NOHA: A Diacritic-Aware Rule-Based Morphophonological Tokenizer for Arabic

Anonymous authors
Paper under double-blind review

Abstract

Arabic is a morphologically rich language whose complex phonological and morphological structure presents fundamental challenges for statistical tokenization methods used in large language model (LLM) pre-training. Although statistical methods such as Byte-Pair Encoding (BPE) are widely used, they are not optimally suited to morphologically rich languages such as Arabic, where frequency-based subword discovery ignores phonological and morphological boundaries and produces inconsistent segmentation of diacritized text. This paper presents NOHA, a fully deterministic diacritic-aware Arabic tokenizer that replaces statistical subword discovery with a rule-based chunking algorithm based on Arabic morphological and phonological structure. NOHA applies clitic preservation and diacritic-driven syllabic chunking to produce subword units that reflect the natural sound and grammatical structure of Arabic, without any statistical training. Despite producing approximately 1.49× more tokens per sequence than BPE (and thus observing fewer unique training documents under identical compute and hyperparameter conditions), a GPT-2-scale model trained with NOHA achieves higher average accuracy across 38 matched-compute evaluation checkpoints on all three targeted Arabic linguistic benchmarks: +2.42% on GAT Reading Comprehension, +2.65% on the three Arabic language subsets of ArabicMMLU, and +2.54% on the Teacher-Authored Arabic Grammar benchmark (TAAG, n=171), a 171-question evaluation set developed and validated by a certified Arabic language teacher for this study. On general knowledge benchmarks, BPE leads by 0.26% on a weighted average across 18 ArabicMMLU subsets, indicating that NOHA preserves general knowledge capacity while improving linguistic understanding. BPE achieves lower bits-per-byte at all 38 matched-compute evaluation checkpoints, with a mean compression advantage of +2.87%. Both models perform near the 25% random baseline on the Nahw-MCQ Arabic grammar benchmark, the remaining GATLc subtasks (Verbal Analogy, Sentence Completion, Contextual Error, and Semantic Association), and most mathematical reasoning subtasks, confirming that model scale rather than tokenizer design is the limiting factor for these tasks. For mathematical reasoning subtasks, limited exposure to mathematical content in the training corpus is an additional contributing factor. These findings show that linguistically grounded tokenization yields measurable improvements on Arabic linguistic tasks at an identical computational budget, and that compression efficiency and linguistic understanding are dissociable properties of Arabic tokenizer design.

1 Introduction

Tokenization is the foundational step that converts raw text into discrete units for language model training, shaping what the model learns and what linguistic structure it retains. Tokenizer quality is a significant factor in downstream performance (Rust et al., 2021). Subword tokenization methods, most notably Byte-Pair Encoding (BPE) (Sennrich et al., 2016), have become the dominant approach in modern large language models (LLMs), adopted in systems such as GPT (OpenAI, 2023), LLaMA (Touvron et al., 2023), DeepSeek (Bi et al., 2024), Qwen (Bai et al., 2023), and Falcon (Almazrouei et al., 2023). Attempts to eliminate tokenization through byte-level architectures such as MegaByte (YU et al., 2023), MambaByte (Wang et al., 2024),

SpaceByte (Slagle, 2024), and the Byte Latent Transformer (Pagnoni et al., 2025) require architectural modifications or entirely new model designs to manage the computational cost of operating directly on raw byte sequences. Consequently, BPE remains the practical standard for LLM pre-training.

Despite its dominance, growing evidence suggests that frequency-based tokenization is not optimally suited to morphologically rich languages. BPE's frequency-driven merge procedure overlooks word segment boundaries, producing insufficient segmentation for languages like Arabic (Asgari et al., 2025). Furthermore, compression efficiency alone is an insufficient predictor of downstream performance (Lotz et al., 2025; Schmidt et al., 2024). Tokenizer choice is particularly consequential for non-English languages (Lotz et al., 2025).

Arabic is among the most challenging test cases for these limitations. Its morphology combines concatenative and templatic mechanisms: a single consonantal root combines with different vowel patterns to produce words with distinct meanings. The root *k-t-b* (ك-ت-ب) alone generates writing (*kataba*, كَتَبَ), writer (*kātib*, كَاتِب), written (*maktūb*, مَكْتُوب), book (*kitāb*, كِتَاب), office (*maktab*, مَكْتَب), and library (*maktaba*, مَكْتَبَة) (Habash, 2010). This templatic productivity results in a combinatorial richness that frequency-based tokenization cannot adequately capture: Arabic exhibits 2.5 times the vocabulary growth rate of English, ten times its out-of-vocabulary rate on a 64K vocabulary, and a theoretically possible POS tag space exceeding 330,000 compared to 46 tags for English (Habash, 2010). BPE's frequency-driven merge procedure is fundamentally misaligned with this non-linear morphological structure, as it struggles specifically with root-and-pattern systems where meaning emerges through root-and-pattern combinations rather than sequential affixation (Asgari et al., 2025).

Despite ongoing efforts to build Arabic-centric LLMs, these models continue to exhibit substantial gaps in genuine linguistic understanding: the best Arabic-centric models achieve only 42% accuracy on Arabic grammar benchmarks (Mubarak et al., 2026), and error rates reach 36.5% on diacritics-reliant tasks across evaluated models (Al-Olimat & Alshareef, 2026). Arabic diacritical marks, known as *harakat*, are absent from the vast majority of naturally occurring Arabic text: they encode short vowels, consonant doubling (shadda), the absence of a vowel (sukun), and morpho-syntactic case distinctions (Habash, 2010), yet most available Arabic text omits them, leaving tokenizers trained on such data with no vocabulary entries for these signals. Figure 1 demonstrates this concretely: every evaluated tokenizer, across vocabulary sizes from 64K to 261K, general-purpose and Arabic-specific alike, either fragments diacritized input into near-character-level pieces or produces unknown tokens.

Alyafeai et al. (2021) systematically compare six tokenization schemes for Arabic across classification tasks, finding that no single approach generalizes reliably and that performance depends heavily on task type and dataset size. Alrefaie et al. (2024) show that augmenting BPE with Farasa morphological pre-segmentation outperforms plain BPE on most Arabic NLP fine-tuning tasks, confirming that morphological awareness improves downstream performance. Yet this approach retains BPE as its statistical core, operates on undiacritized text, and primarily evaluates tokenizer impact at the fine-tuning stage, leaving the question of pre-training tokenizer quality unexplored. Asgari et al. (2025) constrain BPE merges to respect word segment boundaries and demonstrate improved training convergence at the 300M--1B scale, with deployment in the Fanar Arabic LLM. While meaningful, this approach remains fundamentally statistical and lacks phonological awareness beyond word-segment-level segmentation. Its evaluation is also confined to intrinsic morphological metrics and training convergence, without any empirical validation on downstream NLP tasks. None of these approaches leverage diacritical marks as structural tokenization signals, nor ground their design in Arabic language acquisition research, leaving unexplored whether a fully deterministic, diacritic-driven tokenizer can enhance genuine Arabic linguistic understanding in pre-trained LLMs.

Motivated by this gap, this paper presents NOHA, a deterministic Arabic tokenizer for fully diacritized text, designed around two levels of Arabic linguistic structure. At the morphological level, a clitic segmentation step preserves prefix and suffix boundaries as atomic tokens, consistent with Arabic acquisition research showing that prefixes and suffixes constitute distinct processing stages that children master before diacritics are fully acquired (Issa, 2023). At the phonological level, NOHA replaces statistical subword discovery with diacritic-driven rules. Arabic children first encounter the language through fully diacritized text in which every sound boundary is explicitly marked (Habash, 2010; Issa, 2023). NOHA builds on this observation:

UNDIACRITIZED INPUT	كرّم الموظف المدير لأنّه يجلّه فاستعد الجميع وانطلقوا للإحتفال بالتكريم.
DIACRITIZED INPUT	كَرَّمَ الْمُوَظَّفُ الْمُدِيرَ لِأَنَّهُ يُجِلُّهُ فَاسْتَعَدَّ الْجَمِيعُ وَانْطَلَقُوا لِلْإِحْتِفَالِ بِالتَّكْرِيمِ.

Model	Training Dataset Languages	Tokens (U→D)	Ratio
TYPE I Categorical failure — diacritics produce unknown or replacement tokens			
AraGPT2-large	Arabic	36 → 146	4.1×
Jais-2-70B	Arabic–English	34 → 121	3.6×
ArabianGPT-03B	Arabic	26 → 91	3.5×
TYPE II Severe fragmentation — diacritics preserved only by splitting into extra tokens			
Falcon-H1-34B	Multilingual	26 → 63	2.4×
Fanar-2-27B	Arabic–English	29 → 65	2.2×
AceGPT-v2-8B	Arabic	33 → 73	2.2×
ALLaM-7B	Arabic–English	40 → 68	1.7×

OBSERVED FAILURE MODES

Replacement errors (unknown tokens)
- **AraGPT2-large:** 112 of 146 tokens are U+FFFD.
- **Jais-2-70B:** 60 of 121 tokens are U+FFFD.
- **ArabianGPT-03B:** 47 of 91 are *<unk>*; claims diacritic-aware.

