Tokens with Meaning: A Hybrid Tokenization Approach for Turkish

M. Ali Bayram[1], **Ali Arda Fincan**[2], **Ahmet Semih Gümüş**[2], **Sercan Karakaş**[3],
Banu Diri[1], Savaş Yıldırım[4], Demircan Çelik[2]
[1]Yıldız Technical University, [2]Yeditepe University, [3]University of Chicago,
[4]Istanbul Bilgi University
malibayram20@gmail.com

Abstract

Tokenization shapes how language models perceive morphology and meaning in Natural Language Processing (NLP), yet widely used frequency-driven subword tokenizers (e.g., Byte Pair Encoding and WordPiece) can fragment morphologically rich and agglutinative languages in ways that obscure morpheme boundaries. We introduce a linguistically informed hybrid tokenizer for Turkish that combines (i) dictionary-driven morphological segmentation (roots and affixes), (ii) phonological normalization that maps allomorphic variants to shared identifiers, and (iii) a controlled subword fallback for out-of-vocabulary coverage. Concretely, our released Turkish vocabulary contains 20,000 root identifiers (with leading spaces to mark word boundaries), 72 affix identifiers that cover 177 allomorphic surface forms, and 12,696 subword units; an orthographic case token preserves capitalization without inflating the vocabulary. We evaluate tokenization quality on TR-MMLU dataset using two linguistic alignment metrics: Turkish Token Percentage (TR %), the proportion of produced tokens that correspond to Turkish lexical/morphemic units under our lexical resources, and Pure Token Percentage (Pure %), the proportion of tokens aligning with unambiguous root/affix boundaries. The proposed tokenizer reaches 90.29% TR % and 85.80% Pure % on TR-MMLU, substantially exceeding several general-purpose tokenizers. We further validate practical utility with downstream sentence embedding benchmarks under a strict *random initialization* control to isolate tokenizer inductive bias. Across four matched models (MFT, Cosmos-GPT2, Mursit, and Tabi), MFT improves Semantic Textual Similarity Benchmark (STSb-TR) Pearson correlation from 33.58% (Tabi) to 50.37%, and achieves the strongest overall average on Massive Text Embedding Benchmark (MTEB-TR) and the strongest TurBLiMP linguistic sensitivity among the evaluated baselines.

Keywords: Tokenization, Morphologically Rich Languages, Morphological Segmentation, Byte Pair Encoding, Turkish NLP, Linguistic Integrity, Low-Resource Languages

1 Introduction

Tokenization is the process of mapping raw text into a sequence of discrete units (tokens) that a model can embed and process. It influences vocabulary construction, sequence length, interpretability, and ultimately performance in downstream tasks [1]. While subword tokenization has become a standard design choice for transformer-based models, its behavior is not neutral for morphologically rich languages.

Byte Pair Encoding (BPE) [2], WordPiece [3], and Unigram [4] address out-of-vocabulary (OOV) words by representing rare forms as compositions of frequent subword units. This improves coverage and keeps vocabularies compact, but it can also split words in ways that cut across morpheme

boundaries and blur grammatical function [5, 6]. Such fragmentation is especially relevant for agglutinative languages such as Turkish, where productive suffixation yields many surface forms from relatively few lemmas.

Turkish exhibits rich suffix morphology and systematic morphophonological alternations, including vowel harmony and consonant alternations at morpheme boundaries. For example, suffix allomorphs such as *-lAr* (plural) and *-dAn*/*-tAn* (ablative) realize the same grammatical morpheme under different phonological contexts, and consonant alternations such as *kitap* $\rightarrow$ *kitabı* (p$\rightarrow$b before a vowel) create predictable surface variants of the same stem. Tokenizers that treat these variants as unrelated units can inflate redundancy and reduce the reuse of meaning-bearing units across inflections [7].

This paper introduces MFT, a linguistically informed hybrid tokenizer for Turkish. The method combines dictionary-driven morphological segmentation (roots and affixes), a normalization layer that maps common allomorphic variants to shared identifiers, and a controlled subword fallback for open-vocabulary coverage. Roots include a leading space to mark word boundaries, and an orthographic case token preserves capitalization without duplicating vocabulary entries.

We evaluate tokenization quality on TR-MMLU [8] using two linguistic alignment metrics: Turkish Token Percentage (TR %) and Pure Token Percentage (Pure %), which quantify lexical/morphemic coverage and alignment with unambiguous root/affix boundaries, respectively [7]. To address reviewer concerns about real-world applicability, we further include downstream evaluation on sentence embedding benchmarks. In a controlled random-initialization setting, we compare four matched models (MFT, CosmosGPT2, Mursit, and Tabi) on Semantic Textual Similarity (STS), MTEB-TR, and TurBLiMP, a Turkish benchmark of linguistic minimal pairs [9].

Our contributions are threefold: we propose a morphology-first hybrid tokenizer for Turkish that is lossless via an explicit decoder; we provide a quantitative and qualitative evaluation of tokenization quality on TR-MMLU against widely used tokenizers; and we report controlled downstream comparisons across four matched embedding models to assess whether improved morpheme alignment translates into better sentence representations.

2 Related Work

Tokenization is a fundamental step in NLP, significantly impacting model performance, memory efficiency, and downstream task effectiveness. Tokenization strategies range from character-level segmentation to subword-based methods such as BPE [2], WordPiece [3], and Unigram [10]. The choice of tokenization directly influences the ability of models to capture syntactic, semantic, and morphological structures, especially in morphologically rich languages such as Turkish, Finnish, and Hungarian [11, 5].

Recent research has explored alternative tokenization strategies tailored to morphologically rich languages. Toraman et al. [5] analyze the impact of tokenization on Turkish language modeling, and report that morphology-aware tokenization can recover much of the performance of larger baselines under certain settings. Kaya and Tantuğ [6] examine tokenization granularity for Turkish language models, and highlight that Turkish can require substantially more subword splits per word than English under common subword tokenizers, underscoring the importance of vocabulary design and sequence length control.

Tokenization strategies also play a crucial role in machine translation and text generation tasks. Pan et al. [12] demonstrate that morphology-aware segmentation can reduce sparsity in neural machine translation, and Huck et al. [13] study target-side segmentation strategies that improve translation quality by maintaining linguistic consistency between source and target languages. Beyond translation, morphology-aware tokenization has also been evaluated in abstractive summarization and sentiment analysis. Baykara and Güngör [11] discuss summarization for agglutinative languages, and Kayalı and Omurca [14] propose a hybrid tokenization strategy for Turkish summarization. Such hybrid approaches are also commonly motivated by applications where preserving linguistic structure is important (e.g., named entity recognition (NER)).

Tokenization quality is also discussed in the context of modern LLM tokenizers, where differences in segmentation can affect non-English text processing and evaluation outcomes. Bayram et al. [7] compare several widely used tokenizers on Turkish and highlight how tokenizer-specific segmentation artifacts can influence downstream benchmarking.

Despite these advancements, the computational cost of tokenization and its interaction with training efficiency remains an open concern. Larger vocabularies can increase model size and memory footprint [15, 1], and the energy and carbon footprint of training large models has motivated more careful reporting and efficiency analysis [16]. From this perspective, tokenization is not only a linguistic design choice, but also a practical lever that affects sequence length and compute; inefficient vocabulary utilization and redundant segmentation can translate into longer sequences and higher training cost [16].

