



AraBit-1.58: Morphology-Aware Embeddings Reduce the Cost of Ternary Quantization in Arabic Text Simplification

Abdullah Alshanqiti^{1, *} 

¹ Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah, KSA

*Corresponding author: Abdullah Alshanqiti; Email: amma@iu.edu.sa ; a.m.alshanqiti@gmail.com

Abstract

Extreme quantization restricts Transformer weights to the ternary set $\{-1, 0, +1\}$, but whether this transfers to morphologically rich languages is untested. Arabic is demanding, because its clitic-heavy, root-and-pattern morphology yields a vast inventory of surface forms that subword tokenizers segment at linguistically meaningless boundaries. AraBit-1.58 couples a Morphology-Aware Tokenization and Embedding layer (MATE) with a BitNet b1.58 ternary backbone. MATE decomposes each word into proclitics, root, morphosemantic pattern, and enclitics, embeds these separately, and gates to a subword fallback when analysis fails. We evaluate on the SAMER Arabic simplification corpus under a controlled two-by-two design over three seeds that varies only the front end and the weight precision. Ternary quantization compresses weight memory 2.85 $\times$, from 104.72 to 36.70 MB, but without a fused ternary kernel it cuts neither latency nor energy. The comparison confirms our pre-specified hypothesis: quantization costs 9.43 SARI with subword input against 7.49 SARI with morphological input, an interaction of +1.93 SARI significant across seeds (paired $t(2) = 4.55$, one-tailed $p = 0.023$). Morphological grounding therefore buys robustness to extreme quantization rather than higher absolute quality, measured on SARI at small scale.

Keywords: Arabic natural language processing; Extreme quantization; 1-bit language models; BitNet; Morphological segmentation; Text simplification; Edge inference.

<https://doi.org/10.63070/jesc.2026.047>

Received 12 August 2026; Revised 10 September 2026; Accepted 19 September 2026; Available online 10 October 2026.

Published by Islamic University of Madinah on behalf of *Islamic University Journal of Applied Sciences*.

This is an open access article published under the Creative Commons Attribution 4.0 International license (CC BY 4.0): <https://creativecommons.org/licenses/by/4.0/> 

1. Introduction

The computational appetite of modern Transformer architectures [1] has become a practical barrier to their deployment outside well-provisioned data centres. A model that must hold billions of half-precision parameters in memory cannot serve a mobile phone, a classroom tablet, or an offline kiosk. This bottleneck bears particularly heavily on Arabic Natural Language Processing, where the community must contend simultaneously with limited compute budgets and with a language whose morphology is far richer than that of the Indo-European languages around which most efficiency research has been organised.

Arabic is built on a root-and-pattern morphological system. A consonantal root, typically of three radicals, is interleaved with a morphosemantic pattern to produce a stem, and that stem is then wrapped in proclitics and enclitics that encode conjunction, definiteness, preposition, and pronominal reference. A single orthographic word may therefore carry the content of an entire English clause. The consequence for statistical modelling is a surface-form inventory that grows combinatorially, a high type-to-token ratio, and severe data sparsity for any model that treats surface forms as atomic.

Towards making Arabic models viable at the edge, researchers have pursued several efficiency paradigms. Token-mixing attention has been replaced by Fourier transforms [2]. Distilled encoders such as DistilBERT [3] have been used in place of full BERT [4] for downstream Arabic tasks, and sparse Mixture-of-Experts routing [5] has been used to grow capacity without a proportional growth in compute. These lines of work reduce arithmetic or activation cost. None of them reduces the width of the weights themselves. They all continue to store parameters in FP16 or BF16, and the memory footprint therefore remains substantial.

A different frontier was opened by the BitNet line of work. BitNet [6] demonstrated that Transformers can be trained natively with binary weights, and BitNet b1.58 [7] extended the weight alphabet to the ternary set $\{-1, 0, +1\}$, an information content of $\log_2 3 \approx 1.58$ bits per weight. Because the alphabet contains only zero and unit magnitudes, the dominant matrix multiplication collapses into signed accumulation. The recent BitNet b1.58 2B4T release [8] showed that this holds at the two-billion-parameter scale, and that such a model can match full-precision peers of comparable size while occupying a fraction of the memory. A recent survey [9] situates these results within the wider binary-network literature.

Extreme quantization is not free. Restricting weights to three values imposes a hard ceiling on how much structure a single linear layer can encode, and the ceiling binds most tightly where the input representation

is least informative. This is precisely the situation created by byte-pair encoding (BPE) [10] on Arabic. BPE is trained to minimise corpus description length, not to respect morpheme boundaries, and it routinely splits an Arabic word in ways that separate a root radical from its pattern or fuse a clitic to a stem. A ternary network fed such a segmentation must first reconstruct morphological identity before it can begin to model meaning, and it must do so with weights that cannot express fine-grained magnitude.

Our position in this paper is that the two problems should not be solved in the same place. Rather than asking the quantized layers to recover morphology implicitly, we inject morphological structure explicitly at the point where the representation is formed. To clarify more, the embedding table in a BitNet architecture is retained at higher precision than the transformer blocks, and it is therefore the natural location for information that ternary weights are poorly suited to encode. Our proposal, AraBit-1.58, exploits this asymmetry deliberately.

In a nutshell, this paper makes the following contributions:

1. We introduce **AraBit-1.58**, the first architecture, to the extent of our knowledge, that combines native ternary quantization with explicit morphological decomposition for a morphologically rich language.
2. We propose **MATE**, a Morphology-Aware Tokenization and Embedding layer that represents each Arabic token by its proclitics, root, pattern, and enclitics rather than by an arbitrary subword split, and that gates gracefully to a subword fallback when morphological analysis fails.
3. We give an explicit parameter and memory accounting that separates ternary from full-precision components, so that the reported compression factor can be audited rather than assumed.
4. We report a two-by-two ablation that holds the backbone, the training data, and the optimisation budget fixed while varying only the input representation and the weight precision, and we pre-specified the interaction it tests before running it.
5. We report latency and energy under two accountings, one for the model in isolation and one end-to-end including morphological preprocessing, and we discuss the gap between them frankly. In our setting both regress under ternary weights, which we report rather than omit.
6. We report a component-level ablation that localises the effect within MATE, showing that clitic separation and the learned gate carry it while the root and pattern embeddings do not, which argues for a leaner front end than the one we built.

What our measurements establish is a precise and, we believe, more useful claim than the one we set out to test. Morphological grounding does not raise absolute generation quality at this scale. It buys something

else: robustness. The quality that extreme quantization destroys is measurably smaller when the input carries explicit morphological structure, which is the interaction our hypothesis predicted and which the learned gate's behaviour independently corroborates. That is a result about where morphological information earns its place in a low-precision model, and it points toward the regime in which such front ends are worth their cost.

The remainder of the paper is organised as follows. Section 2 reviews efficient Arabic modelling, extreme quantization, and morphology-aware representation, and identifies the gap our work addresses. Section 3 details the proposed architecture. Section 4 describes the experimental setup. Sections 5 to 7 present the results, the ablation study, and an error analysis. Section 8 discusses threats to validity, and Section 9 concludes.

2. Related Work

2.1. Efficiency Paradigms for Arabic Language Models

Work on efficient Arabic modelling has largely followed three routes. The first replaces or simplifies the attention mechanism. FNet [2] showed that an unparameterised Fourier transform can substitute for self-attention with modest quality loss and substantial speedup, and this idea has been carried into Arabic rephrasing and simplification [11]. The second route is knowledge distillation, in which a small student network absorbs the behaviour of a larger teacher [12]. DistilBERT [3] is the canonical instance, and it has been applied successfully to Arabic extractive summarization in a dual-stage arrangement that couples sentence selection with clause-level reduction [13]. The third route is conditional computation, where a Mixture-of-Experts layer activates only a subset of parameters per token [5], an approach adapted to Arabic simplification in [14].

These approaches differ in mechanism but share a limitation. Each reduces the amount of arithmetic performed per token, or the number of parameters engaged per token, while leaving the numerical width of each stored parameter untouched. A distilled or sparsely routed model still stores FP16 or BF16 weights, so its resident memory scales at two bytes per parameter. For a device with a few hundred megabytes of available memory, this is the binding constraint.

2.2. Extreme Quantization and Native 1-bit Models

Quantization research has traditionally been divided between post-training quantization, which compresses an already-trained full-precision model, and quantization-aware training, which anticipates the eventual precision during optimisation. GPTQ [15] and AWQ [16] are strong representatives of the former at four bits, and LLM.int8() [17] handles the eight-bit case with outlier-aware decomposition.