Fragmentation errors (diacritics split into extra tokens)
- **Falcon-H1-34B:** Diacritics isolated as standalone tokens.
- **Fanar-2-27B:** Diacritics split from adjacent letters.
- **AceGPT-v2-8B:** Fragmentation persists despite 128K vocabulary.
- **ALLaM-7B:** 9 *<unk>* on word boundaries; diacritics split.

Figure 1: Tokenizer fragmentation on diacritized versus undiacritized Arabic across seven systems: AraGPT2-large (Antoun et al., 2021), Jais-2-70B (Anwar et al., 2025), ArabianGPT-03B (Koubaa et al., 2024), Falcon-H1-34B (Zuo et al., 2025), Fanar-2-27B (TEAM et al., 2026), AceGPT-v2-8B (Liang et al., 2024; Zhu et al., 2025), and ALLaM-7B (Bari et al., 2025). Token counts are measured on the example sentence above. U = undiacritized; D = diacritized. U+FFFD = Unicode replacement character (token produced when a codepoint has no vocabulary entry).

diacritical marks serve as boundary signals, with syllable boundaries triggered by shadda (Habash, 2010) and sukun (Habash, 2010), and long vowels producing units that correspond to the primary syllable types of Arabic: CV (*ma*, مَ), CVV (*mā*, مَا), and CVC (*man*, مَنْ) (Watson, 2002), where closed syllables are marked by sukun or shadda, and long vowels by the elongation letters (ا و ي). CVVC words such as *bāb* (بَابْ) are split at the long vowel boundary, producing a CVV chunk followed by a closed consonant, rather than a single CVVC unit. A morphological segmentation tool is used solely to detect prefix and suffix boundaries such as *wa-* (وَ), *al-* (ال), *lel-* (لِلْ), and *bi-* (بِ), after which an independent rule system applies diacritic-driven chunking to produce the final token stream. NOHA is among the first Arabic tokenizers to replace statistical tokenization entirely with diacritic-driven rules grounded in Arabic phonological and morphological structure.

To evaluate NOHA, a 190M GPT-2-scale (Radford et al., 2019) model is trained on fully diacritized Arabic text from FineWeb-Edu-Ar (Alrashed et al., 2024) and compared against an architecturally identical BPE-trained model under a strictly matched computational budget (same architecture, vocabulary size of 68,044, batch size, context length, and training steps). Bits-per-byte (BPB) (Slagle, 2024; Pagnoni et al., 2025) is used as the primary intrinsic metric, as it normalizes prediction quality by byte count rather than token count, making it independent of tokenizer fertility and enabling fair cross-tokenizer comparison. BPE achieves lower BPB at all 38 matched-compute validation checkpoints, with a mean compression advantage of 2.87%. Despite this, NOHA achieves higher average accuracy across 38 matched-compute evaluation checkpoints on all three targeted Arabic linguistic benchmarks: +2.42% on GAT Reading Comprehension (AlBallaa et al., 2025), +2.65% on the three Arabic language subsets of ArabicMMLU (Koto et al., 2024), and +2.54% on the Teacher-Authored Arabic Grammar benchmark (TAAG), a 171-question evaluation set developed for this study and validated by a certified Arabic language teacher. On Nahw-MCQ (Mubarak et al., 2026), a comprehensive Arabic grammar benchmark across three difficulty levels, both models score near the 25% random baseline. The remaining GATLc subtasks (Verbal Analogy, Sentence Completion, Contextual Error, and Semantic Association) and most mathematical reasoning subtasks show the same pattern, reflecting the

limits of model scale rather than tokenizer design. For mathematical reasoning subtasks, limited exposure to mathematical content in the training corpus is an additional contributing factor. On general knowledge benchmarks (18 ArabicMMLU general subsets), BPE leads by a small margin of 0.26% on a weighted average, indicating that NOHA preserves general knowledge capacity while improving linguistic understanding. The pattern is confirmed at epoch-aligned checkpoints, ruling out data-coverage as an explanation for NOHA's advantage. These results demonstrate a three-way dissociation: BPE wins compression, NOHA wins Arabic linguistic understanding, and general knowledge capacity is largely preserved.

These findings establish that, at the 190M parameter scale under an identical computational budget, linguistically grounded tokenization achieves higher accuracy on Arabic linguistic tasks at no additional training cost, and that compression efficiency and linguistic understanding are dissociable properties of Arabic tokenizer design. NOHA's linguistic advantage is present from the earliest checkpoints and persists throughout training, while BPE's compression advantage widens, indicating these are stable properties of the two tokenization strategies rather than transient effects. These results indicate that tokenizer design is an influential factor in Arabic linguistic performance under these controlled conditions.

The main contributions of this paper are:

(i) NOHA, among the first Arabic tokenizers with fully deterministic chunking, combining acquisition-motivated morphological clitic preservation with diacritic-driven syllabic chunking, requiring no statistical training.

(ii) Empirical evidence that NOHA achieves higher average accuracy than BPE across 38 matched-compute evaluation checkpoints on all three targeted Arabic linguistic benchmarks despite BPE's persistent compression advantage, while general knowledge capacity is largely preserved, demonstrating a three-way dissociation between compression efficiency, linguistic understanding, and general knowledge.

(iii) TAAG, a 171-question Teacher-Authored Arabic Grammar benchmark developed for this study and validated by a certified Arabic language teacher.

(iv) A controlled pre-training comparison at the 190M GPT-2 scale across 38 matched-compute evaluation checkpoints under strictly matched architecture, vocabulary, and compute budget.

The remainder of this paper is organized as follows. Section 2 formalizes the NOHA pipeline. Section 3 describes the training corpus and evaluation benchmarks. Section 4 describes the experimental setup. Section 5 presents the results. Section 6 discusses the findings, and Section 7 concludes.

2 Methodology

2.1 Overview

NOHA is a deterministic pipeline that converts fully diacritized Arabic text into integer token IDs for language model training through three sequential phases: segmentation, diacritic-driven chunking, and vocabulary construction and tokenization. The design preserves two levels of Arabic linguistic structure that frequency-based subword methods often fail to capture: morphological structure, through explicit identification of clitic boundaries before chunking, and phonological structure, through deterministic diacritic-driven segmentation of each word segment into syllabically motivated units.

At the morphological level, Arabic clitics such as *wa-* (وَ), *bi-* (بِ), and *al-* (ال) attach directly to the following word. Their boundaries must be identified before chunking so that downstream language models can access each clitic and the word it attaches to as distinct units, rather than processing the combined surface form as an indivisible sequence. Issa (2023) shows that Arabic children's spelling follows a hierarchical process in which roots or stems are first accessed and mapped to their word pattern, followed by prefixes, then suffixes, and finally diacritics. The identification of clitic boundaries before diacritic-driven chunking in NOHA is consistent with this hierarchy.

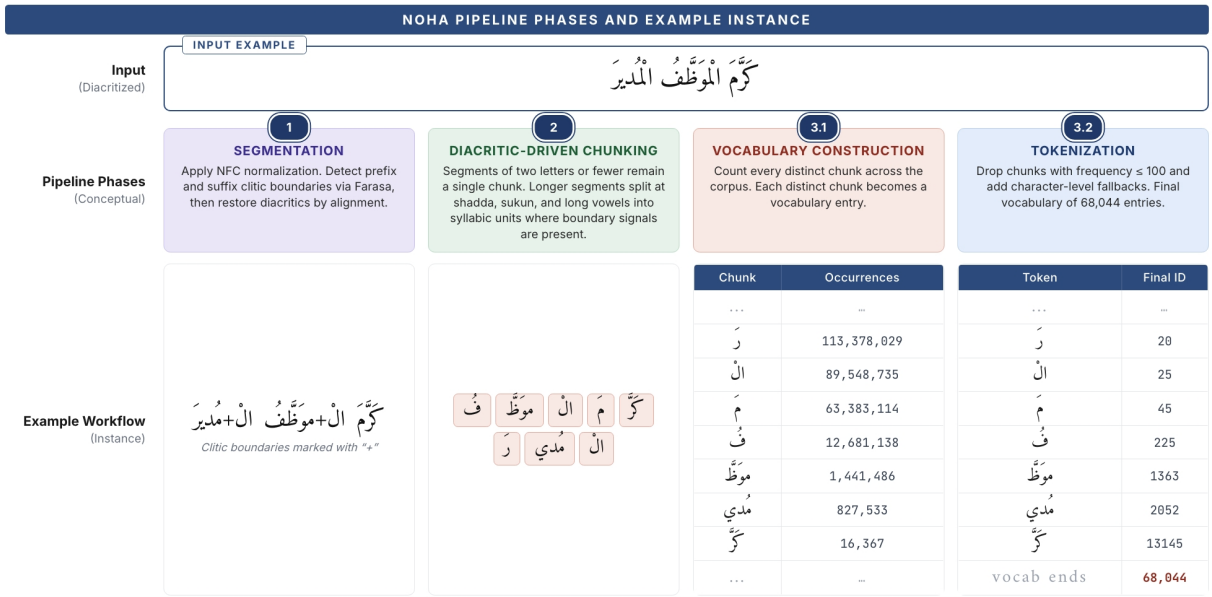

Figure 2: The three-phase NOHA tokenization pipeline: segmentation, diacritic-driven chunking, and vocabulary construction and tokenization.