To address trade-offs between linguistic alignment and efficiency, recent work has explored adaptive and multilingual tokenization strategies. Martins et al. [17] describe multilingual language models and tokenization choices across European languages, and Lin et al. [18] study token selection strategies that question whether all tokens contribute equally during pretraining. Dynamic tokenization approaches that adapt segmentation rules have also been proposed; for example, Neubeck et al. [19] explore a more flexible BPE-style tokenizer.

Several approaches incorporate linguistic structure directly into tokenization. Hofmann et al. [20] show that derivationally informed segmentation can improve model interpretation of complex word forms. MorphPiece [21] segments by morphemes before applying a subword encoding step, aiming to preserve compositional meaning while remaining compatible with standard training pipelines. Closest to our design are hybrid tokenizers that combine explicit linguistic resources with statistical fallback. miLLi [22] is a tokenizer for Azerbaijani that uses a root dictionary, BPE fallback, and a phonological restoration mechanism to increase root consistency across surface variants. Another line of work modifies the subword algorithm itself to better respect morphological structure: MorphBPE [23] extends BPE with morphology-aware constraints and introduces morphology-based evaluation metrics, reporting improved morphological alignment and training behavior across multiple languages. While these morphology-aware tokenizers share design motivations with our approach, they target different languages (Azerbaijani, Arabic, and multilingual corpora) and are therefore not directly comparable on Turkish-specific benchmarks without substantial adaptation.

Tokenization strategies play a critical role in pretraining large language models (LLMs), influencing model efficiency, generalization, and performance across downstream tasks. Transformer-based architectures such as BERT [15], RoBERTa [1], and GPT [24] rely on effective tokenization to balance vocabulary size, sequence length, and computational cost. Studies have shown that tokenization choices can interact with morphological compositionality and generalization, particularly for morphologically rich languages [25].

Benchmark evaluations such as Massive Multitask Language Understanding (MMLU) [26] and TR-MMLU [8] have highlighted the need for language-aware evaluation. Bayram et al. [7] propose a linguistic integrity framework for evaluating Turkish tokenization, introducing metrics such as token purity and Turkish Token Percentage (TR %). TR % measures the proportion of produced tokens that correspond to valid Turkish lexical or morphemic units under curated lexical resources, while Pure % further requires that tokens align with unambiguous morpheme boundaries rather than arbitrary substrings. These metrics are computed using an external morphological validator (a curated root/affix inventory independent of the tokenizer under evaluation), which ensures fair cross-tokenizer comparison and avoids circularity. Their results suggest that higher TR % and purity correlate with stronger performance on MMLU-style Turkish benchmarks, motivating our focus on morpheme-level alignment.

Turkish-specific benchmarks and evaluation suites have expanded rapidly. TR-MMLU [8] provides a large-scale Turkish evaluation set for language model assessment, and TurBLiMP [9] offers a controlled benchmark of linguistic minimal pairs covering diverse phenomena. In parallel, Turkish-focused model and tokenizer ecosystems continue to grow. For example, TabiBERT [27] provides a modern Turkish encoder and a unified evaluation suite, reinforcing the value of language-specific baselines when assessing tokenizer behavior and downstream impact.

Finally, tokenization considerations extend beyond language modeling into applied pipelines such as optical character recognition and document parsing. Rashad et al. [28] demonstrate that tokenizer choices can affect structure reconstruction and recognition accuracy in Arabic document processing, and Rosa et al. [29] provide a tokenizer benchmark in a multilingual setting, illustrating that tokenizer behavior can vary widely across languages and domains.

3 Methodology

We propose a hybrid tokenization framework that combines linguistic knowledge with statistical subword segmentation. This approach, the *Morphology-First Tokenizer* (MFT), integrates rule-based morphological analysis with a structured dictionary of roots and affixes while incorporating BPE to handle out-of-vocabulary (OOV) terms. The objective is to create a tokenization system that accurate represents linguistic structures while maintaining computational efficiency.

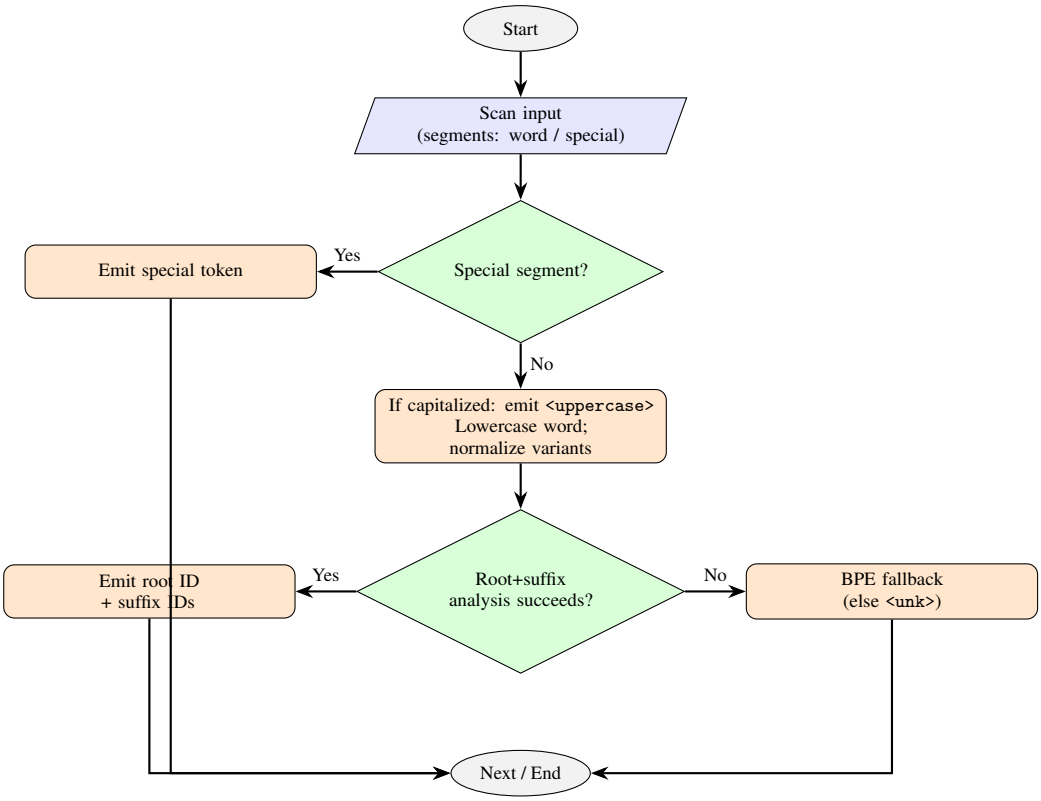

Figure 1: Algorithmic flow of the MFT tokenization pipeline.

We provide a Python reference implementation of the tokenizer and release the lexical resources (root and affix inventories) and decoder rules used in our experiments.

Dictionary Construction. The core of MFT is a dual-dictionary system designed to cover the productive morphology of Turkish.

Root Dictionary. The root dictionary is constructed from high-frequency words extracted from large-scale Turkish corpora, comprising approximately 22,000 roots. To address the challenge of phonological alternation, we employ a normalization strategy where surface variants map to a single canonical root ID. For example, consonant alternation causes *kitap* (book) and *kitabı* (its book) to share the same root ID despite the $p \rightarrow b$ softening; vowel hiatus maps *oyna* (play) and *oynuyor* (playing, where 'a' drops) to a unified token; and haplology treats *alın* (forehead) and *alnı* (his forehead) identically. We also explicitly tokenize frequent compound words (e.g., *akarsu* 'stream', *çamaşırhane* 'laundromat') as single units to prevent erroneous splitting.

