A Comparative Study of On-Device LLM Text Summarization: Evaluating 4-bit, 8-bit, and BFloat16 Quantization of Qwen2.5-0.5B-Instruct on Apple Silicon with the MLX Framework

Nanda Rosyadi, Benben Bagus, Pradiptya Kahvi S, Lailil Muflikhah, and Tirana Noor Fatyanosa*

Universitas Brawijaya, Indonesia
nandamochammad@student.ub.ac.id, benben_bagus@student.ub.ac.id, pradiptya_kahvi@student.ub.ac.id, lailil@ub.ac.id, fatyanosa@ub.ac.id
*Corresponding author

Abstract. This paper evaluates *on-device* abstractive text summarization with a small Large Language Model (LLM) on consumer hardware, focusing on the impact of weight *quantization* on summary quality and computational efficiency. We conduct a paired comparison of Qwen2.5-0.5B-Instruct under three precision configurations (4-bit, 8-bit, and BFloat16) on the same 50 California bills from the BillSum `ca_test` split, executed on a 2024 MacBook Pro with an Apple M4 Pro system-on-chip, 24 GB unified memory, and Metal acceleration through the MLX framework. Quality is assessed with ROUGE, BLEU, and BERTScore against the human-written references, and efficiency through throughput (summaries per second) and peak MLX memory (GB). We find no statistically distinguishable difference in summary quality across the three configurations at this sample size: ROUGE-1 spans 0.350 to 0.355 and BERTScore spans 0.841 to 0.847 (absolute spread 0.006). System metrics, in contrast, are monotonic: 4-bit quantization is 52% faster (1.34 vs. 0.88 summaries/s) and uses 26% less peak memory (1.23 GB vs. 1.66 GB) than the BFloat16 baseline. Together, these results indicate that 4-bit quantization is a Pareto improvement within the resolution of this evaluation and a reasonable default for privacy-preserving, offline summarization with this model and framework on Apple Silicon.

Keywords: Text Summarization · Large Language Models · Quantization · Apple Silicon · MLX Framework · On-Device Inference

1 Introduction

Text summarization is a fundamental task in Natural Language Processing (NLP), compressing lengthy documents into shorter, coherent representations that preserve key information [8,13]. Large Language Models (LLMs) have shifted the field from extractive techniques toward abstractive summarization, in which models paraphrase the source text into fluent, human-like summaries [7,13]. The

© The Author(s) 2026
M. Murtadha et al. (eds.), *Proceedings of the 2026 11th International Conference on Sustainable Information Engineering and Technology (SIET 2026)*, Atlantis Highlights in Computer Sciences 26,
https://doi.org/10.2991/978-94-6239-828-3_25

most capable systems today, however, are cloud-hosted and accessed through paid APIs, raising three concerns: *data privacy*, since sensitive documents must be sent to third-party servers; *operational cost*, as per-token billing scales with usage; and *network dependence*, which precludes offline use.

The emergence of compact instruction-tuned LLMs (sub-billion to a few-billion parameters) combined with weight *quantization* [1,2,9] has opened the door to running such models entirely on consumer devices, where all three concerns above are directly addressed. In parallel, Apple's MLX framework [4] exploits the unified-memory architecture of Apple Silicon for efficient on-device inference. Together, these developments motivate the central question of this paper: *can quantized small LLMs running fully on-device deliver summarization quality comparable to higher-precision baselines while remaining responsive on consumer hardware?*

To answer this question we conduct a paired, controlled comparison of Qwen2.5-0.5B-Instruct[1] under three precision regimes (4-bit, 8-bit, and BFloat16) on the BillSum legislative summarization benchmark, executed entirely on a 2024 MacBook Pro with the Apple M4 Pro system-on-chip through the MLX framework.

The contributions are threefold: (i) a *paired* empirical comparison of 4-bit, 8-bit, and BFloat16 quantization for abstractive summarization on a small instruction-tuned LLM, with weight precision as the only independent variable; (ii) lexical (ROUGE, BLEU) and semantic (BERTScore) quality metrics paired with model-specific system metrics (throughput, peak MLX memory) on real consumer hardware, with full per-document outputs released for replication; and (iii) a concrete deployment recommendation: on this workload, 4-bit quantization is faster and more memory-efficient than higher-precision configurations, with no quality difference that survives correction for multiple comparisons at $n = 50$.

2 Related Work

2.1 Background

Modern abstractive summarization is dominated by Transformer-based encoder-decoder models such as BART [7] and PEGASUS [13], the latter explicitly tailored to summarization via gap-sentence pretraining. More recently, instruction-tuned small LLMs from the Qwen, Llama, and Phi families have been shown to perform competitive zero- or few-shot summarization, in some configurations approaching the quality of 70B-scale models on news benchmarks [11], making such models viable candidates for on-device deployment. Quality is typically assessed with a mix of lexical-overlap metrics (ROUGE [8], BLEU [10]) and embedding-based scores such as BERTScore [14].