At the phonological level, diacritical marks encode short vowels, the shadda (U+0651), and the sukun (U+0652) (Habash, 2010). While these signals are absent from the bulk of naturally occurring Arabic text, including newspapers and books (Habash, 2010; Issa, 2023), Arabic children first encounter the language through fully diacritized text in which sound boundaries are explicitly marked. NOHA builds on this observation, treating diacritical marks as the primary segmentation signal rather than discarding them.

The first phase, **Segmentation** (§2.2), produces fully diacritized word segments with explicit prefix and suffix boundaries. This step is necessary because Phase 2 operates within word segments. Without prior boundary detection, chunking rules could produce units that cross clitic boundaries. The second phase, **Diacritic-driven chunking** (§2.3), segments each diacritized word segment using the shadda, sukun, and long vowels as diacritic triggers that determine chunk boundaries. The third phase, **Vocabulary construction and tokenization** (§2.4), aggregates chunk frequencies across the corpus, assigns integer IDs, and maps the chunked corpus to token sequences.

Figure 2 summarizes the pipeline. The subsections below trace the following example sentence through the complete process:

كَرَّمَ الْمُوَظَّفُ الْمُدِيرَ لِأَنَّهُ يُجِلُّهُ فَاسْتَعَدَّ الْجَمِيعُ وَانْطَلَقُوا لِلْإِحْتِفَالِ بِالتَّكْرِيمِ

(*The employee honored the director because he respects him, so everyone prepared and set off for the celebration of the honoring.*)

2.2 Phase 1: Segmentation

Step 1: Unicode normalization Unicode Canonical Composition (NFC) normalization is applied as the first processing step. A total of 175 token pairs were identified in the constructed vocabulary in which two distinct entries share the same surface form while differing at the codepoint level. All 175 collision pairs are attributable to the shadda (U+0651) combining mark: Unicode does not canonically fix the relative order of shadda and a co-occurring short vowel diacritic in the codepoint sequence, causing the same surface form to appear under two distinct codepoint representations. Left unnormalized, the gradient updates for a single

Table 1: Representative NFC collision pairs in the raw vocabulary.

Surface	Form	Colliding bytes (hex)	ID
مُتَّ	Canonical	`d991 d98e`	385
	Non-canonical	`d98e d991`	462,774
إِنَّ	Canonical	`d991 d98e`	541
	Non-canonical	`d98e d991`	312,154
ثُمَّ	Canonical	`d991 d98e`	566
	Non-canonical	`d98e d991`	373,186
مَّ	Canonical	`d991 d98e`	495
	Non-canonical	`d98e d991`	279,094

surface form are divided across two separate embedding rows, consuming model capacity without benefit. NFC normalization eliminates this ambiguity by enforcing a single canonical codepoint ordering for each surface form. Representative collision pairs from the raw vocabulary are shown in Table 1.

Step 2: Clitic boundary detection Following normalization, the text is processed by the Farasa toolkit for Arabic morphological analysis (Darwish & Mubarak, 2016). Farasa was selected for its clitic segmentation capability, which Phase 1 requires. The same toolkit is used to diacritize the training corpus during preprocessing, as described in Section 3. This step identifies clitic boundaries, marking attached prefixes and suffixes as discrete units before chunking. Farasa returns an undiacritized, segmented output in which each detected boundary is marked with a `+` delimiter.

Step 3: Word-level alignment and diacritic restoration Because Farasa returns undiacritized segmented text, the segmented output must be aligned back to the original diacritized sentence. Each segmented word is matched to its source word by comparing their undiacritized base-letter sequences.

A word-level mismatch arises when Farasa resolves an orthographic contraction, causing the segmented form to contain more base letters than the source word. In the running example, the word (لِلْإِحْتِفالِ) is the contracted surface form of the preposition *li-* (لِ) and the definite article *al-* (ال). Farasa internally expands the contracted *lam-lam* (لل) back to a *lam-alef-lam* (لال) sequence, producing a segmented form with more base letters than the source word. Direct character-by-character alignment is therefore not possible for this word. Such mismatches trigger a resynchronization strategy: the algorithm scans forward to the next word for which alignment succeeds (بِالتَّكْرِيمِ in Table 2), using it as a synchronization anchor to map the intervening mismatched words sequentially.

After establishing the word-level mapping, diacritics are transferred character by character from the source word to the segmented form. The `+` boundary markers inserted by Farasa in Step 2 are temporarily removed from the segment before character-by-character alignment runs, to prevent them from being counted as Arabic base letters. Once diacritic transfer is complete, the boundary markers are restored to their original positions in the segmented form. In cases of expansion, such as the *lam-alef-lam* sequence, the inserted *alef* (ا) is assigned a default *fatha* (َـ) to preserve the standard vocalization of the definite article. A second NFC pass is then applied to ensure codepoint consistency before chunking. Table 3 shows the complete alignment and diacritic restoration for the running example.

2.3 Phase 2: Diacritic-Driven Chunking

This phase implements the phonological level of NOHA's design. In Arabic phonology, long vowels are bimoraic, shadda contributes to syllable weight, and sukun defines syllable closure (Watson, 2002). NOHA uses these orthographic marks as diacritic triggers that determine chunk boundaries: shadda, sukun, and long vowels signal the edges at which the tokenizer splits a word segment into subword units.

Table 2: Word-level alignment between the Farasa segmented output and the source diacritized words, and the resulting fully diacritized restored forms.

Source word	Segmented word	Restored word
كَرَّمَ	كرم	كَرَّمَ
الْمُوَظَّفُ	ال+موظف	الْ+مُوَظَّفُ
الْمُدِيرَ	ال+مدير	الْ+مُدِيرَ
لأَنَّهُ	ل+أن+ه	لِ+أَنَّ+هُ
يُجِلُّهُ	يجل+ه	يُجِلُّ+هُ
فَاسْتَعَدَّ	ف+استعد	فَ+اسْتَعَدَّ
الْجَمِيعُ	ال+جميع	الْ+جَمِيعُ
وَانْطَلَقُوا	و+انطلق+وا	وَ+انْطَلَقَ+وا
لِلْإِحْتِفالِ	ل+ال+إحتفال	لِ+اَلْ+إِحْتِفالِ
بِالتَّكْرِيمِ	ب+ال+تكريم	بِ+ال+تَّكْرِيمِ

Step 1: Character labeling Before labeling, each segment is checked by counting Arabic base letters, excluding diacritics and other combining marks. Segments containing two or fewer Arabic base letters are passed forward unchanged. This shortcut does not alter the chunking outcome, since a one-base-letter segment is necessarily a single chunk and a two-base-letter segment does not require splitting under the boundary rules described below. It serves only to avoid unnecessary computation and to preserve short Farasa-delimited segments explicitly.

All remaining segments are labeled letter by letter before any chunking decision is made. Each Arabic base letter, together with the diacritic marks that directly follow it in the Unicode character sequence, is assigned one of two labels: Diacritized (D) or Undiacritized (U). The first Arabic letter of any segment is always labeled D. A subsequent letter is labeled D if it carries at least one diacritic mark other than sukun alone. It is labeled U if it carries no diacritic mark or carries sukun alone.

A special case arises because Farasa does not reliably supply the short vowel diacritic on the letter immediately preceding a long vowel letter. When this diacritic is absent, the algorithm labels the preceding letter as U, which would incorrectly trigger a chunk boundary before the long vowel and split a single syllable into two chunks. To prevent this, the chunker applies an explicit pairing check. When a long vowel letter (ا, ى, ي, or و) is encountered and the immediately preceding letter carries no short vowel diacritic, the long vowel is recognized as a vowel extension rather than an independent letter. The preceding letter is relabeled D so that both remain in the same chunk.

Step 2: Chunk boundary detection After labeling, the chunker scans each segment from left to right. A D-labeled letter opens a new chunk and is absorbed unconditionally. The chunker then examines each following letter in turn to decide whether to absorb it and continue, absorb it and close the chunk, or close the chunk without absorbing it.

A following D-labeled letter that carries neither shadda nor sukun is absorbed and the scan continues. This accumulation of consecutive D-labeled letters is why a fully diacritized sequence such as كَتَبَ (wrote) produces a single chunk, since every letter is labeled D and no closure trigger fires.

Closure is triggered by three conditions. First, if the following letter carries shadda, it is absorbed into the current chunk and the chunk closes. If the letter immediately after the shadda-bearing letter is a bare long vowel, that long vowel is also absorbed before closure, since a shadda-bearing letter and the long vowel that follows it together form one syllable. The word سَبَّاك (plumber) illustrates this: the sequence بَّا is absorbed as one unit, producing the chunk بَّا. Second, if the following letter carries sukun, it is absorbed and the chunk closes. Third, if the following letter is a bare long vowel letter, it is absorbed and the chunk closes. When

Table 3: Diacritic-driven chunking output for the example sentence.