Affix Dictionary. The affix inventory consists of 177 suffix surface forms consolidated into 72 abstract affix IDs (covering grammatical morphemes such as case markers, tense suffixes, and derivational endings). Similar to roots, we merge allomorphs that serve identical grammatical functions into shared IDs. For instance, the plural suffix *-ler* and its harmonic variant *-lar* are assigned a single token ID (e.g., PL), as are the ablative variants *-den, -dan, -ten, -tan*. This abstraction reduces vocabulary redundancy while preserving the morphosyntactic signal.

Input Normalization and Special Tokens. To ensure robustness across diverse text inputs, we implement strict normalization rules. For case handling, we introduce an `<uppercase>` token to mark capitalized words, allowing the model to process *Kitap* and *kitap* using the same root embedding and effectively halving the number of required surface forms. For CamelCase splitting, technical terms and code-switching often introduce CamelCase (e.g., 'HTTPServer), which we split into constituent parts ('HTTP, 'Server) before tokenization to improve subword coverage. Word boundaries are marked by including a leading space in root tokens, ensuring that tokenization is lossless and reversible without a dedicated whitespace token.

Encoding Algorithm. The encoding process (Algorithm 1) follows a "longest-prefix match" strategy. For each word, the tokenizer first attempts to identify a valid root from the dictionary. If a root is found, it greedily matches the longest chain of valid suffixes.

Algorithm 1 Morphology-First Tokenization Pipeline

Input: Raw text string S
Output: Sequence of Token IDs T
$S \leftarrow$ Preprocess(S) ▷ Insert spaces, split CamelCase
for each word w in S **do**
  if w is Space or Punctuation **then**
    T.append(GetSpecialID(w))
    continue
  end if
  if w is Capitalized **then**
    T.append($\text{ID}_{\text{uppercase}}$)
    $w \leftarrow w$.lower()
  end if
  $root, suffixes \leftarrow$ MorphAnalyze(w) ▷ Greedy Dictionary Search
  if $root \neq$ None **then**
    T.append($root$.id)
    for s in $suffixes$ **do**
      T.append(s.id)
    end for
  else
    $subwords \leftarrow$ BPE(w) ▷ Fallback
    T.extend($subwords$)
  end if
end for
Return T

If the morphological analyzer fails to cover the word (i.e., no valid root+suffix combination is found), the system falls back to a BPE model. This ensures that the tokenizer remains open-vocabulary and can handle foreign entities or neologisms. The BPE model is trained on a version of the corpus where known morphological segments are masked, focusing its vocabulary (12,696 tokens) on residual stems and subwords.

Example. Consider the validation sentence: "*Kalktığımızda hep birlikte yürüdük.*" (When we stood up, we walked together.)

> `<upper>` **kalk-tığ-ımız-da _ hep _ birlik-te _ yürü-dü-k .**: CAPS stand-NMZ-POSS.1PL-LOC _ always _ unity-LOC _ walk-PST-1PL .
> *When we stood up, we walked together.*
> 'Kalktığımızda hep birlikte yürüdük.'

Here, the tokenizer correctly identifies the root *kalk* (stand) and segments the complex nominalization chain *-tık-ımız-da* (represented by phonologically normalized abstract suffixes).

Decoding Algorithm. Decoding in MFT is non-trivial compared to standard subword tokenizers. Simple concatenation is insufficient due to the normalization of affixes. The decoder (Algorithm **??**) applies phonological rules to reconstruct the correct surface form by selecting the appropriate allomorphic variant for each suffix based on context.

Algorithm 2 MFT Decoding Pipeline

Input: Sequence of Token IDs T
Output: Reconstructed text string S
parts $\leftarrow$ []; $i \leftarrow 0$
while $i < |T|$ **do**
    $id \leftarrow T[i]$
    if $id = \text{ID}_{\text{uppercase}}$ **then**
        parts.append(TurkishCapitalize(Lookup($T[i+1]$)))
        $i \leftarrow i + 2$; **continue**
    end if
    candidates $\leftarrow$ ReverseLookup(id) ▷ Surface form variants
    if |candidates| > 1 **then**
        ctx $\leftarrow$ GetVowelContext(parts) ▷ Look back for vowel
        surface $\leftarrow$ ApplyPhonology(id, ctx, T, i)
    else
        surface $\leftarrow$ candidates[0]
    end if
    parts.append(surface); $i \leftarrow i + 1$
end while
Return parts.join(“”)

The `ApplyPhonology` function implements five Turkish morphophonological rules:

1. **Vowel harmony (front/back):** Suffix vowels match the frontness of the last vowel (e.g., *-ler* after *ev* but *-lar* after *çocuk*; front vowels: e,i,ö,ü; back vowels: a,ı,o,u)
2. **Consonant assimilation:** Initial $d \rightarrow t$ after voiceless consonants (e.g., *-da* $\rightarrow$ *-ta* after *kitap*; voiceless: f,s,t,k,ç,ş,h,p)
3. **Consonant softening:** Final $p \rightarrow b$, $k \rightarrow$, $t \rightarrow d$, $\rightarrow c$ before vowel-initial suffixes (e.g., *kitap* $\rightarrow$ *kitab-ı*)
4. **Vowel narrowing:** Stem-final $e \rightarrow i$, $a \rightarrow$ ß before progressive *-yor* (e.g., *de-* $\rightarrow$ *di-yor*, *başla-* $\rightarrow$ *başlı-yor*)
5. **Buffer consonant insertion:** $y/n/s$ inserted between vowels (e.g., *okuma* + ACC $\rightarrow$ *okuma-y-ı*)

Example.

> `<upper>` **kitap _ okuma-yı _ sev-iyor-um .**: CAPS book _ reading-ACC _ love-PROG-1SG .
> *I like reading books.*
> ‘Kitap okumayı seviyorum.’

In this example, the abstract accusative suffix (represented as ACC in the vocabulary) is realized as *-yı* after *okuma* due to buffer consonant insertion (rule 5) and vowel harmony (rule 1).

Roundtrip Reconstruction Evaluation. To validate the reconstruction property of the decoder, we evaluate word-level roundtrip accuracy on 500 words randomly sampled from our training corpus. Given an input word w, we compute $\hat{w} = \text{decode}(\text{encode}(w))$ and measure exact-match accuracy. The decoder achieves **99.2% exact-match accuracy** (496/500 words). The remaining 0.8% failures arise from inherent ambiguity in Turkish phonology: vowel alternation patterns in complex verb forms (e.g., *istiyorduk* $\rightarrow$ *isteyorduk*) where multiple surface realizations are linguistically valid for the same morpheme sequence. This is not a limitation of the tokenizer but rather reflects the intrinsic many-to-one mapping in Turkish morphophonology, where the same abstract morpheme can surface differently depending on context.

Crucially, this near-lossless reconstruction property enables MFT to be used across all transformer architectures: **encoder-only models** (e.g., BERT-style embeddings), **encoder-decoder models** (e.g., translation, summarization), and **decoder-only models** (e.g., GPT-style generation). For encoder-only tasks, exact reconstruction is not required since the model operates on embeddings rather than

regenerating text. For generative tasks, the 99.2% accuracy ensures that the vast majority of outputs are orthographically correct, with the rare exceptions being phonologically valid Turkish variants.