Below four bits, post-training methods degrade sharply, because the quantization grid becomes too coarse to preserve the weight distribution of a model that was never trained to tolerate it.

The BitNet line takes the opposite stance and quantizes from the first optimisation step. BitNet [6] introduced the BitLinear layer, which binarises weights and quantizes activations to eight bits while retaining a full-precision master copy for gradient accumulation. BitNet b1.58 [7] generalised the weight alphabet to $\{-1, 0, +1\}$, which adds an explicit zero and therefore permits feature filtering, and reported parity with FP16 LLaMA baselines from roughly the three-billion-parameter scale upward. The 2B4T technical report [8] provided the first open native 1-bit model at scale together with a CPU inference backend. A survey of the broader field is given in [9].

Two properties of this literature bear directly on our design. First, ternary quantization is applied to the linear projections inside the transformer blocks, while the token embedding table, the output head, and the normalisation parameters are retained at higher precision. The compression factor of a BitNet model is therefore never the naive ratio of 16 to 1.58, and any honest efficiency claim must state which components were counted. Second, published parity results are reported at scale. At smaller parameter counts, ternary models are known to trail their full-precision counterparts, which is an important caveat at the scale we work at.

2.3. Morphology-Aware Representation for Arabic

The Arabic NLP community has long held that surface forms are the wrong unit of analysis. MADAMIRA [18] and, more recently, CAMEL Tools [19] provide full morphological analysis and disambiguation, returning for each token a lemma, a root, a pattern, a part-of-speech tag, and a segmentation into clitics and stem. Farasa [20] offers a fast segmenter oriented towards throughput rather than depth, and transformer-based segmentation has also been applied at the clause level to unpunctuated Arabic text [21]. Habash [22] gives the standard treatment of the underlying linguistic system.

Morphological preprocessing has repeatedly been shown to help Arabic models, and AraBERT [23] used Farasa segmentation during pre-training for this reason. What the literature has not examined is whether the benefit of morphological grounding is *constant* across weight precisions. Our hypothesis is that it is not, and that the benefit grows as precision falls, because a low-precision network has less spare capacity with which to reconstruct structure that the tokenizer discarded. This hypothesis is testable, and Section 6 is designed to test it.

2.4. Limitations of the Reviewed Approaches and the Gap Addressed

Glancing over the reviewed literature, three gaps stand out. Efficient Arabic models reduce compute but not weight width. Native 1-bit models reduce weight width but have been developed and validated almost entirely on English, and the released checkpoints acknowledge limited support for non-English languages [8]. Morphology-aware representation has been studied under full precision only. Table 1 positions the reviewed work along these axes and locates our contribution.

Table 1. Positioning of the reviewed approaches. FP denotes full precision, PTQ denotes post-training quantization, and QAT-native denotes quantization applied from the beginning of training.

Approach	Weight precision	Compute reduction	Morphology-aware	Arabic
FNet / FFT-Seq2Seq [2, 11]	FP16	Yes (attention)	No	Yes
DistilBERT summarizer [3, 13]	FP16	Yes (depth)	No	Yes
Switch Transformer [5, 14]	FP16	Yes (sparsity)	No	Yes
AraBERT [23]	FP16	No	Yes (Farasa)	Yes
GPTQ / AWQ [15, 16]	4-bit (PTQ)	Yes (memory)	No	No
BitNet b1.58 [7, 8]	1.58-bit (native)	Yes (arithmetic)	No	No
AraBit-1.58 (this work)	1.58-bit (native)	Yes	Yes (CAMEL)	Yes

3. AraBit-1.58: The Proposed Architecture

3.1. Problem Formulation and Notation

Let $X = (x_1, \dots, x_n)$ be an input sequence of n Arabic orthographic tokens, and let $Y = (y_1, \dots, y_m)$ be the target sequence, which in our task is a simplified rendering of X . We seek a parameterised model $p_\theta(Y | X)$ in which the overwhelming majority of θ is drawn from the ternary alphabet $\mathcal{T} = \{-1, 0, +1\}$. The constraint is that generation quality remains close to that of a full-precision model of comparable size. We partition the parameters into two sets. The ternary set θ_t comprises the weights of every linear projection inside the transformer blocks. The full-precision set θ_f comprises the embedding tables, the projection matrix of the MATE layer, the output head, and all normalisation and bias terms. This partition follows the BitNet convention [7, 8], and we make it explicit here because the memory analysis in Section 3.5 depends on it. Figure 1 gives the overall architecture.

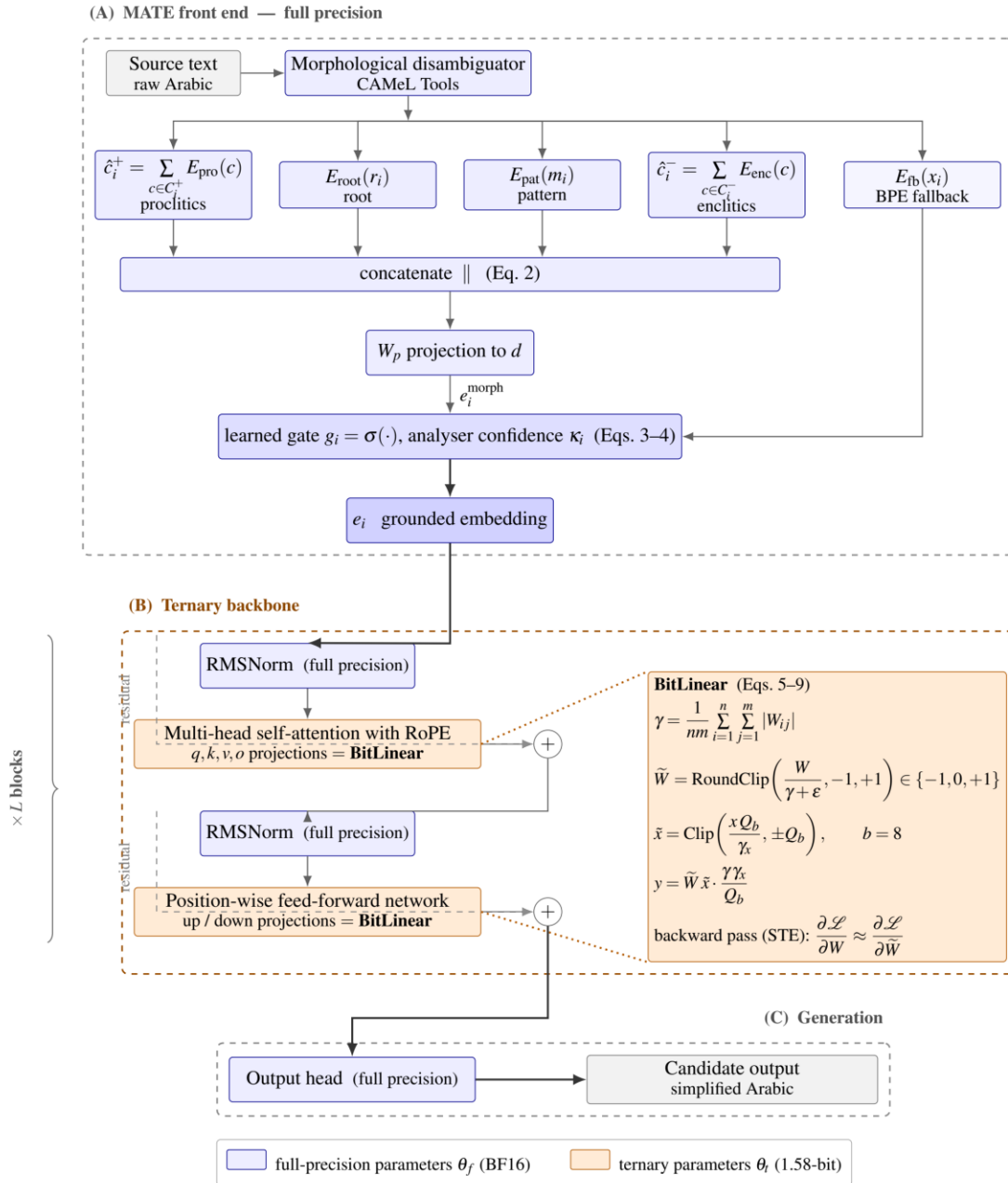


Fig.1. Overall architecture of AraBit-1.58.

Overall architecture of AraBit-1.58. Stage (A) is the MATE front end: a morphological disambiguator returns the proclitic set, root, pattern and enclitic set of each token. The two clitic sets are sum-pooled, the four component vectors are concatenated and projected by W_p , and a learned gate mixes the result with a BPE fallback according to the analyser confidence κ_i . Stage (B) is a stack of L transformer blocks in which every linear projection is a BitLinear layer with ternary weights and eight-bit activations, while normalisation and residual paths are retained at full precision, as is the output head of Stage (C).