Word	Chunk	Trigger
كَرَّمَ	كَرَّ	Shadda
	مَ	Final merge
الْ	الْ	Sukun
مُوَظَّفُ	مُوَظَّ	Shadda
	فُ	Final merge
الْ	الْ	Sukun
مُدير	مُدي	Long vowel
	ر	Final merge
ل	ل	Short segment
أَنَّ	أَنَّ	Shadda
ه	ه	Short segment
يُجِلُّ	يُجِلُّ	Shadda
ه	ه	Short segment
فَ	فَ	Short segment
اسْتَعَدَّ	اسْ	Sukun
	تَعَدَّ	Shadda
الْ	الْ	Sukun
جَميعُ	جَمي	Long vowel
	عُ	Final merge
وَ	وَ	Short segment
انْطَلَق	انْ	Sukun
	طَلَق	Final merge
وا	وا	Long vowel
ل	ل	Short segment
اَلَّ	اَلَّ	Sukun
إِحْتِفَالِ	إِحْ	Sukun
	تِفا	Long vowel
	لِ	Final merge
ب	ب	Short segment
اَل	اَل	Final merge
تَّكْرِيم	تَّكْ	Shadda+sukun
	رِي	Long vowel
	م	Final merge

none of the three closure conditions fires before the end of a segment, all accumulated letters are emitted as a single chunk, a case referred to as a segment-final merge.

Following the same convention as BPE (Sennrich et al., 2016), a `<SPACE>` prefix is prepended to the first chunk of each word, making word boundaries recoverable from the token sequence alone. Table 3 shows the complete chunking result for the running example.

2.4 Phase 3: Vocabulary Construction and Tokenization

2.4.1 Vocabulary Construction

The vocabulary is constructed by counting the frequency of every distinct chunk produced by Phase 2 across the full training corpus. Every distinct chunk becomes a vocabulary entry. The vocabulary includes two special tokens, `<UNK>` and `<|endoftext|>`. The raw vocabulary produced by this phase contains 250,163 unique entries. Vocabulary pruning and the final truncation to 68 044 entries are described in the following subsection.

2.4.2 Tokenization

a) Vocabulary pruning and extension The Phase 3.1 vocabulary mapping contains 250,163 entries, including the two special tokens. Pruning is applied only to the remaining 250,161 chunk entries. The frequency distribution is highly skewed: 54,349 entries occur once, and 182,375 entries occur at most 100 times. These low-frequency entries form about 73% of the chunk vocabulary, but together account for only 0.0122% of all token occurrences in the 18.5 billion-token corpus. They are therefore removed as full tokens.

Each removed chunk is split into character units. Each unit consists of one Arabic base letter together with its attached diacritic marks. If the chunk starts with `<SPACE>`, this prefix is kept on the first character unit. Non-Arabic characters, including digits, are also represented as individual character units: a numeric string such as `111` is tokenized as three separate digit tokens `1`, `1`, `1`, with no multi-digit grouping. This guarantees that every removed chunk can still be represented. The final vocabulary keeps 67,786 frequent chunk entries, adds 256 character-level fallback units, and includes the two special tokens `<UNK>` and `<|endoftext|>`. The final vocabulary therefore contains 68,044 entries. Of the fallback units, 55 are new entries that were not present in the raw vocabulary. Table 4 summarizes these counts.

b) Token ID mapping After the vocabulary is finalized, tokenization is performed by lookup in the extended vocabulary. Frequent chunks are mapped directly to their token IDs, while pruned chunks are encoded using the character-level fallback units defined above. Each resulting unit is assigned its corresponding integer ID. In the processed corpus, all units were resolved by this procedure, so no `<UNK>` token was produced during tokenization.

Table 4: Vocabulary statistics.

Frequency range	**Entries**	**Cumulative pruned**
= 1	54,349	54,349
2–5	52,257	106,606
6–10	20,976	127,582
11–25	24,702	152,284
26–50	15,966	168,250
51–100	14,125	182,375
Total pruned		182,375
Vocabulary transition		
Raw vocabulary mapping (Phase 3.1)		250,163
Special tokens excluded from pruning		−2
Chunk entries considered		250,161
Entries pruned (≤ 100)		−182,375
Frequent chunk entries retained		67,786
Character-level fallback units		+256
of which new (not in raw)		55
Special tokens		+2
Final vocabulary		**68,044**

3 Dataset

3.1 Training Corpus

This subsection describes the source corpus, the three-stage preprocessing pipeline applied before tokenization, and the resulting corpus statistics.

3.1.1 Source Corpus

FineWeb-Edu-Ar is an Arabic corpus derived from 96 Common Crawl snapshots of English educational web text, translated into Arabic using a 600M parameter machine translation model (Penedo et al., 2024; Alrashed et al., 2024). The full corpus comprises over 180 million documents, filtered using an educational quality scorer to promote cleaner vocabulary and more well-formed sentences relative to raw web text. A subset of 14.4 million documents (74 GB of raw text) was used for pre-training, representing the volume tractable within the available compute budget for a controlled comparison at GPT-2 scale.

3.1.2 Preprocessing Pipeline

Stage 1: Pre-diacritization normalization Prior to diacritization, the raw corpus undergoes a single-pass normalization script organized into three successive operations. First, text noise is removed: URLs, email addresses, scientific notation strings, and malformed numeric tokens are stripped from each document. Second, characters are normalized: Latin and extended Latin characters are removed explicitly, followed by all remaining non-Arabic scripts, including CJK (Chinese, Japanese, and Korean), Cyrillic, and Hebrew, via a character filter that retains Arabic Unicode blocks and standard punctuation. Following character normalization, repeated punctuation sequences and special characters are collapsed to a single instance, and whitespace is normalized. Third, deduplication is applied at two levels: at the sentence level, where sentences below ten characters are excluded as they fall below the minimum length threshold for reliable n-gram matching, and at the n-gram level with $n \in [2, 10]$ using a loop that iterates until no further changes occur. Throughout all steps, Alef variants (أ, إ, آ, ٱ) are explicitly preserved, as they encode grammatical distinctions that the downstream diacritizer requires to better represent the language.

Table 5: Pre-training corpus statistics.

Statistic	Value
Total documents	14,359,494
Training documents	13,641,519
Validation documents	717,975
Total words (billions)	6.83
Avg. words per document	475.9
Avg. document length (chars)	4,346
Diacritized words (billions)	6.39
Diacritization coverage*	93.45%

Stage 2: Diacritization The normalized corpus is diacritized using the diacritization component of the Farasa toolkit (Darwish & Mubarak, 2016).

Stage 3: Post-diacritization cleaning Following diacritization, consecutive duplicate diacritics on the same letter are eliminated, resolving orthographically invalid sequences that occasionally arise when Farasa applies the same diacritic twice to a single letter. A final NFC pass is then applied to ensure codepoint consistency before tokenization.

3.1.3 Corpus Statistics

The final corpus was partitioned into training and validation sets at a 95%/5% ratio following a full document-level random shuffle (seed = 123). Table 5 reports the key statistics. The corpus achieves a diacritization coverage of 93.45% over Arabic word tokens. The remaining 6.55% consists predominantly of words whose vowel quality is already recoverable from their consonant and long-vowel structure (e.g., على, فيها, الماضي), short common words such as prepositions and conjunctions (e.g., في, من, لا), foreign-origin terms transliterated into Arabic script (e.g., إنترنت (Internet), كمبيوتر (computer)), and a residual portion of Arabic vocabulary left undiacritized by Farasa.

3.2 Evaluation Benchmarks

Downstream evaluation uses multiple-choice question (MCQ) benchmarks. The model scores each answer option independently by computing the loss of that option conditioned on the question, and selects the option with the lowest loss. Before evaluation, three filtering steps were applied across all benchmarks. First, questions exceeding the model context length of 1,024 tokens were excluded, as they cannot be processed within the fixed-length input window. Second, questions containing English words were excluded, since neither model was trained on English text and the presence of such words could confound model decisions. Third, questions requiring simultaneous inspection of all answer options were excluded when no consistent prompt format could be defined to make them scorable on a per-option basis. All benchmarks were diacritized using the same Farasa pipeline applied to the training corpus, maintaining consistent input distributions between training and evaluation.

The benchmarks are organized into three groups: Arabic linguistic understanding, general knowledge, and mathematical reasoning. Table 6 provides an overview of all benchmarks with original and final question counts after filtering.

3.2.1 GAT

The General Aptitude Test (GAT) benchmark (AlBallaa et al., 2025) comprises two sets of subtasks: five GATLc subtasks targeting verbal reasoning and reading comprehension in Arabic, which form the Arabic

Table 6: Evaluation benchmark summary. Orig.: original question count before filtering. Final: count after all filtering stages.

Benchmark	Orig.	Final
Arabic linguistic understanding		
GAT Reading Comprehension	2,650	2,329
GAT Verbal Analogy	2,750	2,750
GAT Sentence Completion	1,215	1,215
GAT Contextual Error	1,309	1,309
GAT Semantic Association	1,050	1,050
ArabicMMLU Arabic Language	678	643
ArabicMMLU Arabic Lang. General	615	578
ArabicMMLU Arabic Lang. Grammar	368	351
TAAG (Teacher-Authored Grammar)	171	171
Nahw-MCQ Level 1 (easy)	1,284	1,083
Nahw-MCQ Level 2 (medium)	1,685	1,293
Nahw-MCQ Level 3 (hard)	2,031	1,560
General knowledge		
ArabicMMLU General (18 subsets)	12,514	11,615
Mathematical reasoning		
GAT Algebra	2,700	2,579
GAT Arithmetic	2,722	2,653
GAT Geometry	370	350
GAT Comparisons	1,225	1,209

linguistic understanding group, and four GATmath subtasks covering algebra, arithmetic, geometry, and comparisons, which form the mathematical reasoning group.