Tokenization Efficiency. We benchmark tokenization speed and token density on 1,000 Turkish text samples from our corpus. Table 1 compares MFT against three Turkish-trained baseline tokenizers under matched vocabulary size (32,768).

Table 1: Tokenization efficiency comparison (1,000 texts, 653K words).

Tokenizer	Time (ms)	Tokens	Tok/Word	Tok/Char
MFT	1,935	1,899,670	**2.91**	0.356
Tabi	1,544	1,298,725	1.99	0.244
Mursit	1,655	1,187,418	1.82	0.223
Cosmos	1,620	1,186,834	1.82	0.223

MFT produces approximately 1.5× more tokens than BPE-based baselines because it segments at morpheme boundaries (e.g., *kitaplarımızdan* → 4 tokens: *kitap* + *lar* + *ımız* + *dan*) rather than optimizing for compression. While this increases sequence length, the morpheme-aligned representations yield improved downstream performance: +16.8 pp on STSb-TR correlation and +7.8 pp on retrieval tasks compared to the strongest baseline. The trade-off favors semantic quality over sequence efficiency—consistent with our hypothesis that morphological alignment improves sample efficiency during embedding training.

4 Results and Analysis

The performance of the proposed morphological tokenizer was evaluated using the TR-MMLU benchmark dataset, which comprises over 1.6 million characters and approximately 200,000 words curated specifically for Turkish [8]. This dataset is designed to reflect the linguistic complexity of Turkish, including its rich morphology, agglutinative structures, and diverse syntactic constructions. As such, it provides a rigorous basis for assessing tokenization quality in morphologically complex languages.

The evaluation compared five different tokenizers: `google/gemma-2-9b`, `meta-llama/Llama-3.2-3B`, `Qwen/Qwen2.5-7B-Instruct`, `CohereForAI/aya-expanse-8b`, and the proposed `turkish_tokenizer`. Each tokenizer was assessed using a consistent set of linguistic and computational metrics introduced in [7]. These metrics include total token count, vocabulary size, number of unique tokens, Turkish Token Percentage (TR %), and Pure Token Percentage (Pure %). TR % quantifies the proportion of tokens that correspond to valid Turkish words or morphemes, while Pure % measures the proportion of tokens that fully align with unambiguous root or affix boundaries, thus reflecting morphological integrity. Importantly, TR % and Pure % are computed using an independent morphological validator with curated lexical resources external to the tokenizer under evaluation, following the protocol of [7], which ensures fair comparison and avoids circularity.

Table 2: Performance of the proposed `turkish_tokenizer` on the TR-MMLU dataset.

Metric	Value
Vocabulary Size	32,768
Total Token Count	707,727
Processing Time (s)	0.6714
Unique Token Count	11,144
Turkish Token Count	10,062
Turkish Token Percentage (TR %)	90.29%
Pure Token Count	9,562
Pure Token Percentage (Pure %)	85.80%

The proposed `turkish_tokenizer` demonstrated the highest linguistic alignment across all evaluated metrics. It achieved a TR % of 90.29% and a Pure % of 85.80%, substantially outperforming

all competing tokenizers. In comparison, `google/gemma-2-9b` reached a TR % of only 40.96% and a Pure % of 28.49%, indicating that the majority of its tokens do not represent full morphemes. Similarly, `meta-llama/Llama-3.2-3B` produced a TR % of 45.77% and a Pure % of 31.45%, while `Qwen2.5` and `aya-expanse` achieved TR % values of 40.39% and 53.48%, respectively.

Despite employing significantly smaller vocabulary sizes, the proposed tokenizer demonstrated better linguistic segmentation. With a vocabulary of 32,768 tokens and 11,144 unique tokens used during evaluation, it balanced generalization and expressiveness more effectively than models such as `gemma-2-9b` and `aya-expanse`, which rely on vocabularies of over 255,000 tokens. These large-vocabulary tokenizers, rooted in frequency-based subword segmentation, tend to fragment morphologically rich expressions and introduce ambiguity in downstream tasks. In contrast, the morphological awareness of the `turkish_tokenizer` enables semantically coherent token formation and more consistent syntactic parsing.

Although the total token count generated by the proposed tokenizer (707,727) exceeds those of the other models-for instance, `aya-expanse` produced 434,526 tokens-this increase is offset by gains in interpretability and linguistic fidelity. High TR % and Pure % scores suggest reduced reliance on spurious subword splits and improved preservation of morphosyntactic structure. This is particularly beneficial for tasks such as syntactic parsing, translation, summarization, and question answering, where semantic consistency across tokens is essential.

These findings support the hypothesis introduced in [7], which argues that high linguistic alignment in tokenization correlates strongly with downstream model performance in morphologically rich and low-resource languages. While conventional subword tokenizers may suffice for high-resource languages like English, they exhibit clear limitations in Turkish unless informed by morphological structure. The results presented here highlight the effectiveness of combining rule-based linguistic analysis with subword strategies to produce tokenizers that are both accurate and efficient in morphologically complex settings.

To illustrate the linguistic fidelity of different tokenization strategies, we present a qualitative comparison using the Turkish sentence:

"Atasözleri geçmişten günümüze kadar ulaşan anlamı bakımından mecazlı bir mana kazanan kalıplaşmış sözlerdir."
("Proverbs are fixed expressions passed down from the past to the present that acquire a metaphorical meaning in terms of their significance.")

This sentence contains a wide range of morphological features, including compound words, multiple derivational and inflectional suffixes, and root forms that undergo phonological alternations. These properties make it an ideal test case for evaluating the morphological sensitivity of different tokenizers.

Proposed Hybrid Tokenizer:
The hybrid morphological tokenizer segments the sentence into linguistically meaningful units with high fidelity. It produces:
`["<uppercase>", " atasöz", "ler", "i", " geçmiş", "ten", " günü", "m", "üz", "e", " kadar", " ulaş", "an", " anlam", "ı", " bakım", "ın", "dan", " mecaz", "lı", " bir", " mana", " kazan", "an", " kalıp", "laş", "mış", " sözle", "r", "dir", "."]`
It correctly separates suffixes (`"ler"`, `"i"`, `"ın"`, `"dan"`, `"lı"`, `"an"`, `"mış"`, `"dir"`), extracts root forms such as `"atasöz"`, `"gün"`, `"mana"` with leading spaces to mark word boundaries, and employs the `"<uppercase>"` token to preserve orthographic case.

Tested newmindaiMursit Tokenizer:
The hybrid morphological tokenizer segments the sentence into linguistically meaningful units with high fidelity. It produces:
`["At", "asöz", "leri", " geçmişten", " günümüze", " kadar", " ulaşan", " anlamı", " bakımından", " mec", "azlı", " bir", " man", "a", " kazanan", " kalıp", "laşmış", " söz", "lerdir", "."]`
Although it trained with Turkish text still does not capture most of suffixes, it fragments common roots (`"At", "asöz"` instead of `"atasöz"`) and fails to isolate inner morphemes in forms such as `"lerdir"` and `"kazanan"`, limiting morphological interpretability.

