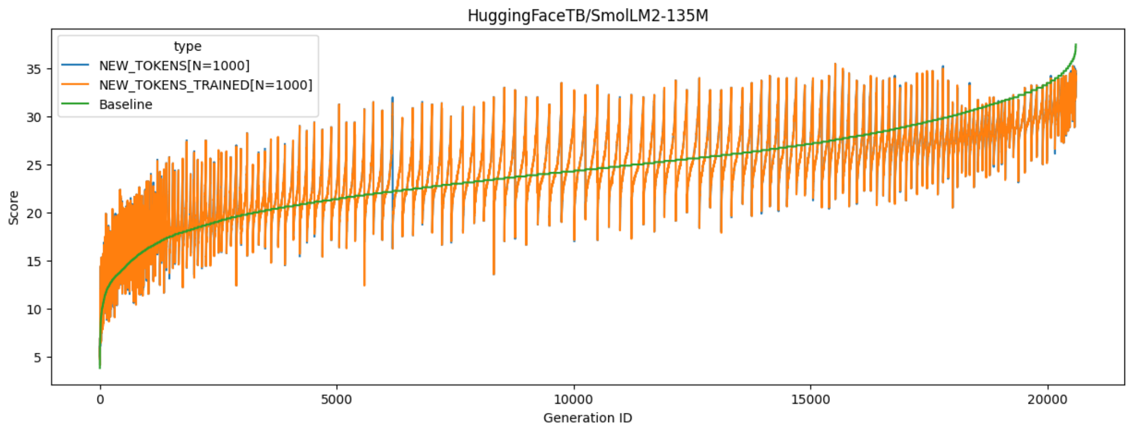

Figure A.1: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.1.2 Number New Tokens = 5000

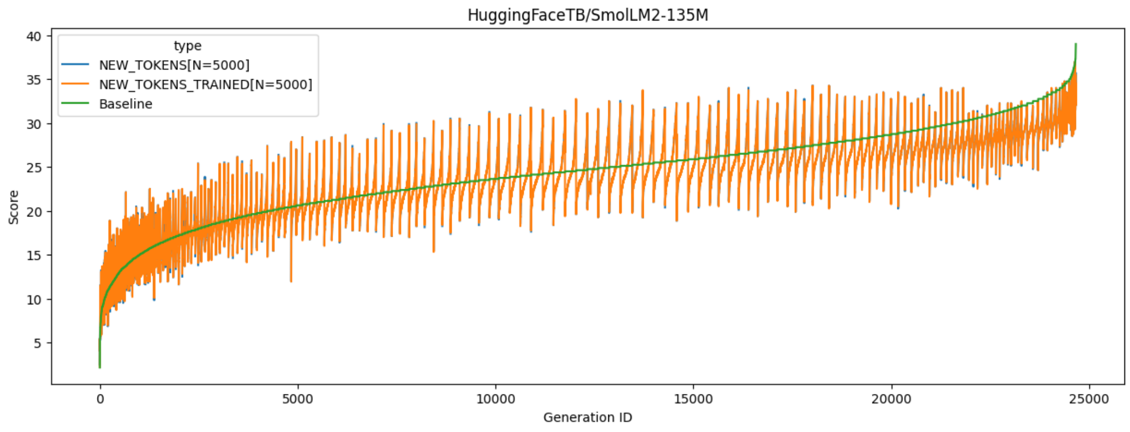

Figure A.2: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.1.3 Number New Tokens = 7500