Several GATLc subtasks required formatting adjustments before evaluation. Because each GATLc subtask covers a single uniform question type, one standardized Arabic prompt could be applied to all questions within the subtask. This is in contrast to other benchmarks where questions requiring simultaneous inspection of all answer options do not follow a consistent pattern, making prompt standardization impractical. In GAT, the consistent structure of each subtask made it possible to reformat rather than exclude such questions. For reading comprehension, the passage was merged into the question field. For sentence completion and contextual error detection, a standardized Arabic prompt was prepended since no explicit question was present in the original questions. For verbal analogy and semantic association, the answer options were embedded inline within the question field and reordered using a fixed seed (seed = 42) to achieve a balanced correct-answer distribution across all four positions (A: 25%, B: 25%, C: 25%, D: 25%). Embedding the options within the question field makes independent per-option scoring valid, since each option is evaluated against a question that already contains the full option set as context. To confirm that prompt format does not influence results, multiple prompt variations were tested, including the original unmodified input, and all variations produced equivalent scores. The GATmath subtasks required no format changes beyond context-length filtering. Table 7 summarizes the adjustments applied to the four GATLc subtasks that required reformatting.

3.2.2 ArabicMMLU

ArabicMMLU (Koto et al., 2024) covers 21 subject areas organized into 21 subsets. Three subsets form the linguistic evaluation group: Arabic Language, Arabic Language General, and Arabic Language Grammar. The Arabic Language subset is itself an aggregation of primary, middle, and high school Arabic curriculum materials. Arabic Language General and Arabic Language Grammar are independent subsets covering general language use and grammatical knowledge respectively. The remaining 18 subsets cover general knowledge domains and form the general knowledge group. Across all 21 subsets, a number of questions requiring simultaneous inspection of all answer options were identified by keyword patterns and excluded,

Table 7: GAT format adjustments: before and after examples for the four GATLc subtasks that required reformatting prior to evaluation. Shaded cells show the original item format.

Subtask	Before (original)	After (reformatted)
Sentence completion	الوعد ..... و..... المطر.	استناداً إلى الجملة التالية: "الوعد ..... و..... المطر."، املأ الفراغ بالكلمة الناقصة. الجواب هو:
Contextual error	إن مفتاح الأمور الأعمال	استناداً إلى الجملة التالية: "إن مفتاح الأمور الأعمال"، اختر الكلمة الخارجة عن سياقها. الجواب هو:
Verbal analogy	مرتفع : منخفض وسيم:قوي ج: قراءة:كتابة ب: شمال:جنوب أ: أبيض:أزرق د:	استناداً إلى الكلمتين: "مرتفع : منخفض"، اختر من الكلمات التالية تلك التي تحاكي هاتين الكلمتين. شمال:جنوب، قراءة:كتابة، وسيم:قوي، أبيض:أزرق. الجواب هو:
Semantic association	اختر المفردة الشاذة الجزائر د: ليبيا ج: مصر ب: سوريا أ:	اختر المفردة الشاذة؟ مصر، ليبيا، الجزائر، سوريا. الجواب هو:

as they are not scorable on a per-option basis. For questions that included a context field alongside the question text, the context was prepended to the question before evaluation.

3.2.3 Teacher-Authored Arabic Grammar (TAAG)

TAAG comprises 171 multiple-choice questions developed specifically for this study from Arabic language teaching materials spanning grades 7 through 9, covering Arabic grammar and syntax (نحو), morphology (صرف), and rhetoric (بلاغة). The questions were designed and validated by a certified Arabic language teacher holding a degree in Arabic language, to ensure topic diversity, avoid question repetition, and guarantee that each question accurately measures Arabic linguistic competence. No items were removed at any filtering stage.

3.2.4 Nahw-MCQ

Nahw-MCQ is the multiple-choice component of the Nahw benchmark introduced by Mubarak et al. (2026), a comprehensive Arabic grammar evaluation suite targeting Modern Standard Arabic grammatical knowledge. The benchmark was constructed entirely from naturally authored Arabic text and covers Arabic grammar theory and practical rule application across three difficulty levels: easy, medium, and hard. The three-level structure reflects the range of grammatical complexity found in Arabic language instruction, from basic morphological agreement to complex case assignment and syntactic dependency rules. The full dataset contains 5,000 questions. After applying the standard filtering steps described above, 3,936 questions remain: 1,083 at the easy level, 1,293 at the medium level, and 1,560 at the hard level. A portion of the dropped questions was attributable to the third filtering criterion: questions requiring simultaneous inspection of all answer options were identified by keyword patterns specific to Nahw-MCQ question formats and excluded, as they are not scorable on a per-option basis.

4 Experimental Setup

4.1 Tokenization

The diacritized FineWeb-Edu-Ar corpus was tokenized independently using NOHA and a BPE tokenizer trained on the same diacritized corpus. The BPE vocabulary size was set to 68,044 to match NOHA's constructed vocabulary exactly, ensuring that any observed performance difference is attributable to tokenization strategy rather than vocabulary capacity.

Due to NOHA's higher token fertility (1.49× higher than BPE), the tokenized corpus contains approximately 12.4 billion tokens under BPE and 18.5 billion tokens under NOHA over the full corpus. The tokenized corpus

Table 8: Training hyperparameters used for both models.

Hyperparameter	**Value**
Optimizer	AdamW
Peak learning rate	6×10^{-4}
β_1, β_2	0.9, 0.95
ε	1×10^{-8}
Weight decay	$0.1^{\dagger}$
Gradient clipping	max norm 1.0
LR schedule	Linear warmup + cosine decay
Warmup steps	max(2000, 2% $\times$ total steps)
Minimum LR	6×10^{-5} (10% of peak)
Batch size (per device)	4
Gradient accumulation	16 steps
Effective batch size	65,536 tokens
Precision	FP16 mixed precision‡
Epochs	2
Evaluation interval	every 10,000 steps

†Applied to 2D+ tensors only.
‡Implemented with GradScaler.

was concatenated with `<|endoftext|>` document separators and packed into fixed-length sequences of 1,024 tokens.

4.2 Model Architecture

A GPT-2-style transformer architecture (Radford et al., 2019) was adopted for both models within the available compute budget for a controlled comparison. Both models are architecturally identical in every respect: 12 transformer layers, 12 attention heads, an embedding dimension of 768, a feed-forward dimension of 3,072, GELU activation, a dropout rate of 0.1, and a context length of 1,024 tokens. The total parameter count is 190.33M for both models, exceeding the standard GPT-2-small parameter count of 117M (based on the original vocabulary of 50,257 tokens) due to the larger shared vocabulary of 68,044 entries.

4.3 Training Configuration

Both models are trained for two full epochs. Given the token counts reported in Section 4.1, two epochs correspond to 378,000 steps for BPE and 565,690 steps for NOHA. To maintain an equal compute budget, both models are evaluated at the same training steps from step 10,000 to step 378,000 at intervals of 10,000, producing 38 matched-compute evaluation checkpoints. At each checkpoint, both models have executed the same number of floating-point operations. Since NOHA produces more tokens per word, each context window of 1,024 tokens covers proportionally fewer source words, meaning NOHA processes approximately 33% fewer unique documents than BPE at any given checkpoint.

Training for two full epochs also enables a second evaluation framework based on data coverage rather than compute: the epoch-aligned checkpoints compare both models after each has completed the same number of passes over the training corpus, removing the document-coverage asymmetry caused by NOHA's higher token fertility. BPE completes epoch 1 at step 189,281 and epoch 2 at step 378,562. NOHA completes epoch 1 at step 282,845 and epoch 2 at step 565,690.

All hyperparameters are identical across both models and are reported in Table 8.

4.4 Evaluation Methodology

4.4.1 Intrinsic Metric: Bits-per-Byte

Bits-per-byte (BPB) is used as the primary intrinsic metric. BPB measures how well the model predicts each byte of the original text: lower values indicate better prediction, and the per-byte normalization ensures that the score is independent of how many tokens each tokenizer uses to cover the same text. Without this normalization, NOHA would appear to have lower loss than BPE simply because its smaller tokens are individually easier to predict, not because its model understands the text better. BPB is computed from the token-level cross-entropy loss as follows:

$$\text{BPB} = \frac{\mathcal{L}_{\text{token}} \times N_{\text{tokens}}}{N_{\text{bytes}} \times \ln 2} \tag{1}$$

where $\mathcal{L}_{\text{token}}$ is the mean token-level cross-entropy loss, N_{tokens} is the total number of tokens in the evaluation text, and N_{bytes} is its total byte count. Both tokenizers are applied to the same text, so BPB ensures the comparison reflects model quality rather than tokenizer fertility.

4.4.2 Benchmark Evaluation Metric: Length-Normalized Accuracy

All multiple-choice benchmarks are evaluated using length-normalized accuracy. For each candidate answer, the sum of log-probabilities of the answer tokens conditioned on the question is divided by the number of tokens in that answer. The candidate with the highest normalized score is selected. This normalization removes the length bias inherent in multiple-choice evaluation: without it, a model would systematically favor shorter answer options regardless of their correctness, since shorter token sequences accumulate smaller total log-probability penalties. The benchmarks used are described in Section 3.