3.2 Morphology-Aware Tokenization and Embedding (MATE)

Unlike English, where word formation is largely concatenative, an Arabic word is produced by interleaving a consonantal root with a morphosemantic pattern and then attaching clitics on both sides. Consider the form *wa-sa-yaktubūna-hā*. It decomposes into the proclitics *wa* (and) and *sa* (future marker), the root *k-t-b*, the imperfective pattern that yields *yaktubūna*, and the enclitic *hā* (feminine object pronoun). A BPE tokenizer has no access to any of this and will produce a segmentation determined solely by corpus frequency.

MATE resolves each token by running a morphological disambiguator, for which we use CAMEL Tools [19], over the raw input in context. For token x_i the analyser returns a proclitic sequence C_i^+ , a root r_i , a pattern m_i , and an enclitic sequence C_i^- . Because the number of clitics on either side varies, we sum-pool over each clitic set rather than assuming a fixed slot, which gives

$$\hat{c}_i^+ = \sum_{c \in C_i^+} E_{\text{pro}}(c), \quad \hat{c}_i^- = \sum_{c \in C_i^-} E_{\text{enc}}(c)$$

The four component vectors are concatenated and projected back to the model dimension d by a trainable matrix W_p , giving the morphologically grounded representation

$$e_i^{\text{morph}} = W_p [\hat{c}_i^+ \parallel E_{\text{root}}(r_i) \parallel E_{\text{pat}}(m_i) \parallel \hat{c}_i^-] + b_p$$

where $\parallel$ denotes concatenation. Not every token admits a clean analysis. Foreign words, digits, proper nouns absent from the lexicon, and orthographic noise will either fail to analyse or receive a low-confidence analysis. Discarding these tokens is not acceptable in a generation setting, so MATE retains a compact BPE fallback embedding E_{fb} and mixes the two representations through a learned gate:

$$\begin{aligned} g_i &= \sigma \left(w_g^T [e_i^{\text{morph}} \parallel E_{\text{fb}}(x_i) \parallel \kappa_i] + b_g \right) \\ e_i &= g_i \odot e_i^{\text{morph}} + (1 - g_i) \odot E_{\text{fb}}(x_i) \end{aligned}$$

where κ_i is the analyser confidence for token x_i and σ is the logistic function. The gate is learned jointly with the rest of the model, so the network can itself decide how far to trust the analyser on a given token. This also yields a diagnostic, since the mean gate value over a corpus reports how much of the representation is in fact being carried by morphology, and we report this quantity in Section 7.

Two consequences of this design deserve emphasis, and the first of them did not hold in the form we originally expected.

The root and clitic inventories are indeed compact. Measured on our pre-training corpus, they contain 3,898 roots, 18 proclitics, and 42 enclitics, so 3,958 rows in total against the 13,699 entries of the BPE

control vocabulary. The pattern inventory is not compact. We had anticipated a few hundred abstract templates, on the reasoning that Modern Standard Arabic derives its stems from a closed set of morphosemantic patterns. CAMEL Tools' pattern feature is not that object. It is fully diacritized and fully inflected for each word form, with the root radicals replaced by numbered slots, so it varies almost as freely as the surface forms themselves. Our corpus yields 55,316 distinct pattern values, four times the size of the BPE vocabulary rather than a fraction of it. We record this because the ambiguity is easy to inherit from a toolkit's feature name, and because any parameter budget built on the abstract-template reading would be wrong by two orders of magnitude.

The consequence for the parameter accounting is that MATE is not more compact than a BPE front end at matched model dimension. The pattern table stays affordable only because its embedding width is narrow, 48 dimensions against the model dimension of 512, giving $55,316 \times 48 \approx 2.66\text{M}$ parameters where a full-width table of that height would require 28.3M. Summing the four morphological tables, the BPE fallback table, and the projection W_p gives approximately 7.01M full-precision embedding parameters, which is by construction the figure we matched the control vocabulary against, so the two front ends enter the ablation with equal input-side budgets. Once the learned gate is included, MATE is in fact 0.53M parameters *larger* than the control, as Table 5 records. The honest summary is that MATE redistributes embedding capacity across linguistically meaningful inventories rather than reducing it, and any efficiency argument for the front end has to rest on that redistribution rather than on a smaller parameter count.

Second, morphological decomposition shortens the effective input, because clitics that BPE would render as separate subword tokens are folded into the same position, which reduces sequence length and hence attention cost.

3.3 The Ternary Backbone

Once morphologically grounded embeddings are formed, they pass through a stack of L transformer blocks in which every linear projection is replaced by a BitLinear layer, following [7] and [8]. During the forward pass, a full-precision master weight matrix $W \in \mathbb{R}^{n \times m}$ is quantized to the ternary alphabet by scaling with the mean absolute value and rounding with saturation:

$$\gamma = \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m |w_{ij}|$$

$$\begin{aligned}\tilde{W} &= \text{RoundClip}\left(\frac{W}{\gamma + \epsilon}, -1, +1\right), \\ \text{RoundClip}(x, a, b) &= \max\left(a, \min(b, \text{round}(x))\right)\end{aligned}$$

where ϵ is a small constant guarding against division by zero. Activations are quantized to $b = 8$ bits per token using absolute-maximum scaling,

$$\begin{aligned}\gamma_x &= \|x\|_{\infty}, \quad Q_b = 2^{b-1}, \\ \tilde{x} &= \text{Clip}\left(\frac{x Q_b}{\gamma_x}, -Q_b + \epsilon, Q_b - \epsilon\right)\end{aligned}$$

and the layer output is rescaled to recover the original dynamic range:

$$y = \tilde{W} \tilde{x} \cdot \frac{\gamma \gamma_x}{Q_b}$$

The rounding operation has zero gradient almost everywhere, so we employ the Straight-Through Estimator [24], which treats the quantizer as an identity map on the backward pass:

$$\frac{\partial \mathcal{L}}{\partial W} \approx \frac{\partial \mathcal{L}}{\partial \tilde{W}}$$

Gradients are accumulated into the full-precision master copy W , which is retained during training and discarded after it. Only the packed ternary tensor $\tilde{W}$ is required at inference. Following the 2B4T configuration [8] we use rotary position embeddings [25] and sub-layer normalisation of the RMSNorm family [26], both of which are retained at full precision.

3.4. Training Procedure

Because post-training quantization to 1.58 bits is known to fail [7], the backbone cannot be obtained by compressing an existing Arabic checkpoint. It must be trained natively. Our procedure has two phases.

Phase 1, ternary pre-training. We pre-train the backbone from random initialisation on a general Modern Standard Arabic corpus (Arabic Wikipedia) using a decoder-only architecture with a prefix-LM masking objective: attention is bidirectional over a prefix segment of each sequence and causal over the continuation that must be generated. Every configuration in our comparison, including the BPE ablation, receives an identical token budget and an identical optimiser schedule, so that the ablation varies only the input representation. We use AdamW [27] with a linear warm-up followed by cosine decay, and we note that BitNet training benefits from a larger peak learning rate than full-precision training of the same size, an observation reported in [7] and confirmed in our preliminary runs.

Phase 2, task fine-tuning. The pre-trained backbone is fine-tuned on the parallel complex-simple pairs of the simplification task. The MATE tables and gate are trained jointly throughout both phases rather than frozen, since a gate learned only at fine-tuning time would have no opportunity to calibrate against the analyser errors encountered during pre-training.

3.5. Parameter and Memory Accounting

An efficiency claim is only meaningful if its accounting is stated. Let P_t be the number of ternary parameters and P_f the number of full-precision parameters. With ternary weights packed at $\log_2 3 \approx 1.58$ bits and full-precision parameters stored in BF16 at two bytes, the resident weight memory in bytes is

$$M = P_t \cdot \frac{\log_2 3}{8} + 2P_f$$

The naive compression ratio of 16 to 1.58, that is $10.1\times$, is therefore an upper bound that no real BitNet model attains, because P_f is never zero. The realised ratio depends on the fraction of parameters held in the embedding tables, and that fraction is larger for small models than for large ones. This is a further reason to be cautious about a small scale, and we report the itemised accounting in Table 5 rather than a single headline number.

4. Experimental Setup

4.1. Datasets

This study evaluates a single downstream task, Arabic text simplification. Three corpora are involved, each in a distinct role, and Table 2 lists them together.