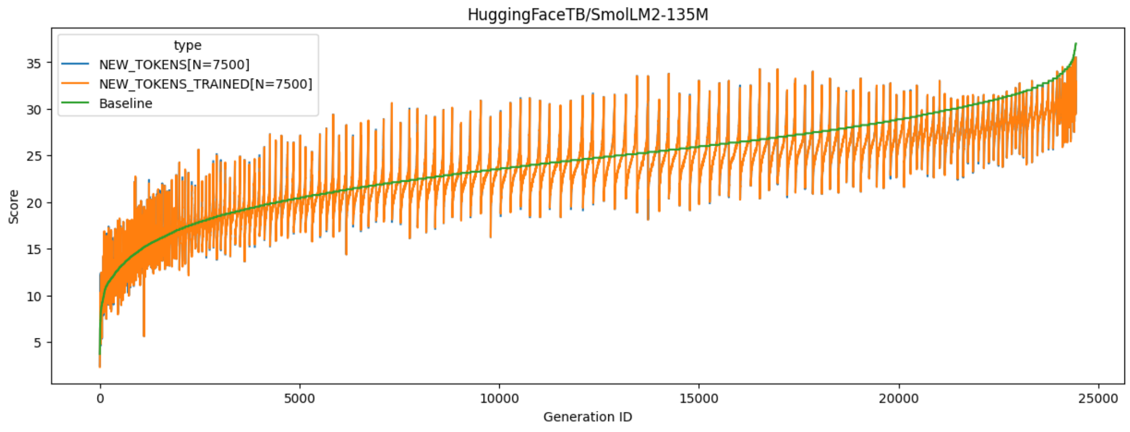

Figure A.3: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.2 Model *Qwen/Qwen2.5-1.5B-Instruct*

A.2.2.1 Number New Tokens = 1000

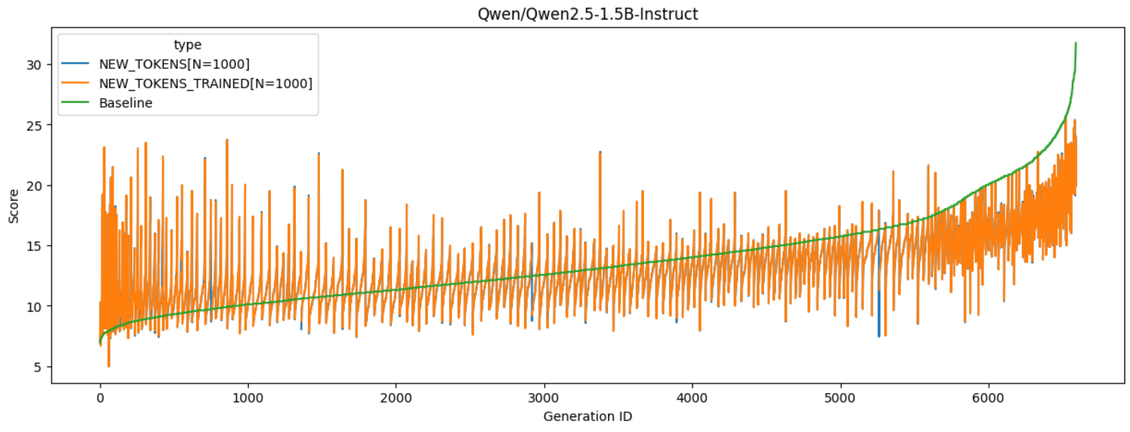

Figure A.4: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.2.2 Number New Tokens = 5000