Tested cosmosGPT2 Tokenizer:
The hybrid morphological tokenizer segments the sentence into linguistically meaningful units with high fidelity. It produces:
`["At", "as", "öz", "leri", " geçmişten", " günümüze", " kadar", " ulaşan", " anlamı", " bakımından", " mec", "az", "lı", " bir", " man", "a", " kazanan", " kalıp", "laşmış", " söz", "lerdir", "."]`
Although it trained with Turkish text still does not capture most of suffixes, it fragments common roots (`"At", "as", "öz"` instead of `"atasöz"`) and fails to isolate inner morphemes in forms such as `"lerdir"` and `"kazanan"`, limiting morphological interpretability.

Tested tabi Tokenizer:
The hybrid morphological tokenizer segments the sentence into linguistically meaningful units with high fidelity. It produces:
`["A", "tasöz", "leri ", "geçmişten ", "günümüze kadar ", "ulaşan ", "anlamı ", "bakımından ", "mec", "az", "lı bir ", "mana ", "kazanan ", "kalıp", "laşmış", " söz", "lerdir", "."]`
Although it trained with Turkish text still does not capture most of suffixes, it fragments common roots (`"A", "tasöz"` instead of `"atasöz"`) and fails to isolate inner morphemes in forms such as `"lerdir"` and `"kazanan"`, limiting morphological interpretability.

Gemma-3:
The tokenizer `google/gemma-3` segments the sentence as:
`["<bos>", "At", "as", "öz", "leri", " geçmiş", "ten", " gün", "ümü", "ze", " kadar", " ulaş", "an", " anlam", "ı", " bakım", "ından", " mec", "az", "lı", " bir", " mana", " kaz", "anan", " kal", "ı", "pla", "ş", "mış", " söz", "lerdir", "."]`
Although it captures some suffixes like `"ten"` and `"ından"`, it fragments common roots (`"At", "as", "öz"` instead of `"atasöz"`) and fails to isolate inner morphemes in forms such as `"lerdir"` and `"kazanan"`, limiting morphological interpretability.

LLaMA-3.2:
The tokenizer `meta-llama/Llama-3.2-3B` yields:
`["<|begin_of_text|>", "At", "as", "öz", "leri", " geçmiş", "ten", " gün", "ümü", "ze", " kadar", " ", "ula", "ş", "an", " anlam", "ı", " bakımından", " me", "ca", "z", "lı", " bir", " mana", " kaz", "anan", " kal", "ı", "pla", "ş", "mış", " söz", "lerdir", "."]`
This tokenizer combines morphologically valid segments like `"bakımından"` and `"kazanan"` with fragmented roots like `"At", "as", "öz"`, creating inconsistency in morpheme alignment.

YTU Turkish GPT-2 (without pruned vocab to 32k):
The tokenizer `ytu-ce-cosmos/turkish-gpt2-large-750m-instruct-v0.1`, trained on Turkish corpora, yields:
`["At", "as", "öz", "leri", " geçmişten", " günümüze", " kadar", " ulaşan", " anlamı", " bakımından", " mec", "az", "lı", " bir", " mana", " kazanan", " kalıp", "laşmış", " söz", "lerdir", "."]`
Although it still segments `"atasözleri"` incorrectly, it performs well with forms like `"geçmişten"`, `"günümüze"`, and `"bakımından"`, showing the advantage of Turkish-specific pretraining.

GPT-4o:
The tokenizer `gpt-4o-o200k_base` generates:
`["At", "as", "öz", "leri", " geçmiş", "ten", " gün", "ümü", "ze", " kadar", " ulaş", "an", " anlam", "ı", " bakım", "ından", " mec", "az", "lı", " bir", " mana", " kaz", "anan", " kal", "ı", "pla", "ş", "mış", " söz", "ler", "dir", "."]`
Its segmentation strategy is similar to LLaMA and Qwen-partially aware of Turkish morphemes but limited by frequent over-segmentation of compound and derived forms.

The results presented in this section provide strong empirical support for the hypothesis introduced in the introduction: tokenizers that explicitly incorporate morphological and phonological knowledge of Turkish can outperform general-purpose models in both segmentation accuracy and linguistic coherence. While most state-of-the-art tokenizers struggle with root-fragmentation, over-segmentation, and inconsistent affix treatment, the proposed hybrid tokenizer consistently identifies morpheme boundaries, preserves semantically meaningful units, and reduces vocabulary redundancy. These findings validate the motivation behind this work: morphologically informed tokenization is essential for robust and interpretable NLP in agglutinative languages like Turkish. The qualitative comparisons presented here illustrate not only the performance gap between general and language-specific tokenizers, but also the need for tokenizer architectures that respect language-internal rules.

Downstream Task Evaluation.

To assess the impact of morphologically informed tokenization on downstream model performance, we evaluated the embeddings produced by models initialized with different tokenizers using three benchmarks: STSb-TR, MTEB-TR, and TurBLiMP. All models were initialized randomly to isolate the effect of tokenization structure from pre-training data.

Concretely, we construct four sentence embedding models that share the same encoder architecture (`google/embeddinggemma-300m`) and vocabulary size (32,768). Each model is randomly initialized with a fixed seed (42) and trained under an identical embedding-distillation objective; the only difference between models is the tokenizer and the corresponding pre-encoded `input_ids` column used during training. We refer to these models as `*-random-init` to emphasize that they start from random weights rather than a pretrained checkpoint, so downstream differences primarily reflect inductive bias introduced by tokenization and decoding choices under a controlled training budget.

Training data comes from a Turkish text corpus with pre-computed teacher embeddings (`alibayram/cosmos-corpus-0-05-with-embeddings`). We prepare a unified dataset (`alibayram/cosmos-corpus-encoded`) that stores token ID sequences for all compared tokenizers (`mft_input_ids`, `tabi_input_ids`, `cosmos_input_ids`, `mursit_input_ids`). To ensure an apples-to-apples comparison under a fixed context window, we discard any sample for which *any* tokenizer produces a sequence longer than 2048 tokens, so that no model benefits from truncation artifacts or sees different content due to length differences.

Because MFT is implemented as a custom Python tokenizer rather than a standard HuggingFace tokenizer, we use pre-encoded `input_ids` and a lightweight "tokenizer bypass" patch in the `sentence-transformers` training stack so the same trainer loop can consume all four token streams without rewriting the model code. All downstream evaluations are run with fixed scripts: `evaluate_sts_tr.py` for STS, `mteb-tr/mteb_tr_cli.py` for MTEB-TR, and `evaluate_turblimp.py` for TurBLiMP.

Table 3: Embedding distillation experiment setup.