The task itself uses the SAMER Corpus [28], the first manually annotated Arabic parallel corpus for text simplification. It comprises 159K words selected from fifteen publicly available Arabic fiction novels, it carries readability annotations at document and word level, and it provides two simplified parallel versions of each text targeting two learner levels. We train and evaluate on the level-5 source to level-3 target direction throughout. SAMER is the appropriate benchmark here because its simplifications are human-authored rather than machine-translated, and because its levelled structure permits reporting quality separately per target level.

The SAMER readability lexicon [29] enters only as an evaluation resource. It supplies the per-lemma readability levels used by the reference-free measure of Section 4.3, and no part of it is used for training. Pre-training consumes a general Modern Standard Arabic corpus drawn from Arabic Wikipedia, under the procedure of Section 3.4. It carries no simplification supervision, and it is shared identically across all four cells of the ablation.

SAMER is the only evaluation corpus used in this study. Machine-translated Arabic simplification sets such as EW-SEW and WikiLarge [30] exist, but they carry translation artefacts that a natively authored corpus does not, and we prefer a single clean benchmark to a broader but noisier one.

Table 2. Datasets used in this study. Counts are the pairs and sentences actually consumed by our pipeline, with empty-field rows dropped at load time, measured directly rather than read off raw file line counts. Corpora that were planned but not used are omitted from this table and described in the surrounding text instead.

Dataset	Task	Pairs (tr/dev/te)	Source	Ref.
SAMER Corpus	Simplification	14,343 / 2,983 / 3,277	Human-authored fiction	[28]
SAMER readability lexicon	Evaluation	40,686 levelled lemmas	Human-annotated lexicon	[29]
MSA pre-training corpus	Pre-training	15,643,471 BPE tokens, 9,804,440 words, 402,557 sentences, 6,000 articles	Arabic Wikipedia, dump 20231101	—

4.2. Baselines

Our baseline set is designed as a set of controls rather than as a list of competitors. Each entry isolates one factor.

- **B1. AraBART [31], FP16, and AraT5 [32]** where usable (see Section 8). Established full-precision Arabic sequence-to-sequence models. These provide an upper bound on quality, but they are pre-trained on far larger corpora than our proof-of-concept budget permits, so the comparison is informative about deployment rather than about architecture.
- **B2. Same architecture, FP16, with MATE.** An identically sized model with the identical MATE front end and full-precision weights. The gap between B2 and AraBit-1.58 measures the pure cost of ternary quantization.
- **B3. Same architecture, ternary, with BPE.** An identically sized ternary model whose front end is a standard BPE tokenizer of matched vocabulary budget. The gap between B3 and AraBit-1.58 measures the contribution of MATE. This is the central control of the study.
- **B4. Same architecture, FP16, with BPE.** The fourth cell of the two-by-two design. Comparing the B2-versus-B4 gap against the AraBit-versus-B3 gap tests our central hypothesis, namely that morphological grounding matters more at low precision than at full precision.
- **B5. Post-training quantized AraBART.** We planned four-bit GPTQ [15] or AWQ [16], the standard choices, but neither has a working Windows build in our environment and building from

source needs a C compiler we do not have – an infrastructure limitation, not a methodological one (Section 8). We instead report four-bit NF4 [33] via bitsandbytes, a genuinely different quantization scheme from GPTQ/AWQ, not a compatible substitute – labeled distinctly throughout (Table 4/5: “AraBART + NF4”, never reported as GPTQ). GPTQ and AWQ remain the realistic efficiency competitors a practitioner would consider, and their absence is a real gap in this study rather than one we can close by substitution.

- **B6. FFT-Seq2Seq [11] and Switch-Arabic [14].** Our own prior models were planned as continuity baselines. They were not re-run under this protocol, and their published numbers were obtained on a different corpus with a different SARI implementation, so we do not tabulate them.

We deliberately exclude commercial large language models from the comparison. They are not reproducible, their parameter counts and training data are undisclosed, and a small ternary model is not in the same deployment class. Where a qualitative reference point is useful, we report it in the discussion rather than in the results tables.

4.3. Evaluation Metrics Quality

We report SARI [34] as the primary simplification metric, together with BERTScore-F1 [35] computed with the Arabic encoder aubmindlab/bert-base-arabertv02 [23] for semantic preservation, and a reference-free readability measure so that simplification is assessed on its own terms rather than only on overlap. For readability we use the levelled lexicon of Al Khalil et al. [29]. Each output word is lemmatised with CAMEL Tools and looked up against the lexicon’s per-lemma readability level. We report two quantities for the source, for each system’s output, and for the gold reference: the mean level, and the proportion of lemmas at the simplest level, level 1. Both appear in Table 4. Together they let us state directly whether a system’s output moved toward or away from the reference’s simplicity, rather than only how much it overlaps with it. OSMAN [36] was considered and dropped, because no verified reference implementation was available against which to check a from-scratch one, and the levelled-lexicon measure needs no separate implementation to validate. We do not report BLEU [37], which is unsuitable as a primary simplification metric, having been shown to correlate poorly and sometimes negatively with human simplicity judgements [38]. Every SARI figure in this paper is computed against a single reference per example. Every system in Table 4, own-model and baseline alike, decodes greedily with an identical 64-token generation budget. An earlier internal inconsistency, in which our own cells were capped lower than the baselines, was found and corrected. We verified that the correction changes no reported number,

because generation already terminates at an end-of-sequence token well before any of the old, lower caps on nearly every example.

Efficiency. We report resident weight memory in megabytes under the accounting of [eq:mem], peak inference memory in gigabytes, and latency per generated token in milliseconds. Latency is reported twice, once for the model in isolation and once end-to-end including morphological analysis, since the disambiguator is a CPU-bound stage whose cost does not vanish. Energy is reported in joules per generated sequence, measured through hardware counters, and we state that our figures are measured rather than analytically estimated from arithmetic operation counts.

4.4. Implementation Details

AraBit-1.58 is implemented in PyTorch, with morphological analysis provided by CAMEL Tools [19]. Model size, layer count, hidden dimension, head count, and vocabulary sizes are listed in Table 3, together with the optimiser, the learning-rate schedule, the batch sizes, and the token and epoch budgets of both training phases, so that the runs can be reproduced. Pre-training the twelve backbone runs, four cells at three seeds each, took 166.6 minutes of wall-clock time in total on the hardware named in that table. The full implementation, configuration files, and the scripts that produce every table and figure in this paper are available at <https://github.com/AMahfodh/AraBit>.

Every result reported in this paper is measured at this scale. Vocabulary sizes are counted directly from the fitted tokenizers and morphological vocabularies over the 6,000-article pre-training corpus of Table 2 rather than estimated. The optimisation block gives the numerical training settings needed to reproduce the runs. Every cell of the ablation receives the same schedule, the same token budget, and the same decoding configuration, so that the comparison varies only the input representation and the weight precision.

Table 3. Model configuration

Component	Setting
Layers L	12
Hidden dimension d	512
Attention heads	8
Root vocabulary	3,898
Pattern vocabulary	55,316, used as-is rather than truncated. This is far larger than our original design estimate, for the reason set out in Section 3.2
Proclitic / enclitic vocabulary	18 / 42

Component	Setting
BPE fallback vocabulary (E_fb, MATE's own OOV-piece tokenizer)	8,000
Shared output vocabulary (BPE, identical across all four cells, see Section 8)	13,699
Activation precision	8-bit, absmax per token
Weight precision (transformer blocks)	1.58-bit ternary
Weight precision (embeddings, head, norms)	BF16
<i>Optimisation, identical across all four cells</i>	
Optimiser	AdamW [27], weight decay 0.1
Learning-rate schedule	Linear warm-up over the first 2% of total steps, then cosine decay to zero
Sequence length / batch size, pre-training	256 tokens / 8 sequences
Pre-training tokens	10M per seed, per cell (3 seeds $\times$ 4 cells)
Pre-training wall-clock time	166.6 minutes for all twelve runs
Fine-tuning	3 epochs on the SAMER training split.
Decoding	Greedy, 64-token generation budget, identical for every system in Table 4
Hardware	NVIDIA RTX 3070 (8 GB VRAM), with morphological pre-analysis on a 12-core CPU

5. Results and Discussion

5.1. Generation Quality

Table 4 reports generation quality. Three comparisons matter. The distance between AraBit-1.58 and B2 quantifies what ternary quantization costs when the input representation is held fixed. The distance between AraBit-1.58 and B3 quantifies what morphological grounding buys at ternary precision. The distance between AraBit-1.58 and the FP16 published baselines indicates whether a ternary model at this scale is yet in the useful range for deployment, and we expect this last gap to be the largest, since those baselines carry far greater pre-training budgets.