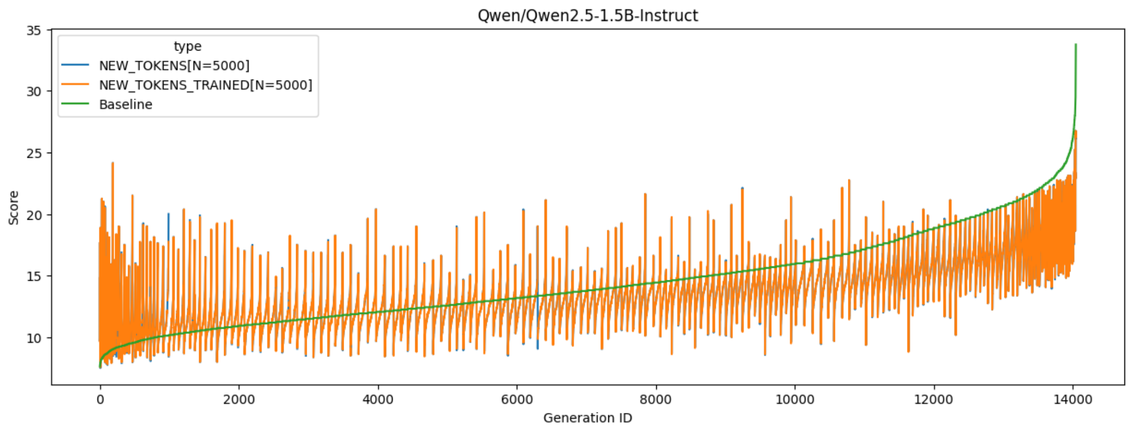

Figure A.5: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.2.3 Number New Tokens = 7500

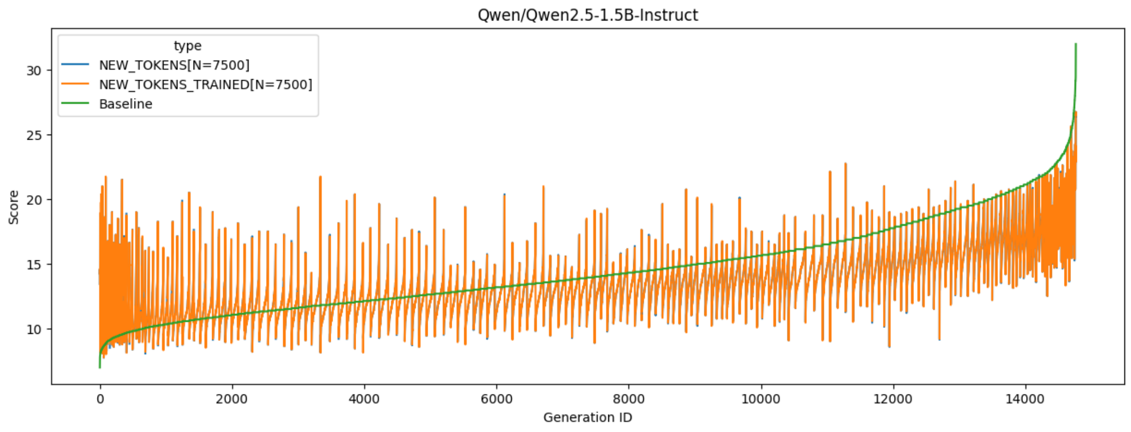

Figure A.6: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.3 Model *HuggingFaceTB/SmolLM3-3B*

A.2.3.1 Number New Tokens = 1000

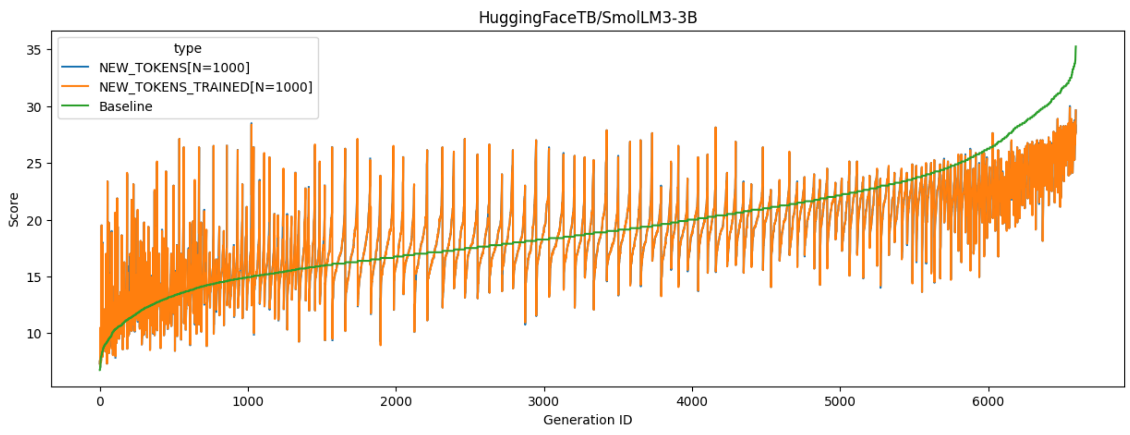

Figure A.7: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.3.2 Number New Tokens = 5000