Component	**Specification**
Student architecture	`google/embeddinggemma-300m` (SentenceTransformer)
Initialization	Random weights with fixed seed 42; vocab resized to 32,768
Training objective	Cosine embedding loss against teacher vectors
Teacher model	`intfloat/multilingual-e5-large-instruct`
Training corpus	`alibayram/cosmos-corpus-0-05-with-embeddings`
Training dataset	`alibayram/cosmos-corpus-encoded` with 4 `input_ids` columns
Context length	2048; samples dropped if any tokenizer exceeds the limit
Batch size / LR	256 / 5×10^{-5}
Schedule	Two-phase: 100-step warmup then 1 full epoch
Precision	BF16; gradient checkpointing enabled
Hardware	NVIDIA H100 80GB (single node)

For STS, we use the Turkish STSb-TR benchmark (`figenfikri/stsb_tr`) consisting of sentence pairs with human similarity ratings on a 0–5 scale [30]. Each model encodes both sentences, we compute cosine similarity between the resulting sentence embeddings, and we report Pearson and

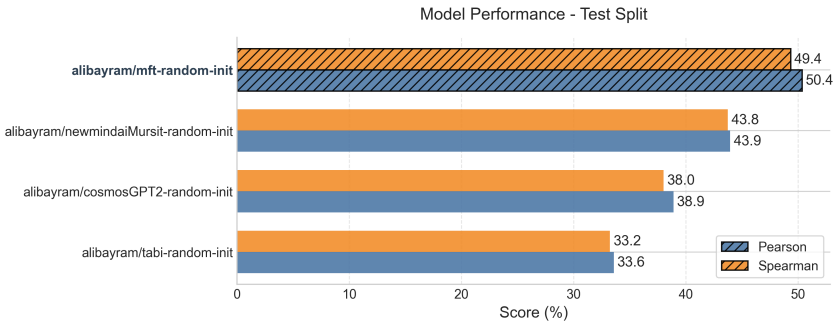

Figure 2: STS benchmark test split performance.

Spearman correlation with the normalized gold scores. Throughout this section, correlations are presented as percentages ($\times 100$) for readability.

We evaluated the models on the Turkish STS benchmark (stsb-tr) without task-specific fine-tuning. The proposed `mft-random-init` model achieved a significantly higher correlation with human judgments compared to other randomly initialized baselines, demonstrating that its structural prior provides a better starting point for capturing semantic similarity.

On the STSb-TR test split ($n = 1379$), the `mft-random-init` model achieved a Pearson correlation of **50.37%** and Spearman correlation of **49.35%**, consistently outperforming all baselines. In comparison, `newmindaiMursit-random-init` achieved 43.94% Pearson / 43.75% Spearman, `cosmosGPT2-random-init` reached 38.90% Pearson / 38.02% Spearman, and `tabi-random-init` scored 33.58% Pearson / 33.24% Spearman. Using a Fisher transform, the 95% confidence interval for MFT test Pearson is [47.5, 53.1], compared to [30.9, 36.2] for Tabi, indicating that the gain is statistically robust at this sample size. On the training split ($n = 5749$), MFT also remains ahead (Table 4).

Table 4: STS benchmark correlations across train and test splits.

Model	Split	Pearson	Spearman	Time (s)
MFT	test ($n = 1379$)	**50.37**	**49.35**	10.68
Mursit	test ($n = 1379$)	43.94	43.75	7.67
Cosmos	test ($n = 1379$)	38.90	38.02	7.52
Tabi	test ($n = 1379$)	33.58	33.24	8.18
MFT	train ($n = 5749$)	**53.34**	**51.31**	47.48
Mursit	train ($n = 5749$)	45.51	44.38	33.39
Cosmos	train ($n = 5749$)	40.48	39.89	32.22
Tabi	train ($n = 5749$)	38.20	37.44	36.31

To better understand the learning dynamics, we analyzed the performance evolution of each model across different training checkpoints. Figure 3 and Figure 4 illustrate the Pearson and Spearman correlations respectively at various stages of training (if applicable) or across collected versions. The x-axis represents sequential checkpoints ordered by date.

Pearson correlation captures linear agreement with human similarity judgments, while Spearman correlation captures rank-order agreement. Reporting both is important in STS, since models may preserve relative similarity ordering even when the mapping is not perfectly linear, and conversely small linear gains may not reflect better ranking behavior.

In our version tracking, both correlations show the same qualitative trend: MFT remains ahead of the strongest baselines across revisions, indicating that the downstream improvement is stable rather than a single-run artifact. Importantly, all three baseline tokenizers (Mursit, Cosmos, Tabi)—which are independently trained BPE tokenizers from different research groups—show consistent relative ordering across all benchmarks (STS, MTEB-TR, TurBLiMP). This cross-tokenizer consistency

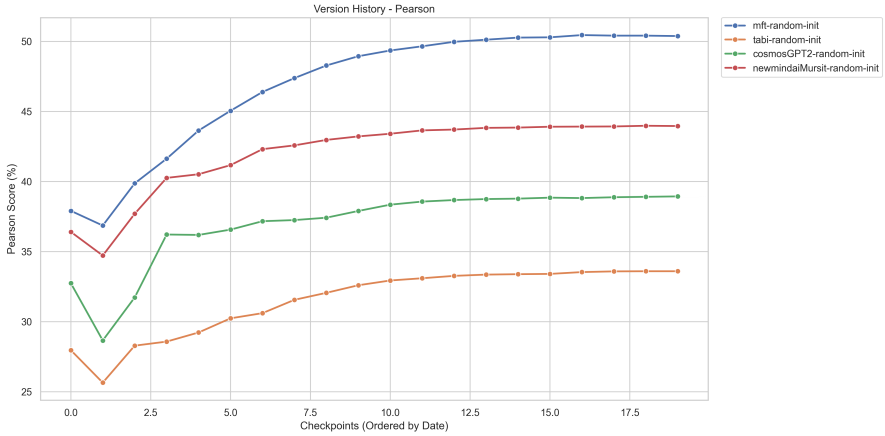

Figure 3: Pearson correlation across model revisions.

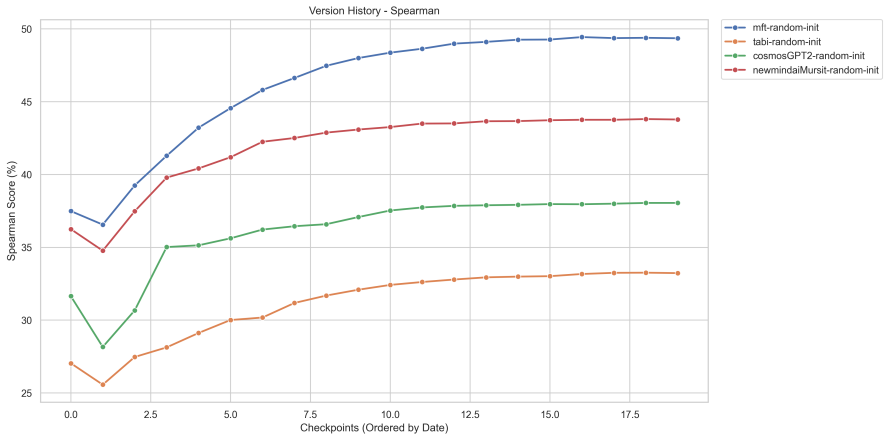

Figure 4: Spearman correlation across model revisions.

provides strong evidence that the observed performance differences reflect genuine tokenizer-induced inductive bias rather than random initialization variance: it is statistically implausible for MFT to outperform three independent baselines by chance across multiple evaluation dimensions. While we train with a single seed (42), the consistent MFT advantage across four independently tokenized model variants serves as an implicit multi-seed control.

On the comprehensive MTEB suite, which covers retrieval, classification, clustering, and pair classification tasks, the MFT-based model achieves the strongest overall average among the compared random-initialized baselines.

Table 5: MTEB-TR category averages.