4.4.3 Consistency of Results Across Checkpoints

For each benchmark, the accuracy difference between NOHA and BPE is computed at every checkpoint as NOHA minus BPE. Results are summarized by the mean difference across all 38 checkpoints and the number of checkpoints at which NOHA leads, providing a direct measure of whether an advantage holds throughout training or appears only at isolated points.

5 Results

5.1 Compression Efficiency: Bits-per-Byte

BPE achieves lower validation BPB than NOHA at every one of the 38 evaluation checkpoints, with a mean relative gap of +2.87% across the full training trajectory. The gap narrows early as both models converge rapidly, reaches a minimum of +1.66% at step 50,000, then widens steadily to +3.91% at the final checkpoint (step 378,000). Training BPB tracks validation BPB closely for both models throughout training, confirming that neither model overfits. Figure 3 shows both training and validation BPB curves for both models. Although BPE compresses the text more efficiently, the question of whether this compression advantage translates into better language understanding is addressed in the downstream evaluation that follows.

5.2 Downstream Evaluation

The downstream evaluation assesses whether NOHA's linguistically motivated tokenization strategy leads to better Arabic language understanding than BPE. Both models are evaluated on multiple-choice benchmarks organized into three groups: Arabic linguistic understanding, general knowledge, and mathematical reasoning. Table 9 reports mean accuracy across all 38 evaluation checkpoints, together with accuracy at the end of each training epoch. Each 38-checkpoint mean is accompanied by the number of checkpoints at which NOHA leads BPE.

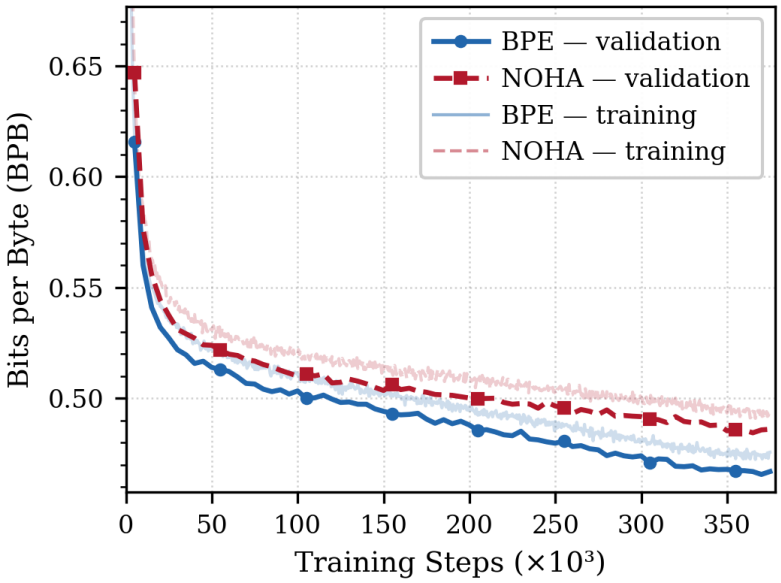

Figure 3: Training and validation BPB for NOHA and BPE over 378,000 steps.

Table 9: Combined evaluation results. Left block: mean length-normalized accuracy (%) across 38 checkpoints with number of checkpoints at which NOHA leads BPE (out of 38). Right blocks: accuracy at epoch-aligned checkpoints. N = NOHA, B = BPE, Δ = NOHA minus BPE. **Bold** = better value in each column pair. Gray rows mark near-chance performance.

	38-ckpt mean				Epoch 1			Epoch 2		
Benchmark	**N**	**B**	**Δ**	**Leads**	**N**	**B**	**Δ**	**N**	**B**	**Δ**
Arabic linguistic understanding										
GAT Reading Comprehension	**31.42**	29.00	+2.42	38/38	**31.56**	28.90	+2.66	**31.34**	29.58	+1.76
ArabicMMLU Arabic Language	**29.51**	27.85	+1.66	34/38	**29.55**	27.99	+1.56	**30.17**	27.22	+2.95
ArabicMMLU Arabic Lang. General	**34.91**	30.30	+4.60	38/38	**35.99**	30.97	+5.02	**35.12**	30.80	+4.32
ArabicMMLU Arabic Lang. Grammar	**31.97**	30.28	+1.69	32/38	**32.48**	31.91	+0.57	**31.91**	30.77	+1.14
TAAG	**29.15**	26.61	+2.54	30/38	**32.16**	25.73	+6.43	**30.41**	24.56	+5.85
Nahw-MCQ Level 1†	**25.31**	23.44	+1.87	38/38	**25.76**	24.19	+1.57	**26.04**	22.62	+3.42
Nahw-MCQ Level 2†	25.05	**25.67**	−0.62	11/38	25.60	**25.91**	−0.31	24.21	**24.98**	−0.77
Nahw-MCQ Level 3†	**25.91**	25.51	+0.40	24/38	**25.90**	23.85	+2.05	**26.60**	24.87	+1.73
GATLc Verbal Analogy†	25.40	**26.51**	−1.11	2/38	25.42	**26.15**	−0.73	25.64	**27.05**	−1.41
GATLc Sentence Completion†	25.41	**26.06**	−0.64	7/38	**25.93**	25.43	+0.50	**26.67**	25.76	+0.91
GATLc Contextual Error†	**23.43**	21.57	+1.86	38/38	**22.84**	21.70	+1.14	**22.99**	21.08	+1.91
GATLc Semantic Association†	**25.57**	23.42	+2.15	37/38	**25.33**	24.29	+1.04	**25.90**	23.90	+2.00
General knowledge										
ArabicMMLU General (18 subsets)[a]	31.09	**31.35**	−0.26	12/38	30.98	**31.45**	−0.48	30.53	**31.49**	−0.96
Mathematical reasoning† (most subtasks near chance)										
GAT Algebra	23.96	23.97	≈0	18/38	**24.08**	23.50	+0.58	**24.43**	23.07	+1.36
GAT Arithmetic	**22.95**	21.90	+1.05	30/38	**23.63**	22.47	+1.16	**24.05**	22.73	+1.32
GAT Geometry†	**27.59**	23.55	+4.04	37/38	**29.71**	23.14	+6.57	**30.57**	24.00	+6.57
GAT Comparisons[b]	**29.10**	19.35	+9.74	38/38	**28.37**	18.20	+10.17	**29.03**	18.53	+10.50

†Random baseline = 25.0% (four-choice MCQ).
[a]Weighted average over 18 subsets (weighted by question count).
[b]GAT Comparisons results reflect opposing tokenization priors on fixed answer options; see Section 6.3.

5.2.1 Arabic Linguistic Understanding

NOHA achieves higher average accuracy than BPE on all three targeted Arabic linguistic benchmarks, with advantages that are consistent throughout training. NOHA leads on GAT Reading Comprehension (+2.42%), TAAG (+2.54%), and all three ArabicMMLU Arabic Language subsets, with the largest gap on Arabic Language General (+4.60%).

On Nahw-MCQ, both models score within one percentage point of the 25% random baseline at all three difficulty levels, indicating that neither model has acquired meaningful Arabic grammar knowledge at 190 M parameters. The accuracy gap between NOHA and BPE on these benchmarks reflects directional trends in near-chance performance rather than evidence of grammatical competence. On the four near-chance GATLc subtasks, the mixed directional pattern provides no basis for drawing conclusions about tokenizer differences. The epoch-aligned columns in Table 9 confirm that NOHA leads on all three targeted Arabic linguistic benchmarks at both epoch 1 and epoch 2, confirming that the advantage persists under equal data-coverage conditions.

5.2.2 General Knowledge

On the 18 ArabicMMLU general knowledge subsets (weighted by question count per subset), BPE leads by 0.26% on a weighted average. The epoch-aligned checkpoints provide a complementary data-coverage comparison in which both models have processed the same number of training documents rather than the same number of gradient updates, and are therefore not directly comparable to the 38-checkpoint mean. Under this comparison, the gap widens: 0.48% at epoch 1 and 0.96% at epoch 2. Both values confirm that the general knowledge gap remains substantially smaller than NOHA's linguistic advantages under both comparison frameworks, and the three-way dissociation holds.

5.2.3 Mathematical Reasoning

Both models perform at or near the 25% random baseline on all four GAT mathematical reasoning subtasks. For mathematical reasoning subtasks, limited exposure to mathematical content in the training corpus is an additional contributing factor. On GAT Comparisons, BPE scores substantially below the random baseline at 19.35%, while NOHA scores above it at 29.10%. The mechanism behind this gap is discussed in Section 6.3. The epoch-aligned results confirm both models remain near chance at each epoch.

6 Discussion

6.1 Dissociation of Compression and Linguistic Understanding

The results reveal a three-way dissociation: BPE achieves better compression throughout training, NOHA achieves higher average accuracy on Arabic linguistic benchmarks, and general knowledge capacity is largely preserved under NOHA tokenization. The general knowledge gap of 0.26% in BPE's favour indicates that NOHA preserves general knowledge capacity while improving linguistic understanding. This pattern confirms that compression efficiency alone does not predict downstream linguistic performance. Lotz et al. (2025) reach a similar conclusion across several non-Arabic languages, suggesting the dissociation is not specific to Arabic. The present study provides among the first such evidence for a morphologically complex language with a diacritic-rich writing system.