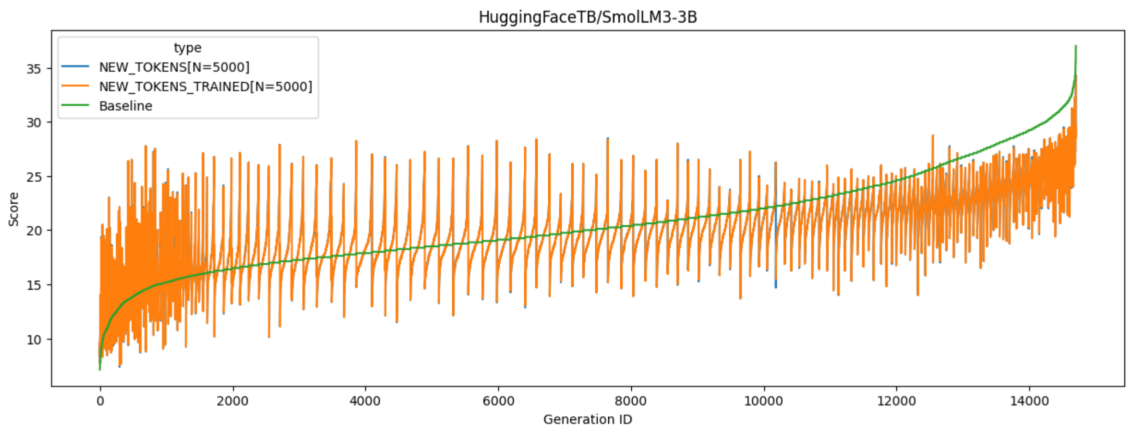

Figure A.8: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.2.3.3 Number New Tokens = 7500

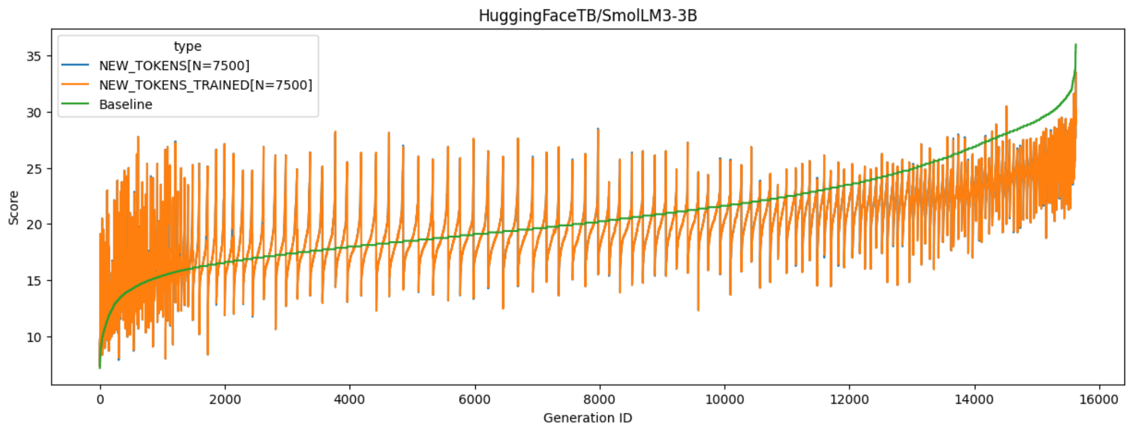

Figure A.9: Score comparison between original sub-token sequences and newly introduced tokens. Higher ranks indicate higher model preference.