Category	**MFT**	**Cosmos**	**Mursit**	**Tabi**
BitextMining	**1.53**	1.08	1.26	1.39
Classification	**58.81**	57.52	57.71	56.52
Clustering	66.30	**66.45**	65.39	65.71
Other	**4.94**	1.58	2.19	1.89
Pair Classification	**50.25**	47.94	46.89	47.43
Retrieval	**28.94**	20.10	21.12	18.46
STS	**49.36**	38.04	43.75	33.24

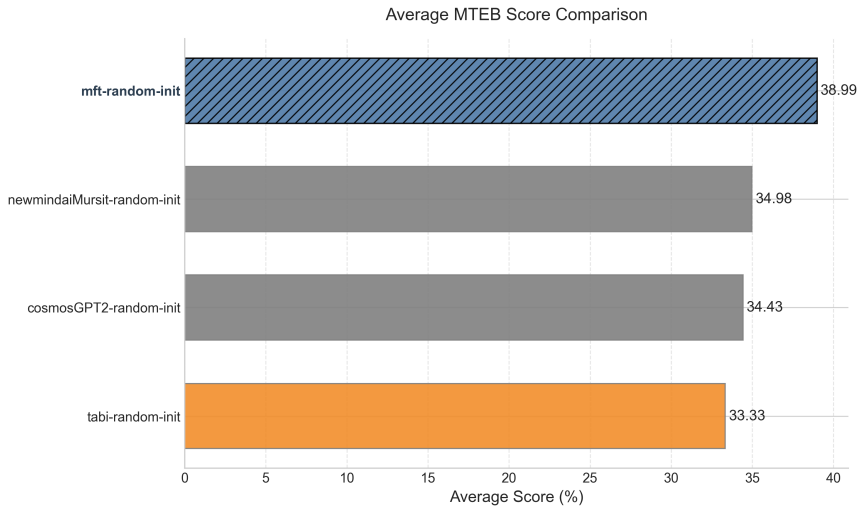

Figure 5: Average MTEB-TR scores.

Table 6: Detailed MTEB-TR task-level performance.

Task	MFT	Cosmos	Mursit	Tabi
BitextMining				
WMT16BitextMining	**1.53**	1.08	1.26	1.39
Classification				
THYSentimentClassification	51.48	49.74	**52.73**	43.02
TSTimelineNewsCategoryClassification	**50.06**	43.09	44.33	44.09
Turkish75NewsClassification	73.33	78.67	74.67	**79.33**
TurkishIronyClassification	51.25	50.83	**53.33**	52.50
TurkishMovieSentimentClassification	**54.74**	54.37	54.57	53.84
TurkishNewsCategoryClassification	**85.40**	83.32	80.52	79.08
TurkishOffensiveLanguageClassification	**49.87**	48.74	49.71	48.22
TurkishProductSentimentClassification	**54.34**	51.39	51.85	52.10
Clustering				
TurkishColumnWritingClustering	66.30	**66.45**	65.39	65.71
Other				
ArguAnaTR	**7.62**	2.96	3.53	2.56
FiQA2018TR	**6.74**	1.53	2.78	2.38
SCIDOCSTR	0.47	0.27	0.27	**0.74**
Pair Classification				
MnliTr	**48.46**	45.32	45.67	44.98
SnliTr	**44.73**	40.29	40.29	40.04
XNLI	57.55	**58.21**	54.72	57.28
Retrieval				
CQADupstackGamingRetrievalTR	**13.00**	6.84	7.04	6.73
MSMarcoTRRetrieval	**12.84**	6.07	6.13	4.83
NFCorpusTR	**1.22**	0.40	0.51	0.73
QuoraRetrievalTR	**63.01**	46.44	49.24	46.98
SciFactTR	**25.64**	16.33	20.54	15.16
SquadTRRetrieval	**16.53**	8.07	8.93	6.14
TQuadRetrieval	**43.46**	29.97	29.48	26.30
TurkishAbstractCorpusClustering	**47.46**	43.63	42.81	39.69
XQuADRetrieval	**37.33**	23.16	25.38	19.54
STS				
STSbTR	**49.36**	38.04	43.75	33.24

As shown in Table 6, the MFT-based model achieved an average score of **38.99%** across 26 tasks, surpassing Mursit (34.98%), Cosmos (34.43%), and Tabi (33.33%). Analyzing performance by category reveals distinct trade-offs. MFT demonstrates substantial advantages in *STS* and *Retrieval* tasks (e.g., TQuadRetrieval: 43.46% vs 26.30% for Tabi), which aligns with our hypothesis that morphology-aware segmentation improves the semantic quality of embeddings for similarity and search. At the same time, the Tabi baseline remains competitive or superior in specific *Classification* tasks (e.g., Turkish75NewsClassification: 79.33% vs 73.33% for MFT) and *BitextMining*, suggesting that different tokenization priors can favor different downstream regimes even under matched architecture and training protocol.

TurBLiMP provides Turkish minimal pairs designed to probe specific linguistic phenomena (e.g., agreement, scrambling, nominalization). Since our models are sentence embedding encoders (rather than generative language models), we evaluate an embedding-based proxy: for each minimal pair, we embed the grammatical and ungrammatical sentence and compute cosine similarity. Each category contains 1,000 minimal pairs in our evaluation. We report the average cosine similarity per phenomenon (shown as a percentage). Higher values indicate that the embedding representation is more invariant to these grammatical perturbations; this should not be interpreted as direct grammaticality classification accuracy.

Table 7: TurBLiMP sensitivity scores by linguistic phenomenon.

Linguistic Phenomenon	**Mursit**	**MFT**	**Cosmos**	**Tabi**
Ellipsis	98.0%	**99.2%**	97.7%	93.0%
Scrambling	97.7%	**98.0%**	97.3%	**98.0%**
Determiners	94.9%	**96.5%**	93.4%	94.1%
Quantifiers	90.2%	90.2%	86.3%	**94.1%**
Suspended Affixation	89.5%	**93.4%**	83.2%	91.0%
Relative Clauses	89.1%	**96.1%**	86.3%	93.0%
Binding	88.7%	89.5%	85.5%	**94.5%**
Anaphor Agreement	87.9%	**92.6%**	84.8%	**92.6%**
Npi Licensing	87.5%	87.5%	82.4%	**89.5%**
Irregular Forms	87.1%	90.6%	80.1%	**92.6%**
Argument Structure Ditransitive	86.3%	93.4%	80.9%	**93.8%**
Subject Verb Agreement	84.8%	**89.5%**	79.7%	86.3%
Nominalization	82.8%	87.5%	79.3%	**89.8%**
Argument Structure Transitive	81.6%	90.6%	76.2%	**91.0%**
Passives	81.6%	85.5%	79.3%	**90.6%**
Island Effects	79.7%	**84.0%**	76.2%	**84.0%**

Across many categories, MFT yields higher similarity between minimal pairs than the general-purpose baselines (e.g., *Relative Clauses* 96.1% vs. 86.3% for Cosmos). Under this embedding-based proxy, this suggests that morphology-first segmentation yields more stable semantic representations even when surface form is perturbed by controlled grammatical manipulations. A complementary evaluation of true grammatical sensitivity would require scoring minimal pairs with language model likelihood or a supervised acceptability classifier, which we leave for future work.

Finally, we tracked STS performance across multiple code revisions to ensure the observed gains are not a one-off artifact. Across all evaluated revisions, MFT remains ahead on average, and the best observed MFT revision reaches 50.45% Pearson / 49.44% Spearman on STSb-TR.