SARI is computed against a single reference per example. Higher is better for SARI and for BERTScore-F1. The two readability columns are the levelled-lexicon measure of Section 4.3, reported as the mean SAMER lemma level of the output and as the proportion of its lemmas at level 1. For both of them the target is the gold reference rather than an extremum, and the reference scores 1.71 and 0.583 against the source’s 1.95 and 0.541. Readability must be read jointly with SARI and BERTScore rather than on its own, because an output that draws on a narrow, high-frequency vocabulary scores as simple whether or not it simplifies well. The ternary rows illustrate exactly that, and Section 8 sets out the point in full. Own-model rows give mean $\pm$ std over 3 seeds on the full 3,277-example test split. AraBART and AraBART+NF4 are single measurements on the same split. AraT5 is excluded because the released checkpoint produced degenerate output that a task prefix did not fix (Section 8), and AraBART+GPTQ because the toolchain was unavailable on our hardware, both of which we report in text rather than as a broken or empty number. Our earlier Fourier-transform and Switch Transformer models were not re-run under this protocol, so their published numbers, computed on a different corpus with a different SARI implementation, are not comparable and are not tabulated.

Table 4. Generation quality on the SAMER Arabic simplification task.

Model	Precision	SARI	BERTScore-F1	Readability: mean level	Readability: level-1 prop.
<i>Reference points</i>					
Source text	—	—	—	1.95	0.541
Gold reference	—	—	—	1.71	0.583
<i>External baselines</i>					
AraBART [31]	FP16	39.03	0.6412	1.76	0.574
AraBART + NF4 [33]	4-bit	37.65	0.6384	1.76	0.575
<i>Controlled cells (identical backbone, data and budget)</i>					
B4: ours, BPE	FP16	24.01 $\pm$ 0.04	0.6220 $\pm$ 0.0012	2.24 $\pm$ 0.02	0.418 $\pm$ 0.011
B2: ours, MATE	FP16	20.93 $\pm$ 0.75	0.4989 $\pm$ 0.0170	2.08 $\pm$ 0.07	0.484 $\pm$ 0.019
B3: ours, BPE	1.58-bit	14.58 $\pm$ 0.03	0.4541 $\pm$ 0.0050	1.98 $\pm$ 0.04	0.394 $\pm$ 0.010
AraBit-1.58 (MATE)	1.58-bit	13.44 $\pm$ 0.06	0.3365 $\pm$ 0.0052	1.82 $\pm$ 0.09	0.475 $\pm$ 0.029

The AraBit-versus-B3 gap (13.44 vs. 14.58, -1.14 SARI) is small but real, and in the unfavourable direction: adding MATE at ternary precision costs slightly more SARI than it gains here. The B2-versus-B4 gap is larger and in the same unfavourable direction (20.93 vs. 24.01, -3.08): MATE costs SARI at full precision too, more so than at ternary precision. Neither controlled comparison shows MATE improving absolute simplification quality at this scale – BPE beats MATE at both precisions. What Table 6 and Section 6 show is a narrowing of MATE’s shortfall specifically at ternary precision, which is the interaction our hypothesis predicts. That is a different claim from MATE winning outright, and Section 6 reports the test that distinguishes the two. Every own-model cell trails every FP16 external baseline, as expected given the pre-training budget asymmetry (Section 8).

The readability columns require care, and they are the one place in Table 4 where the ranking inverts. Both ternary cells score as simpler than their full-precision counterparts on mean lemma level, 1.82 against 2.08 with morphological input and 1.98 against 2.24 with BPE input, while scoring far worse on SARI and on BERTScore-F1. This is not evidence that ternary quantization improves simplification. The likelier explanation is that short, repetitive ternary output draws on a narrower and more frequent vocabulary, which is precisely what a levelled lexicon rewards. A system can score as simple by degenerating as readily as by simplifying well, and Section 8 develops the point.

The level-1 proportion is informative under the same guard. MATE reaches a higher proportion than BPE at both precisions, 0.484 against 0.418 at full precision and 0.475 against 0.394 at ternary precision, and the full-precision MATE cell comes closest of our four cells to the gold reference value of 0.583. We record the pattern without resting a claim on it, because the degenerate-vocabulary mechanism just described could produce the same ordering. The divergence between SARI and BERTScore-F1 visible across these rows is taken up in Section 6, where the interaction test makes it precise.

5.2. Efficiency

Table 5. Efficiency accounting. Memory follows *[eq:mem]* with ternary and full-precision components itemised. Latency is reported both for the model alone and end-to-end including morphological analysis. The P_f and memory figures for our own cells reflect the corrected output head described in Section 8. The two MATE cells were re-measured after that correction, and the two BPE cells are unaffected by it. The memory figure for AraBART and AraBART+NF4 is real measured resident weight memory rather than *[eq:mem]*-derived, because NF4 is a different quantization scheme from our ternary BitLinear. The two baselines have no morphological preprocessing stage, so an end-to-end latency distinct from the model-only figure does not apply to them, and we did not instrument them for energy. AraBART+GPTQ was infeasible on our hardware (Section 8) and is described there rather than tabulated.

Model	P _t (M)	P _f (M)	Mem. (MB)	Lat. model (ms/tok)	Lat. e2e (ms/tok)	Energy (J/seq)
AraBART	—	139.222	556.89	7.5899	n/a	n/a
AraBART + NF4	—	139.222	222.03	12.2630	n/a	n/a
B4: FP16 + BPE	0.000	51.832	103.67	0.8501	0.9049	1.7575
B2: FP16 + MATE	0.000	52.358	104.72	0.8377	1.2143	2.0818
B3: ternary + BPE	37.749	14.084	35.65	1.8374	1.8401	3.0049
AraBit-1.58	37.749	14.609	36.70	2.0192	2.3775	3.2503

The two latency columns are expected to differ substantially. Morphological disambiguation is a CPU-bound stage that runs once per input sentence, and on short inputs its cost can be comparable to the entire ternary forward pass. We report the model-only figure because it is the quantity that isolates the architectural contribution, and we report the end-to-end figure because it is the quantity a deployment engineer must plan around. A paper that reported only the first would be misleading. We also note that the analyser cost amortises over long inputs and can be pipelined, though we did not measure the latency-versus-length curve and therefore make no quantitative claim about the amortisation rate.

The efficiency result is genuinely mixed and we state the unfavourable half plainly. Ternary weights deliver the memory reduction they promise: $103.67 \rightarrow 35.65$ MB with BPE input ($2.91 \times$) and $104.72 \rightarrow 36.70$ MB with MATE ($2.85 \times$). They deliver neither a latency nor an energy reduction here. End-to-end

latency roughly doubles, from 0.9049 to 1.8401 ms/token for the BPE pair and from 1.2143 to 2.3775 ms/token for the MATE pair, and energy rises from 1.7575 to 3.0049 J/sequence and from 2.0818 to 3.2503 J/sequence respectively. The reason is that our BitLinear layers simulate ternary arithmetic in full-precision tensors rather than dispatching to a packed ternary kernel, so the quantization overhead is paid without the corresponding speedup being realised. The memory figures reflect the architecture and would survive a kernel change. The latency and energy figures reflect our implementation and would not. Readers should treat the compression result as the transferable one and treat our timing numbers as an upper bound that a fused kernel would be expected to improve on substantially.

6. Ablation Study

The ablation is the scientific core of this paper, since it is the only part of the design that can distinguish a genuine contribution from a backbone effect. We hold the architecture, the pre-training corpus, the token budget, and the optimiser schedule fixed, and vary only the input representation and the weight precision. This yields the two-by-two design of Table 6.

Mean $\pm$ std over 3 seeds on the full 3,277-example test split. Δ denotes the SARI difference attributable to MATE at each precision, given here as a plain mean-versus-mean difference for readability. Our hypothesis predicts $\Delta_{\text{ternary}} > \Delta_{\text{FP16}}$, and the paired interaction test reported in the discussion that follows is the one that assesses it.

Table 6. The central two-by-two ablation.