A.3. Rank Differences Distribution

Referenced on [7.1.5]

A.3.1 Model *HuggingFaceTB/SmolLM2-135M*

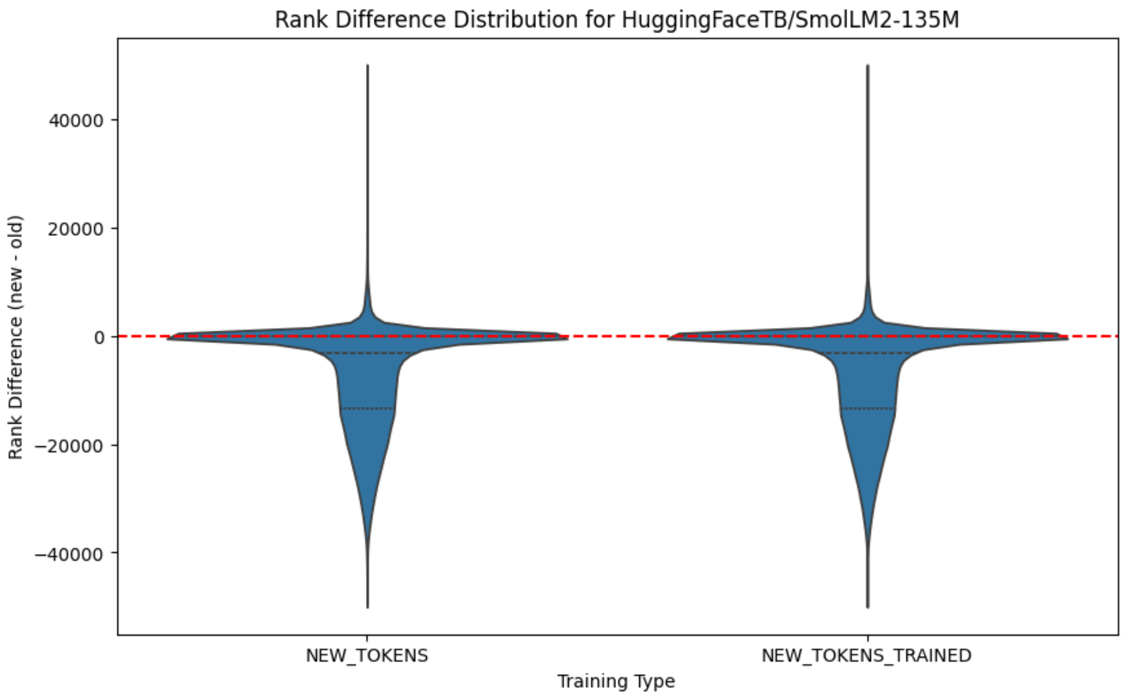

Figure A.10: Distribution of rank differences across the entire results data. Negative numbers show preference towards `new_token`.