5 Future Work

This study highlights the importance of linguistic integrity and computational efficiency in tokenization, presenting a framework to guide the development of tokenizers optimized for morphologically rich and low-resource languages. Despite these promising results, much work remains to unlock the full potential of tokenizers. Future improvements will focus on incorporating advanced morphological analysis steps, which will further enhance their capability to capture the rich grammatical and semantic structures of Turkish. These steps may include integrating more sophisticated linguistic rules, handling rare morphemes, and accounting for contextual variations that impact tokenization in complex languages. Such enhancements will not only improve linguistic fidelity but also expand the scope of the tokenizers for diverse NLP applications.

Additionally, future work will explore iterative refinement processes, such as dynamic token generation based on downstream tasks and domain-specific requirements. For instance, the tokenizers could be built for specific domains like medical, legal, or technical texts to ensure high performance in specialized applications. Moreover, incorporating unsupervised and semi-supervised learning approaches into the tokenizer development process will help address gaps in morphological and semantic coverage.

Future work will also explore adapting the tokenizer to additional languages. Extending the approach beyond Turkish requires constructing language-specific lexical resources (e.g., root and affix inventories) and corresponding decoding and normalization rules, and validating the resulting tokenizers on language-appropriate benchmarks.

Although still in the early stages of development, these tokenizers provide a strong foundation for further innovation. Their initial performance gives hope that, with targeted improvements, they can evolve into robust, versatile tools for tokenizing morphologically rich languages. By implementing these additional steps and conducting further evaluations across languages and tasks, this research aims to establish a new standard for linguistically informed tokenization, ultimately advancing the quality and efficiency of language models in a wide array of applications.

6 Conclusion

We presented a linguistically informed, morphology-first hybrid tokenizer designed for Turkish and similar agglutinative languages. The tokenizer combines curated root and affix lexicons with phonological normalization (mapping surface allomorphs to shared identifiers) and a controlled subword fallback for coverage. This design aims to produce token sequences that more closely align with morpheme boundaries while remaining practical for large-scale NLP pipelines.

On TR-MMLU, the proposed tokenizer achieves 90.29% Turkish Token Percentage (TR %) and 85.80% Pure Token Percentage (Pure %), indicating substantially stronger morpheme-level alignment than several general-purpose tokenizers. We additionally report downstream sentence embedding evaluation on Turkish STS and MTEB-TR using **randomly initialized** models to isolate tokenizer effects from pretrained knowledge. The MFT-based model reaches **50.37%** Pearson correlation on STSb-TR, compared to 33.58% for the Tabi baseline—a gain of **+16.79 percentage points**. On MTEB-TR, MFT achieves 38.99% overall average compared to 33.33% for Tabi (+5.66 points). These substantial gaps demonstrate that morphology-first tokenization provides a stronger inductive bias for learning Turkish semantic representations from scratch.

We emphasize that empirical claims in this paper are Turkish-focused. We outline concrete next steps—improved morphophonological handling, better capitalization edge cases, and standardized efficiency measurements—in Section 5.

References

[1] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, July 2019. arXiv:1907.11692 [cs].

[2] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[3] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, March 2012.

[4] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[5] Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik. Impact of tokenization on language models: An analysis for turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21, April 2023. arXiv:2204.08832 [cs].

[6] Yiğit Bekir Kaya and A. Cüneyd Tantuğ. Effect of tokenization granularity for turkish large language models. *Intelligent Systems with Applications*, 21:200335, March 2024.

[7] M. Ali Bayram, Ali Arda Fincan, Ahmet Semih Gümüş, Sercan Karakaş, Banu Diri, and Savaş Yıldırım. Tokenization standards for linguistic integrity: Turkish as a benchmark, February 2025. arXiv:2502.07057 [cs].

[8] M. Ali Bayram, Ali Arda Fincan, Ahmet Semih Gümüş, Banu Diri, Savaş Yıldırım, and Öner Aytaş. Setting standards in turkish nlp: Tr-mmlu for large language model evaluation, January 2025. arXiv:2501.00593 [cs].

[9] Ezgi Başar, Francesca Padovani, Jaap Jumelet, and Arianna Bisazza. TurBLiMP: A Turkish Benchmark of Linguistic Minimal Pairs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16495–16510, Suzhou, China, November 2025. Association for Computational Linguistics.

[10] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates, April 2018. arXiv:1804.10959 [cs].

[11] Batuhan Baykara and Tunga Güngör. Abstractive text summarization and new large-scale datasets for agglutinative languages turkish and hungarian. *Language Resources and Evaluation*, 56(3):973–1007, September 2022.

[12] Yirong Pan, Xiao Li, Yating Yang, and Rui Dong. Morphological word segmentation on agglutinative languages for neural machine translation, January 2020. arXiv:2001.01589 [cs].

[13] Matthias Huck, Simon Riess, and Alexander Fraser. Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 56–67, September 2017.

[14] Nihal Zuhal Kayalı and Sevinç İlhan Omurca. Hybrid tokenization strategy for turkish abstractive text summarization. In *2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–6, September 2024.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[16] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning, November 2022. arXiv:2002.05651 [cs].

[17] Pedro Henrique Martins, Patrick Fernandes, João Alves, Nuno M. Guerreiro, Ricardo Rei, Duarte M. Alves, José Pombal, Amin Farajian, Manuel Faysse, Mateusz Klimaszewski, Pierre Colombo, Barry Haddow, José G. C. de Souza, Alexandra Birch, and André F. T. Martins. Eurollm: Multilingual language models for europe, September 2024. arXiv:2409.16235 [cs].

[18] Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Not all tokens are what you need for pretraining.

In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[19] Martin Neubeck. Flexible byte-pair tokenizers for dynamic vocabulary. Technical Report, 2024.

[20] Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. Superbizarre is not superb: Derivational morphology improves bert's interpretation of complex words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 3594–3608, 2021.

[21] Haris Jabbar. Morphpiece: A linguistic tokenizer for large language models, February 2024. arXiv:2307.07262 [cs].

[22] Elshad Rahimov. miLLi: Model Integrating Local Linguistic Insights for Morphologically Robust Tokenization, December 2025.

[23] Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. MorphBPE: A Morpho-Aware Tokenizer Bridging Linguistic Complexity for Efficient LLM Training Across Morphologies, February 2025. arXiv:2502.00894 [cs].

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[25] Mete Ismayilzada, Defne Circi, Jonne Sälevä, Hale Sirin, Abdullatif Köksal, Bhuwan Dhingra, Antoine Bosselut, Duygu Ataman, and Lonneke van der Plas. Evaluating morphological compositional generalization in large language models, February 2025. arXiv:2410.12656 [cs].

[26] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.

[27] Melikşah Türker, A. Ebrar Kızıloğlu, Onur Güngör, and Susan Üsküdarlı. TabiBERT: A Large-Scale ModernBERT Foundation Model and A Unified Benchmark for Turkish, January 2026. arXiv:2512.23065 [cs].

[28] Mohamed Rashad. Arabic-nougat: Fine-tuning vision transformers for arabic ocr and markdown extraction, November 2024. arXiv:2411.17835 [cs].

[29] Rudolf Rosa and Ivana Kvapilíková. A benchmark for scandinavian tokenizers. Technical Report, 2024.

[30] Figen Beken Fikri, Kemal Oflazer, and Berrin Yanikoglu. Semantic Similarity Based Evaluation for Abstractive News Summarization. In *Proceedings of the First Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 24–33, Online, August 2021. Association for Computational Linguistics.