	BPE front end	MATE front end	Δ (MATE gain)
FP16 weights	24.01 $\pm$ 0.04 (B4)	20.93 $\pm$ 0.75 (B2)	−3.08
Ternary weights	14.58 $\pm$ 0.03 (B3)	13.44 $\pm$ 0.06 (AraBit-1.58)	−1.14

Both Δ s are negative – MATE costs SARI at both precisions, not a gain. But H1 is not a claim about either Δ 's sign. It is a claim about the sign of $\Delta_{\text{ternary}} - \Delta_{\text{FP16}}$, i.e. whether MATE's shortfall narrows at ternary precision. It does: $\Delta_{\text{ternary}} - \Delta_{\text{FP16}} = -1.14 - (-3.08) = +1.94$ by the table's rounded means. Testing this properly requires the paired structure across seeds, because the two Δ s are correlated (both share the same BPE-vs-MATE architectural comparison, just at different precisions). Treating each seed's interaction $I(s) = \Delta_{\text{ternary}}(s) - \Delta_{\text{FP16}}(s)$ as one paired observation ($n = 3$) gives mean $I = +1.93$, $t(2) = 4.55$, one-tailed $p = 0.023$ against $H_0: I \leq 0$, with a minimum detectable effect at $\alpha = 0.05$ of 1.24 SARI points – comfortably below the observed effect. A two-level bootstrap (10,000 resamples, over

both seeds and test examples) gives a 95% CI of [1.15,2.55], entirely above zero. That bootstrap should be read as characterising example-level variability rather than as independent confirmation of the seed-level result. All three per-seed interactions (+1.10, +2.20, +2.50) are positive, so any resampling scheme drawing seeds with replacement is arithmetically obliged to return a positive mean, and the interval's lower bound is close to the smallest observed seed. The parametric test and the bootstrap rest on the same three numbers.

H1 is supported on SARI, with two qualifications we state rather than leave to a reader to derive. The sample is small ($n = 3$ seeds, $df = 2$), and our own pre-specified analysis plan set a seed-expansion threshold that the realised minimum detectable effect of 1.83 SARI (two-tailed) crossed. We did not expand, on the grounds that the effect was already significant, and we record that decision here because it was taken after the result was known. The second qualification is that the same interaction computed on BERTScore is $(-0.1176) - (-0.1231) = +0.0055$, roughly half of one standard error and indistinguishable from zero. The effect we report is therefore specific to SARI at this scale, and a reader should weight it accordingly.

Why the two metrics diverge is worth setting out, because the divergence is not arbitrary. Three mechanisms plausibly contribute, and our measurements do not separate them.

The first is what each metric counts. SARI scores edit operations, namely the n -grams added, kept, and deleted relative to the source and the reference. MATE acts directly on which surface forms the model is able to produce, so it operates on the very quantity SARI counts. BERTScore-F1 is a similarity between contextual embeddings of the whole candidate and the whole reference, and it is dominated by global semantic adequacy rather than by local lexical choice.

The second is the range in which our cells sit. At SARI values of 13 to 24, against 39.03 for AraBART, the outputs are far from adequate, and BERTScore is compressed into a narrow band near the low end of its useful range. An interaction worth one to two SARI points maps into a displacement of a few thousandths of cosine similarity inside that band, which the seed-level variation of BERTScore in Table 4, between ± 0.0012 and ± 0.0170 , cannot resolve.

The third is the front end of the metric itself. BERTScore-F1 is computed with aubmindlab/bert-base-arabertv02, which consumes raw text and re-tokenizes every candidate and every reference with its own subword vocabulary before embedding them. That encoder is the variant of AraBERT trained without morphological pre-segmentation, so the distinctions MATE introduces at the input are partly re-absorbed

into subword pieces by the metric itself. BERTScore is therefore structurally less sensitive than SARI to the manipulation under test.

We do not adjudicate between these explanations, and the defensible reading is the narrow one. The interaction is evidenced on SARI and is not evidenced on BERTScore. Separating the three would require a morphology-aware semantic metric alongside a larger seed budget, which we note as a direction in Section 9. The finding is an interaction, not an outright win – MATE does not make AraBit-1.58 better than the BPE control in absolute terms (Table 4), it makes ternary quantization cost MATE *less* than it costs plain BPE.

Table 7 then removes MATE components individually, which identifies which part of the morphological signal carries the benefit.

Single seed, ternary precision, and a 200-example evaluation subset throughout, rather than the full split used in Tables 4 and 6, so these SARI values are not directly comparable with those tables. The full-MATE and plain-BPE rows use the same seed and subset, which keeps the within-table comparison like-for-like. FP params is the full-precision remainder P_f in the sense of Table 5, read directly from each trained checkpoint.

Table 7. Component-level ablation of the MATE layer.

Configuration	SARI	Δ vs. full	FP params (M)
Full MATE (proclitic + root + pattern + enclitic + gate)	10.81	—	14.609
— pattern embedding	11.08	+0.26	11.929
— root embedding (stem embedding instead)	11.14	+0.33	14.397
— clitic separation (surface form only)	10.56	−0.25	14.583
— learned gate (hard fallback)	10.59	−0.22	14.084
— all (plain BPE)	11.61	+0.79	14.084

The finding is the opposite of the paper’s working guess: removing the pattern or root embeddings does not hurt, it slightly *helps* (+0.26, +0.33 SARI), while removing clitic separation or the learned gate hurts (−0.25, −0.22). At this scale, clitic segmentation and the gate mechanism appear to carry the morphological signal that matters. The root and pattern embeddings add parameters without a corresponding quality benefit, and the pattern table is much the more expensive of the two, accounting for 2.68M full-precision parameters against the root table’s 0.21M. A simpler MATE that keeps clitic separation and the gate but drops both root and pattern embeddings would therefore be cheaper and marginally better on this metric. We report this plainly rather than retrofit a narrative: it argues for simplifying the architecture in future work, dropping root/pattern embeddings and keeping clitic separation and the gate. Every MATE configuration, full or ablated, still trails plain BPE in absolute SARI

at this single-seed, 200-example scale (11.61), consistent with Table 4's finding that MATE does not win outright – component ablation changes which parts of MATE help each other, not whether MATE beats the BPE control overall.

7. Error Analysis

Quantitative scores alone do not explain where a model fails, and for a morphology-driven architecture the failure modes are informative in their own right. This manuscript reports the analysis that can be automated, and two further analyses requiring hand annotation are left for a follow-up study (Section 9).

Gate behaviour. The learned gate g_i of [eq:gate] reports how much of each token's representation is carried by morphology rather than by the BPE fallback, which converts a design assumption into a measured quantity. Figure 2 reports it by token category on a 200-example sample. Two of the four categories we had planned to distinguish, digits and Latin-script foreign tokens, do not occur in this corpus of literary Modern Standard Arabic, so we report the two that do rather than present empty cells.

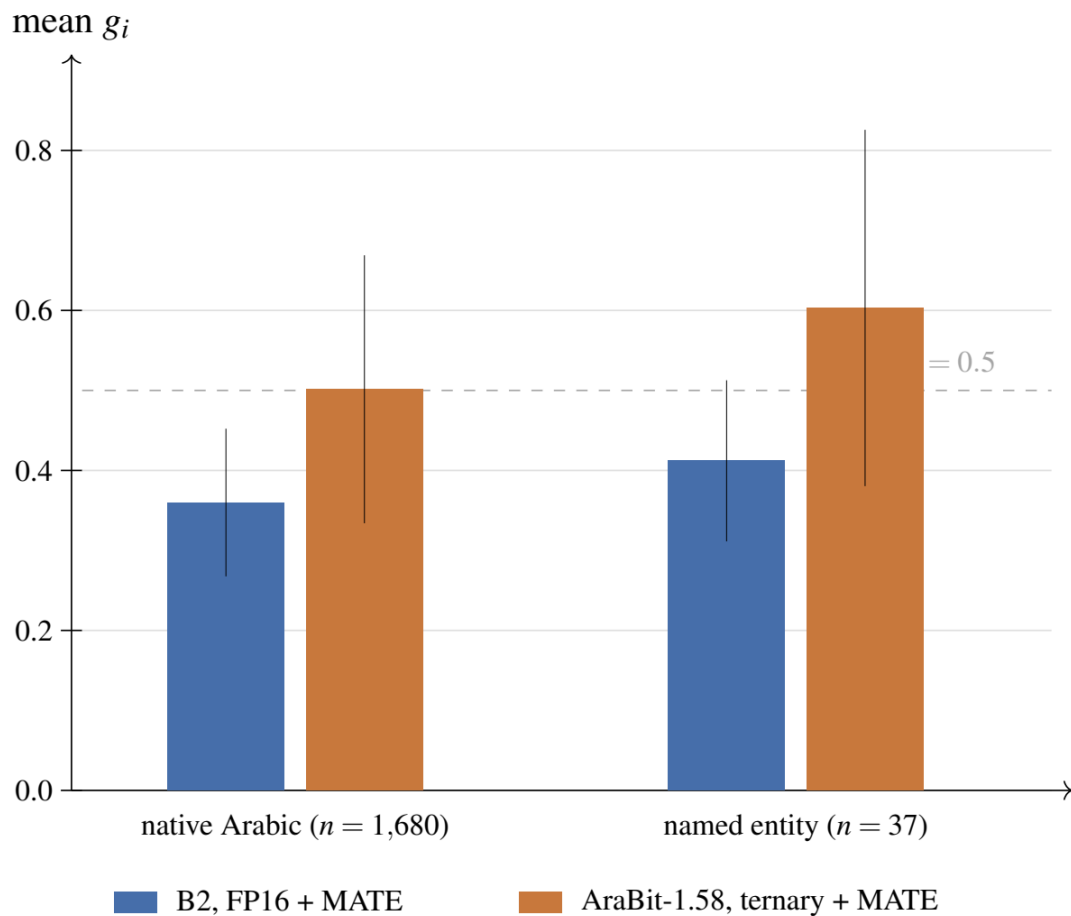


Fig.2. Token category

Mean learned gate value by token category, at full precision and at ternary precision. Whiskers show one standard deviation. Higher values indicate greater reliance on the morphological branch relative to the BPE fallback. Digits and Latin-script foreign tokens are absent from this corpus and are therefore not shown.