A.3.2 Model *Qwen/Qwen2.5-1.5B-Instruct*

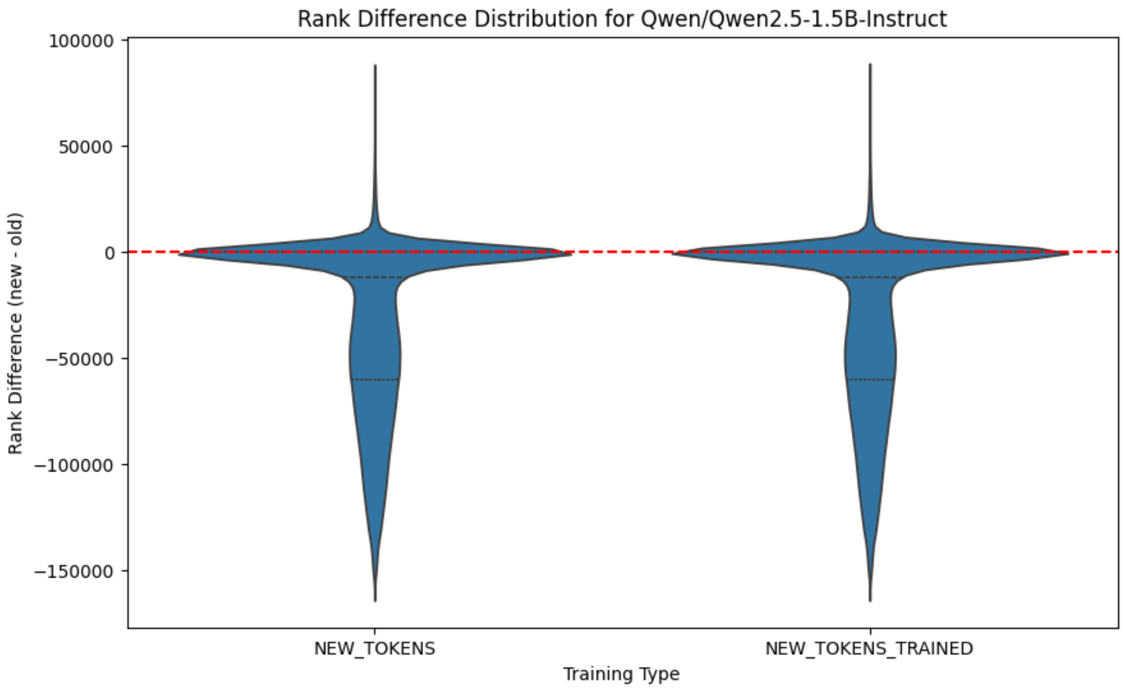

Figure A.11: Distribution of rank differences across the entire results data. Negative numbers show preference towards `new_token`.

A.3.3 Model *HuggingFaceTB/SmolLM3-3B*

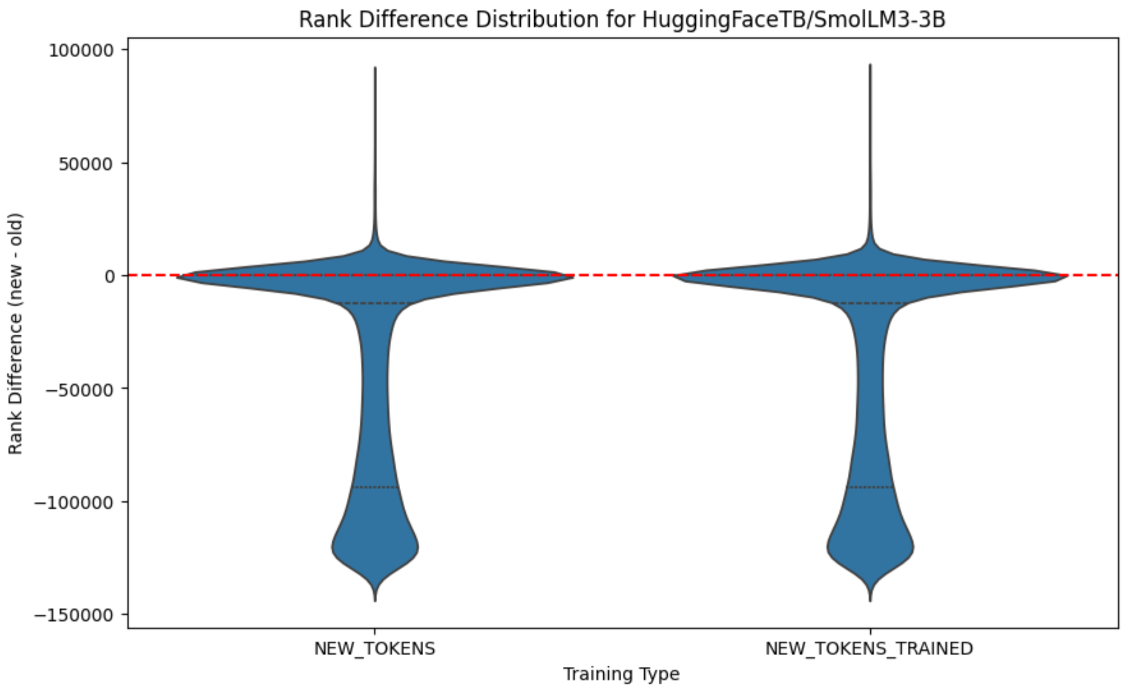

Figure A.12: Distribution of rank differences across the entire results data. Negative numbers show preference towards `new_token`.