6.2 Why Linguistic Benchmarks Benefit

The NOHA advantage on Arabic linguistic benchmarks is attributable to two properties of its design. First, NOHA's diacritic-driven chunking produces tokens whose boundaries align with phonological and morphological structure. Each boundary corresponds to a linguistically motivated edge: a clitic boundary, a syllable closure triggered by sukun or shadda, or a long vowel extension. BPE boundaries are determined by co-occurrence frequency and may cut through morphological or syllable units depending on local corpus statistics. By breaking words into morphologically motivated units rather than statistically derived subwords,

NOHA may allow the model to learn more stable morpho-syntactic patterns. Second, NOHA preserves Arabic clitics such as بِـ وَ الْ and لِـ as atomic vocabulary entries, ensuring that these grammatically significant function words are never split across token boundaries regardless of context. BPE may merge or split these units depending on local co-occurrence statistics, introducing representational inconsistency for the words that carry the most syntactic information in Arabic.

The largely preserved general knowledge performance indicates that NOHA's benefit is specific to tasks requiring Arabic linguistic processing rather than a general improvement in language modeling capacity, which is consistent with the design intent.

6.3 Near-Chance Performance on GATLc, Mathematical Reasoning, and Nahw-MCQ

Both models perform at or near the 25% random baseline on the four GATLc subtasks: Verbal Analogy, Sentence Completion, Contextual Error, and Semantic Association, and on most GAT mathematical reasoning subtasks. These subtasks come from a standardized university entrance examination and require reasoning capacity well beyond 190 M parameters. For the GATLc subtasks, symmetric near-chance performance reflects model scale limitations. For mathematical reasoning subtasks, limited exposure to mathematical content in the training corpus is an additional contributing factor.

GAT Comparisons is an exception that requires specific explanation. NOHA scores 29.10% while BPE scores 19.35%, falling substantially below the random baseline. Analysis reveals that the GAT Comparisons subtask uses identical answer options across all items. The same four fixed Arabic relational phrases appear regardless of which question is asked. NOHA's deterministic chunking assigns identical token boundaries to these fixed phrases at every occurrence, producing consistent log-probability values that create a systematic prior for option ب, which matches the trivial always-ب baseline of 29.20% exactly. BPE's below-chance score reflects the inverse: BPE assigns systematically higher log-probability to a less frequent correct answer option, likely because BPE tokenizes certain fixed answer phrases into fewer or more common tokens, artificially lowering their loss. Neither model is responding to question content. The reported gap is entirely an artifact of opposing tokenization priors on a fixed-option benchmark and carries no implications for mathematical reasoning ability. To verify that this prior-driven mechanism does not operate on the headline benchmarks, the position-skew of each model's predictions was examined across all reported benchmarks. The four targeted linguistic benchmarks show position-skews of 0.26 to 0.35, qualitatively unlike the 0.99 to 1.00 observed on GAT Comparisons, confirming that the artifact is isolated to the fixed-option subtask and is absent from the benchmarks on which NOHA's advantage is claimed.

The Nahw-MCQ results are near chance across all three difficulty levels. State-of-the-art models score at most 67% on Nahw-MCQ (Mubarak et al., 2026), confirming that this benchmark remains challenging even at orders-of-magnitude larger scale. Near-chance performance at 190 M parameters reflects a model-scale ceiling, not a tokenization deficiency.

6.4 Limitations

Several limitations bound the scope of the present findings.

Model scale. All experiments are conducted at the 190 M parameter scale. Whether NOHA's linguistic performance advantages persist, widen, or narrow at larger model scales remains an open question, as does whether the observed dissociation between compression efficiency and linguistic understanding is maintained. The present study establishes the existence and direction of the effect under strictly controlled conditions where the tokenizer is the sole experimental variable. Prior work suggests that tokenizer effects on downstream performance may either amplify or diminish with scale depending on the task (Lotz et al., 2025).

Diacritization quality. The preprocessing pipeline uses Farasa for diacritization, which achieved state-of-the-art performance at the time of its release but has since been surpassed by neural approaches. Mohamed & Mubarak (2025) report a BiLSTM-based diacritizer that achieves 2.70% word error rate on a 2024 Arabic news benchmark, substantially lower than Farasa's error rate. Since NOHA's chunking rules depend directly on diacritic accuracy, diacritization errors propagate into incorrect token boundaries. Training on a corpus

diacritized with a more accurate model could strengthen the training signal and potentially widen NOHA's advantage on linguistic benchmarks.

Diacritized input. NOHA requires fully diacritized input and is evaluated on models trained exclusively on diacritized text. The majority of Arabic text in practical deployment is undiacritized. Whether the advantages demonstrated here transfer to continued pre-training or fine-tuning on undiacritized text is not empirically validated in this work, though Arabic literacy research provides theoretical motivation for expecting positive transfer (Issa, 2023).

Corpus origin. The training corpus is derived from machine-translated English educational web text. Both models train on identical text, so the internal comparison is valid, but the corpus may not fully represent the morphological and lexical diversity of native Arabic prose.

7 Conclusion

This paper introduced NOHA, an Arabic tokenizer with fully deterministic chunking given segmented diacritized input, replacing statistical subword discovery with diacritic-driven rules grounded in Arabic phonological and morphological structure. NOHA combines acquisition-motivated clitic preservation with diacritically guided chunking, producing a 68,044-entry vocabulary derived from corpus frequencies without any statistical merge procedure.

A controlled pre-training comparison at the 190 M parameter scale demonstrates a three-way dissociation across 38 matched-compute evaluation checkpoints. On Arabic linguistic benchmarks, NOHA achieves higher average accuracy than a matched BPE baseline: +2.42% on GAT Reading Comprehension, +2.65% on the three ArabicMMLU Arabic Language subsets (simple mean), and +2.54% on TAAG, a Teacher-Authored Arabic Grammar benchmark developed for this study. On general knowledge, BPE leads by 0.26% on the weighted average across 18 ArabicMMLU subsets, indicating that NOHA preserves general knowledge capacity while improving linguistic understanding. BPE achieves lower BPB at all 38 matched-compute checkpoints, with a mean compression advantage of +2.87% that grows to +3.91% by step 378,000. The persistence of NOHA's linguistic advantages despite this growing compression disadvantage provides empirical evidence that compression efficiency and linguistic understanding are dissociable properties of Arabic tokenizer design, and that linguistically grounded tokenization yields measurable improvements on Arabic linguistic tasks at an identical computational budget.

Future work should evaluate NOHA at larger model scales, assess transfer to undiacritized Arabic through continued pre-training, examine the effect of higher-accuracy diacritization on NOHA's tokenization quality, and extend the evaluation to models trained on native Arabic corpora.

Broader Impact Statement

Arabic is a language spoken by hundreds of millions of people across 22 countries. Improved Arabic language modeling has the potential to benefit Arabic speakers in education, information access, and digital inclusion. The training corpus (FineWeb-Edu-Ar) is a publicly available dataset of machine-translated educational web text containing no personally identifiable information. As noted in the limitations, the corpus may not fully represent the morphological and lexical diversity of native Arabic prose, which could limit generalization across dialects and registers. The authors are not aware of any meaningful risks of harm arising from this research.

References

Hussein S. Al-Olimat and Ahmad Alshareef. ALPS: A Diagnostic Challenge Set for Arabic Linguistic & Pragmatic Reasoning, February 2026. URL `http://arxiv.org/abs/2602.17054`. arXiv:2602.17054 [cs].

Safa AlBallaa, Nora AlTwairesh, Abdulmalik AlSalman, and Sultan Alfarhood. GATmath and GATLc: Comprehensive benchmarks for evaluating Arabic large language models. *PLOS One*, 20(9):e0329129, September 2025. ISSN 1932-6203. doi: 10.1371/journal.pone.0329129. URL `https://dx.plos.org/10.1371/journal.pone.0329129`.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The Falcon Series of Open Language Models, November 2023. URL http://arxiv.org/abs/2311.16867. arXiv:2311.16867 [cs].

Sultan Alrashed, Dmitrii Khizbullin, and David R. Pugh. Fineweb-Edu-Ar: Machine-translated Corpus to Support Arabic Small Language Models, November 2024. URL http://arxiv.org/abs/2411.06402. arXiv:2411.06402 [cs].

Mohamed Taher Alrefaie, Nour Eldin Morsy, and Nada Samir. Exploring tokenization strategies and vocabulary sizes for enhanced arabic language models, 2024. URL https://arxiv.org/abs/2403.11130.

Zaid Alyafeai, Maged S. Al-shaibani, Mustafa Ghaleb, and Irfan Ahmad. Evaluating various tokenizers for arabic text classification, 2021. URL https://arxiv.org/abs/2106.07540.

Wissam Antoun, Fady Baly, and Hazem Hajj. AraGPT2: Pre-trained transformer for Arabic language generation. In Nizar Habash, Houda Bouamor, Hazem Hajj, Walid Magdy, Wajdi Zaghouani, Fethi Bougares, Nadi Tomeh, Ibrahim Abu Farha, and Samia Touileb (eds.), *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pp. 196--207, Kyiv, Ukraine (Virtual), April 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.wanlp-1.21/.