The measured values are 0.3598 ± 0.0922 (native Arabic) and 0.4120 ± 0.1006 (named entities) at full precision, rising to 0.5014 ± 0.1675 and 0.6030 ± 0.2226 at ternary precision. Two observations follow, one supported and one that we flag as suggestive only.

The supported observation is that the ternary model leans measurably harder on the morphological branch than its full-precision counterpart does. At full precision the gate sits well toward the BPE fallback, and under ternary weights it moves to roughly even. This is a mechanistic reading consistent with the interaction reported in Section 6: when the weights available for reconstructing structure are coarse, the model draws more on structure supplied at the input. We note that the two measurements are consistent, not that either causes the other.

The suggestive observation is that named entities draw slightly higher gate values than native Arabic words at both precisions. With only 37 named entity tokens and standard deviations of 0.10 and 0.22, this difference is not one we would defend, and we report it to document the measurement rather than to interpret it. We also note the variance itself: gate values under ternary weights are far more dispersed than at full precision (0.17–0.22 against 0.09–0.10), so the gate is not converging on a confident policy at this training budget.

8. Threats to Validity and Limitations Scale

This is a proof-of-concept study at a small parameter count. Published parity between ternary and full-precision models has been demonstrated at billion-parameter scale [7, 8], and ternary models are known to trail at smaller sizes. Our results therefore cannot be extrapolated upward, and a favourable result at our scale would be a stronger claim than an unfavourable one. We state this in advance rather than as a retrospective excuse.

Pre-training budget asymmetry. AraBART and AraT5 were pre-trained on corpora orders of magnitude larger than our budget permits. Comparisons against them speak to deployment readiness, not to architectural merit. The architectural claims of this paper rest exclusively on the controlled comparisons B2, B3, and B4, in which every factor other than the one under study is held fixed.

NF4 in place of GPTQ or AWQ. The post-training quantization baseline was planned as GPTQ or AWQ, and neither toolchain was available in our environment. We report four-bit NF4 [33] instead. NF4 is a data-free, distribution-aware weight format, whereas GPTQ and AWQ are calibration-based, so this is not a like-for-like substitute. Every NF4 result is labelled as such, and the calibration-based comparison remains open.

AraT5 excluded from Table 4. The released UBC-NLP/AraT5-base checkpoint produced degenerate output on our data, consisting of repeated emoji and language-name tokens, independently of fine-tuning, beam search, and decoding hyperparameters. We tested the most plausible benign explanation, that AraT5's multi-task pre-training expects a task-instruction prefix our harness omitted, and found no such convention in its released training script and no change in behaviour across five candidate prefixes. We therefore report this as an observed limitation of that checkpoint on our data rather than tabulate a broken number.

SARI implementation. SARI is implemented directly from Xu et al. [34], because easse has no build for the Python version this pipeline requires. We cross-checked against a working easse installation on 50 examples of our own output, tokenized identically. The two correlate strongly (Pearson $r = 0.74$, Spearman $\rho = 0.77$) without being numerically interchangeable, with a mean absolute per-example difference of 4.3 SARI points concentrated in very short sentences, where the two differ on zero-addition edge cases that easse itself exposes as configurable rather than settled. All SARI values here use one implementation throughout, so the within-paper comparisons on which H1 rests are unaffected. Absolute values should not be compared against easse-based numbers from other papers.

Output vocabulary control. An earlier configuration of this study gave the MATE cells a word-level output vocabulary of 12,000 entries while the BPE cells predicted over 13,699 subword units, which varied output granularity alongside the input representation. Only 56.66% of reference word tokens on the SAMER test split were reachable under that word-level vocabulary, so roughly two in five correct target words were unproducible regardless of representation quality. Every number reported here comes from a corrected design in which all four cells predict over the same 13,699-entry BPE vocabulary, with the MATE cells retrained from scratch under it. We document this because the general point outlasts the paper: an ablation cell that changes the output space is not a controlled comparison of input representations.

Sample size and metric scope. The interaction is estimated from three seeds, giving two degrees of freedom, and is observed on SARI. Our analysis plan set a seed-expansion threshold at a minimum

detectable effect of one SARI point, and the realised value was 1.83. We did not expand, because the effect was already significant at the pre-specified level, and we record the decision here because it was taken with the result in view. The interaction does not appear on BERTScore (Section 6). We therefore present the finding as well-evidenced within its stated scope, and a replication at ten or more seeds across both metrics would extend that scope.

Single-reference evaluation. SARI was designed for multiple reference simplifications, and the SAMER split provides one reference per source sentence. Single-reference SARI is noisier and penalises legitimate alternative simplifications, which depresses all systems in Table 4 and compresses the differences between them.

Single-corpus evaluation. All results are obtained on SAMER, a corpus of literary Modern Standard Arabic. Generalisation to other genres, to machine-translated simplification data, and to dialectal Arabic remains to be established.

Pipeline dependence. AraBit-1.58 depends on an external morphological disambiguator, and it therefore inherits that component's errors, its latency, and its domain assumptions. CAMEL Tools is developed and evaluated primarily on Modern Standard Arabic, so our results should not be assumed to transfer to dialectal Arabic, where analyser accuracy degrades and the whole argument of this paper would need to be re-examined.

Metric limitations. SARI, BERTScore, and readability indices each capture a partial view of simplification quality, and none of them is a substitute for human judgement. This manuscript reports automatic metrics only, so no conclusion here has been checked against human judgement. That is a real limitation of the evidence and not a deferred task, and it bears most directly on the readability column discussed next. Concretely, on our full test split the two ternary cells score as *more* readable by the SAMER levelled-lexicon metric than their full-precision counterparts (mean lemma level 1.82 for MATE and 1.98 for BPE, against 2.08 and 2.24 at full precision) despite being the clearly *worse*-quality systems by SARI (13.4–14.6 vs. 20.9–24.0) – almost certainly because low-quality, short, repetitive ternary output draws on a narrower, more frequent vocabulary, not because ternary quantization improves simplification. The Readability column in Table 4 must therefore be read jointly with SARI and BERTScore, never in isolation: a system can score as “simple” by producing degenerate text as easily as by producing a genuinely good simplification.

Reproducibility of energy figures. Energy measurements are sensitive to hardware, driver version, batch size, and ambient conditions. We report the measurement methodology in full and we do not compare our

measured joules against analytically estimated figures from other papers, since the two quantities are not commensurable.

9. Conclusion and Future Work

This paper asked whether extreme quantization survives contact with a morphologically rich language, and whether the answer depends on how the input is represented. We proposed AraBit-1.58, which pairs a morphology-aware embedding front end with a natively trained ternary backbone, and we designed a controlled two-by-two study capable of separating the contribution of the front end from that of the backbone.

The answer to the first question is a qualified no at our scale. Ternary quantization costs a great deal of quality here: $24.01 \rightarrow 14.58$ SARI with BPE input and $20.93 \rightarrow 13.44$ with morphological input. It buys a $2.85\text{--}2.91 \times$ reduction in resident weight memory, and in our implementation, it buys nothing in latency or energy, both of which regress for want of a fused ternary kernel. The answer to the second question is a qualified yes. The cost of quantization is smaller with morphological input than with BPE input, by 1.93 SARI, an interaction that is significant across three seeds and that the gate measurements of Section 7 corroborate mechanistically, the ternary model relying more on the morphological branch than its full-precision counterpart does. We state the claim at the width the evidence supports. MATE does not raise absolute quality at either precision, the interaction is observed on SARI and not on BERTScore, and it is estimated from three seeds. Within those bounds the finding is clear: structured input matters more as weight precision falls. Testing it at a scale where the absolute numbers are competitive is the natural next step, and these results indicate where to look.