Mohamed Anwar, Abdelhakim Freihat, George Ibrahim, Mostafa Awad, Abdelrahman Atef Mohamed Ali Sadallah, Gurpreet Gosal, Gokul Ramakrishnan, Sarath Chandran, Biswajit Mishra, Rituraj Joshi, Ahmed Frikha, Etienne Goffinet, Abhishek Maiti, Ali El Filali, Sarah Al Barri, Samujjwal Ghosh, Rahul Pal, Parvez Mullah, Awantika Shukla, Sajid Siddiki, Samta Kamboj, Onkar Pandit, Sunil Sahu, Abelrahman El Badawy, Amr Mohamed, Ahmad Chamma, Evan Dufraisse, Abdelaziz Bounhar, Dani Bouch, Hadi Abdine, Guokan Shang, Fajri Koto, Yuxia Wang, Zhuohan Xie, Ali Mekky, Rania Hossam Elmohamady Elbadry, Sarfraz Ahmad, Momina Ahsan, Omar Emad Mohamed El-Herraoui, Daniil Orel, Hasan Iqbal, Kareem Mohamed Naguib Abdelmohsen Fahmy Elzeky, Mervat Abassy, Kareem Ali, Saadeldine Eletter, Farah Atif, Nurdaulet Mukhituly, Haonan Li, Xudong Han, Aaryamonvikram Singh, Zain Quraishi, Neha Sengupta, Larry Murray, Avraham Sheinin, Joel Hestness, Natalia Vassilieva, Hector Ren, Zhengzhong Liu, Michalis Vazirgiannis, and Preslav Nakov. Jais 2: A family of Arabic-centric open large language models. Technical report, IFM, December 2025.

Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. Morphbpe: A morpho-aware tokenizer bridging linguistic complexity for efficient llm training across morphologies. *arXiv preprint arXiv:2502.00894*, 2025.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen Technical Report, September 2023. URL http://arxiv.org/abs/2309.16609. arXiv:2309.16609 [cs].

M Saiful Bari, Yazeed Alnumay, Norah A. Alzahrani, Nouf M. Alotaibi, Hisham Abdullah Alyahya, Sultan AlRashed, Faisal Abdulrahman Mirza, Shaykhah Z. Alsubaie, Hassan A. Alahmed, Ghadah Alabduljabbar, Raghad Alkhathran, Yousef Almushayqih, Raneem Alnajim, Salman Alsubaihi, Maryam Al Mansour, Saad Amin Hassan, Dr. Majed Alrubaian, Ali Alammari, Zaki Alawami, Abdulmohsen Al-Thubaity, Ahmed Abdelali, Jeril Kuriakose, Abdalghani Abujabal, Nora Al-Twairesh, Areeb Alowisheq, and Haidar Khan. ALLam: Large language models for arabic and english. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MscdsFVZrN.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Kareem Darwish and Hamdy Mubarak. Farasa: A new fast and accurate Arabic word segmenter. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1070--1074, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1170/.

Nizar Y. Habash. *Introduction to Arabic Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, San Rafael, CA, 2010. ISBN 978-1-59829-795-9. doi: 10.2200/S00277ED1V01Y201008HLT010. URL https://aclanthology.org.

Iyad Issa. Morphological Complexity in Arabic Spelling and Its Implication for Cognitive Processing. *Journal of Psycholinguistic Research*, 52(1):331--357, February 2023. ISSN 1573-6555. doi: 10.1007/s10936-022-09896-2. URL https://doi.org/10.1007/s10936-022-09896-2.

Fajri Koto, Haonan Li, Sara Shatnawi, Jad Doughman, Abdelrahman Sadallah, Aisha Alraeesi, Khalid Almubarak, Zaid Alyafeai, Neha Sengupta, Shady Shehata, Nizar Habash, Preslav Nakov, and Timothy Baldwin. ArabicMMLU: Assessing massive multitask language understanding in Arabic. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 5622--5640, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.334. URL https://aclanthology.org/2024.findings-acl.334/.

Anis Koubaa, Adel Ammar, Lahouari Ghouti, Omar Najar, and Serry Sibaee. ArabianGPT: Native Arabic GPT-based Large Language Model, February 2024. URL http://arxiv.org/abs/2402.15313. arXiv:2402.15313 [cs].

Juhao Liang, Zhenyang Cai, Jianqing Zhu, Huang Huang, Kewei Zong, Bang An, Mosen Alharthi, Juncai He, Lian Zhang, Haizhou Li, Benyou Wang, and Jinchao Xu. Alignment at pre-training! towards native alignment for arabic LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=woRFmNJiLp.

Jonas F. Lotz, António V. Lopes, Stephan Peitz, Hendra Setiawan, and Leonardo Emili. Beyond text compression: Evaluating tokenizers across scales. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32155--32173, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1546. URL https://aclanthology.org/2025.acl-long.1546/.

Abubakr Mohamed and Hamdy Mubarak. Advancing Arabic diacritization: Improved datasets, benchmarking, and state-of-the-art models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16707--16719, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.846. URL https://aclanthology.org/2025.emnlp-main.846/.

Hamdy Mubarak, Majd Hawasly, and Abubakr Mohamed. Nahw: A comprehensive benchmark of Arabic grammar understanding, error detection, correction, and explanation. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6310--6328, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.296. URL https://aclanthology.org/2026.eacl-long.296/.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. doi: 10.48550/arXiv.2303.08774. URL https://arxiv.org/abs/2303.08774.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. Byte latent transformer: Patches scale better than tokens. In Wanxiang Che, Joyce Nabende,

Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9238--9258, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.453. URL https://aclanthology.org/2025.acl-long.453/.

Guilherme Penedo, Hynek Kydlíek, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale, October 2024. URL http://arxiv.org/abs/2406.17557. arXiv:2406.17557 [cs].

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

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3118--3135, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.243. URL https://aclanthology.org/2021.acl-long.243/.

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. Tokenization is more than compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 678--702, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.40. URL https://aclanthology.org/2024.emnlp-main.40/.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715--1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162/.

Kevin Slagle. SpaceByte: Towards Deleting Tokenization from Large Language Modeling. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 124925--124950. Curran Associates, Inc., 2024. doi: 10.52202/079017-3967. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e1f418450107c4a0ddc16d008d131573-Paper-Conference.pdf.

FANAR TEAM, Ummar Abbas, Mohammad Shahmeer Ahmad, Minhaj Ahmad, Abdulaziz Al-Homaid, Anas Al-Nuaimi, Enes Altinisik, Ehsaneddin Asgari, Sanjay Chawla, Shammur Chowdhury, Fahim Dalvi, Kareem Darwish, Nadir Durrani, Mohamed Elfeky, Ahmed Elmagarmid, Mohamed Eltabakh, Asim Ersoy, Masoomali Fatehkia, Mohammed Qusay Hashim, Majd Hawasly, Mohamed Hefeeda, Mus'ab Husaini, Keivin Isufaj, Soon-Gyo Jung, Houssam Lachemat, Ji Kim Lucas, Abubakr Mohamed, Tasnim Mohiuddin, Basel Mousi, Hamdy Mubarak, Ahmad Musleh, Mourad Ouzzani, Amin Sadeghi, Husrev Taha Sencar, Mohammed Shinoy, Omar Sinan, and Yifan Zhang. Fanar 2.0: Arabic generative ai stack, 2026. URL https://arxiv.org/abs/2603.16397.

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

Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M Rush. Mambabyte: Token-free selective state space model. *arXiv preprint arXiv:2401.13660*, 2024.

Janet CE Watson. *The phonology and morphology of Arabic*. OUP Oxford, 2002.

LILI YU, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 78808--78823. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/f8f78f8043f35890181a824e53a57134-Paper-Conference.pdf.

Jianqing Zhu, Huang Huang, Zhihang Lin, Juhao Liang, Zhengyang Tang, Khalid Almubarak, Mosen Alharthi, Bang An, Juncai He, Xiangbo Wu, Fei Yu, Junying Chen, Ma Zhuoheng, Yuhao Du, He Zhang, Saied Alshahrani, Emad A. Alghamdi, Lian Zhang, Ruoyu Sun, Haizhou Li, Benyou Wang, and Jinchao Xu. Second language (Arabic) acquisition of LLMs via progressive vocabulary expansion. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2025--2042, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.100. URL https://aclanthology.org/2025.acl-long.100/.

Jingwei Zuo, Maksim Velikanov, Ilyas Chahed, Younes Belkada, Dhia Eddine Rhayem, Guillaume Kunsch, Hakim Hacid, Hamza Yous, Brahim Farhat, Ibrahim Khadraoui, Mugariya Farooq, Giulia Campesan, Ruxandra Cojocaru, Yasser Djilali, Shi Hu, Iheb Chaabane, Puneesh Khanna, Mohamed El Amine Seddik, Ngoc Dung Huynh, Phuc Le Khac, Leen AlQadi, Billel Mokeddem, Mohamed Chami, Abdalgader Abubaker, Mikhail Lubinets, Kacper Piskorski, and Slim Frikha. Falcon-h1: A family of hybrid-head language models redefining efficiency and performance. *arXiv preprint arXiv:2507.22448*, 2025.