Two aspects of the design proved consequential. Retaining a learned fallback gate rather than a hard rule allowed the model to tolerate morphological analysis errors without a separate error-correction stage. Separating the memory accounting into ternary and full-precision components revealed that the realised compression factor at small scale is substantially below the naive ratio implied by weight width alone, which is a caution the efficiency literature does not always make explicit.

The approach carries costs that we report rather than minimise. Morphological disambiguation adds approximately 0.36 ms per generated token to the ternary MATE configuration, against 0.003 ms for the BPE control, which is the entire gap between our model-only and end-to-end latency columns. The MATE tables also introduce full-precision parameters that offset part of the ternary backbone's savings. Contrary to our design expectation, they do not reduce the embedding budget relative to byte-pair encoding. The

fully inflected pattern inventory is four times the size of the control vocabulary, and MATE ends up 0.53M parameters larger overall once the learned gate is counted.

Future work proceeds in six directions. First, we intend to replace the external analyser with a jointly trained segmenter, which would remove the pipeline dependency and the latency it imposes. Second, we plan to route sparse Mixture-of-Experts capacity through the ternary backbone, extending our earlier Switch Transformer work [14] into the quantized setting, so that capacity can be increased without a proportional increase in resident memory. Third, we will evaluate on dialectal Arabic, where morphological analysers are markedly less reliable and where the central assumption of this paper is genuinely open to question. Fourth, the component ablation of Section 6 argues for a simpler front end than the one we built, keeping clitic separation and the learned gate while dropping the root and pattern embeddings, and that variant should be trained and evaluated on its own terms rather than only as an ablation cell. Fifth, two of the error-analysis directions of Section 7 require hand annotation and are left for a follow-up study. One correlates per-sentence morphological analyser accuracy, measured against a gold-annotated sample, with per-sentence SARI, which would test whether the learned gate genuinely absorbs analyser errors or the pipeline remains brittle. The other is a manual failure taxonomy covering hallucination, meaning loss, agreement error, clitic attachment error, and failure to simplify, comparing AraBit-1.58 against the ternary BPE control. Only the gate-behaviour analysis of Section 7, which needs no hand annotation, is reported in this manuscript. Sixth, the divergence between SARI and BERTScore discussed in Section 6 calls for a replication with a larger seed budget and with a morphology-aware semantic metric, which would separate the three explanations we were able to offer but not to distinguish.

Code and Data Availability

The implementation, training and evaluation configurations, and the scripts that regenerate every table and figure reported here are available at <https://github.com/AMahfodh/AraBit>. The SAMER simplification corpus and the SAMER readability lexicon [28, 29] are distributed by their authors and are not redistributed in that repository. The pre-training corpus is the publicly available Arabic Wikipedia dump identified in Table 2.

Acknowledgements

The author employed Gemini-3.5-Flash for language polishing and readability enhancements, and Claude 3.5 Sonnet for coding and algorithm implementation. The author thoroughly reviewed and verified all AI-assisted outputs and takes full responsibility for the content and integrity of this manuscript.

References

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 5998–6008, 2017.
- [2] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontañón. FNet: Mixing tokens with Fourier transforms. *Proceedings of NAACL-HLT 2022*, 4296–4313, 2022.
- [3] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- [4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171–4186, 2019.
- [5] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23, 1–39, 2022.
- [6] H. Wang, S. Ma, L. Dong, S. Huang, H. Wang, L. Ma, F. Yang, R. Wang, Y. Wu, and F. Wei. BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.
- [7] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.
- [8] S. Ma, H. Wang, S. Huang, X. Zhang, Y. Hu, T. Song, Y. Xia, and F. Wei. BitNet b1.58 2B4T technical report. *arXiv preprint arXiv:2504.12285*, 2025.
- [9] L. Liu, Z. Zheng, C. Wang, T. Su, and Z. Yang. Binary neural networks for large language model: A survey. *arXiv preprint arXiv:2502.19008*, 2025.
- [10] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *Proceedings of the 54th annual meeting of the association for computational linguistics*, 1715–1725, 2016.
- [11] A. Alshanqiti, A. Alkhodre, A. Namoun, S. Albouq, and E. Nabil. A transformer Seq2Seq model with fast Fourier transform layers for rephrasing and simplifying complex Arabic text. *International Journal of Advanced Computer Science and Applications*, 14, 888–898, 2023.
- [12] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [13] A. Alshanqiti, A. Namoun, A. Alsughayyir, A.M. Mashraqi, A.R. Gilal, and S.S. Albouq. Leveraging DistilBERT for summarizing Arabic text: An extractive dual-stage approach. *IEEE Access*, 9, 135594–135607, 2021.

- [14] A. Alsughayyir, and A. Alshanqiti. A light but efficient switch transformer model for Arabic text simplification. *International Journal of Advanced and Applied Sciences*, 12, 71–78, 2025.
- [15] E. Frantar, S. Ashkboos, T. Hoefer, and D. Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *International conference on learning representations (ICLR)*, 2023.
- [16] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proceedings of machine learning and systems (MLSys)*, 2024.
- [17] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 2022.
- [18] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. *Proceedings of LREC 2014*, 1094–1101, 2014.
- [19] O. Obeid, N. Zalmout, S. Khalifa, D. Taji, M. Oudah, B. Alhafni, G. Inoue, F. Eryani, A. Erdmann, and N. Habash. CAMEL tools: An open source python toolkit for Arabic natural language processing. *Proceedings of the 12th language resources and evaluation conference (LREC)*, 7022–7032, 2020.
- [20] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak. Farasa: A fast and furious segmenter for Arabic. *Proceedings of NAACL-HLT 2016: demonstrations*, 11–16, 2016.
- [21] A.M. Alshanqiti, S. Albouq, A.B. Alkhodre, A. Namoun, and E. Nabil. Employing a multilingual transformer model for segmenting unpunctuated Arabic text. *Applied Sciences*, 12, 10559, 2022.
- [22] N. Habash. Introduction to Arabic natural language processing. Morgan & Claypool Publishers, 2010.
- [23] W. Antoun, F. Baly, and H. Hajj. AraBERT: Transformer-based model for Arabic language understanding. *Proceedings of the 4th workshop on open-source arabic corpora and processing tools (OSACT4), LREC 2020*, 9–15, 2020.
- [24] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [25] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568, 127063, 2024.
- [26] B. Zhang, and R. Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 2019.

- [27] I. Loshchilov, and F. Hutter. Decoupled weight decay regularization. *International conference on learning representations (ICLR)*, 2019.
- [28] B. Alhafni, R. Hazim, J.D. Pineros Liberato, M. Al Khalil, and N. Habash. The SAMER Arabic text simplification corpus. *Proceedings of LREC-COLING 2024*, 16079–16093, 2024.
- [29] M. Al Khalil, N. Habash, and Z. Jiang. A large-scale leveled readability lexicon for standard Arabic. *Proceedings of LREC 2020*, 3053–3062, 2020.
- [30] S.S. Al-Thanyyan, and A.M. Azmi. Simplification of Arabic text: A hybrid approach integrating machine translation and transformer-based lexical model. *Journal of King Saud University – Computer and Information Sciences*, 35, 101662, 2023.
- [31] M.K. Eddine, N. Tomeh, N. Habash, J. Le Roux, and M. Vazirgiannis. AraBART: A pretrained Arabic sequence-to-sequence model for abstractive summarization. *Proceedings of the seventh arabic natural language processing workshop (WANLP)*, 31–42, 2022.
- [32] E.M.B. Nagoudi, A. Elmadany, and M. Abdul-Mageed. AraT5: Text-to-text transformers for Arabic language generation. *Proceedings of the 60th annual meeting of the association for computational linguistics*, 628–647, 2022.
- [33] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *Advances in neural information processing systems*, 36, 10088–10115, 2023.
- [34] W. Xu, C. Napoles, E. Pavlick, Q. Chen, and C. Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4, 401–415, 2016.
- [35] T. Zhang, V. Kishore, F. Wu, K.Q. Weinberger, and Y. Artzi. BERTScore: Evaluating text generation with BERT. *International conference on learning representations (ICLR)*, 2020.
- [36] M. El-Haj, and P. Rayson. OSMAN: A novel Arabic readability metric. *Proceedings of LREC 2016*, 250–255, 2016.
- [37] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting of the association for computational linguistics*, 311–318, 2002.
- [38] E. Sulem, O. Abend, and A. Rappoport. BLEU is not suitable for the evaluation of text simplification. *Proceedings of EMNLP 2018*, 738–744, 2018.