Weight *quantization* reduces model precision to lower memory footprint and accelerate inference. Dettmers et al. [1] showed that 8-bit matrix multiplication preserves accuracy for billion-parameter LLMs. GPTQ [2] and AWQ [9] further

[1] Model checkpoint: `https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct`

enable 3 to 4 bit quantization with minimal degradation. Comparable group-wise schemes are implemented by on-device runtimes such as MLX [4] and llama.cpp [3], with MLX designed natively around Apple Silicon's unified-memory architecture.

2.2 Position of This Work

Two recent peer-reviewed studies are most relevant. Xu et al. [11] evaluated 19 sub-4B models on news summarization in a server setting, varying the *model* at fixed precision. Yadav and Bhargavi [12] instead fix the model (Llama-3.2-3B) and apply 4-bit post-training quantization for Android, reporting a 68.66% size reduction but no comparison across precision levels. Our study sits between these: like Yadav and Bhargavi we fix the model and study quantization, but like Xu et al. we focus on summarization quality measured with ROUGE, BLEU, and BERTScore. We go beyond both by (i) comparing three precision levels (4-bit, 8-bit, BFloat16) on the *same* 50 BillSum documents (paired evaluation), and (ii) using the MLX framework on Apple Silicon rather than the BitsAndBytes/CUDA stack on mobile or cloud. To the best of our knowledge, no prior peer-reviewed work reports a controlled comparison of 4-bit, 8-bit, and BFloat16 precision under the MLX runtime for legal-domain summarization.

3 Methodology

3.1 Dataset

We use the BillSum corpus [6], a publicly available collection of US Congressional and California state bills paired with human-written summaries. Specifically, we use the `ca_test` split of the `FiscalNote/billsum` release: 1,237 California bills, treated in the original BillSum paper as an out-of-distribution transfer set rather than the standard test split. Its documents are long (median 1,367 words) and written in dense, formal legislative language, providing a stress test for paraphrasing ability with professionally written references. For tractable repeated runs across three quantization regimes on a single laptop, we draw a fixed sample of 50 documents from this split using a deterministic shuffle (`seed=42`) and reuse the *same* 50 documents across all three configurations. This guarantees a paired comparison: any between-configuration difference in metrics is attributable to weight precision, not to input variability.

3.2 Model and Quantization Configurations

We evaluate **Qwen2.5-0.5B-Instruct**, a 0.5-billion-parameter instruction-tuned LLM from Alibaba's Qwen series, under three precision regimes drawn from the publicly available `mlx-community` checkpoints:[2] **BFloat16** (∼0.9 GB on disk), the highest-precision checkpoint distributed through MLX, serving as the quality

[2] Checkpoints `Qwen2.5-0.5B-Instruct-{bf16,8bit,4bit}` at `https://huggingface.co/mlx-community/`.

baseline; **8-bit** (∼0.5 GB), affine quantization through MLX's built-in quantizer; and **4-bit** (∼0.3 GB), produced by the same quantizer.

All other generation hyperparameters (prompt template, maximum number of new tokens, sampling strategy) are held constant across configurations to isolate quantization as the sole independent variable.

3.3 Prompt Template

We use a minimal completion-style prompt, passed to the model as a raw string. The Qwen2.5 chat template is *not* applied. The prompt consists of the line `Summarize the following text concisely:`, a blank line, the full BillSum source text, a blank line, and the trailing literal `Summary:`. The trailing literal acts as a stop-anchor: the generated continuation is treated as the candidate summary and is recovered by splitting the model output on this marker. We deliberately avoid few-shot exemplars to keep the input length predictable and the prompt comparable across configurations. Because the chat template is not applied, the model operates in text-continuation rather than instruction-following mode (Section 5).

3.4 Decoding Configuration

To remove sampling variance as a confound, all configurations use *deterministic greedy decoding*: temperature 0.0 (greedy argmax at each step via the `mlx_lm` sampler), maximum new tokens of 150, and no top-k/top-p sampling.

3.5 Quantization Scheme

We rely on MLX's built-in post-training weight quantizer [4], which applies group-wise affine quantization to the model's linear layers. Given a weight tensor $W \in \mathbb{R}^{m\times n}$ and a target bit width $b \in \{4, 8\}$, the quantizer partitions each row into contiguous groups $W_i \in \mathbb{R}^g$ and computes a scale s_i and a floating-point bias β_i per group:

$$\begin{aligned} \beta_i &= \min(W_i), \qquad s_i = \frac{\max(W_i) - \beta_i}{2^b - 1}, \\ \widetilde{W}_i &= s_i\, \hat{W}_i + \beta_i, \end{aligned} \tag{1}$$

where $\hat{W}_i = \mathrm{clip}(\mathrm{round}((W_i - \beta_i)/s_i),\, 0,\, 2^b - 1)$ is the stored integer code and s_i, β_i are kept in half precision. At inference, Equation (1) recovers an approximation $\widetilde{W}_i$ without materialising the full BF16 tensor: $b = 4$ yields 16 representable levels per group and $b = 8$ yields 256. The BFloat16 baseline is the unquantized hub checkpoint, while the 8-bit and 4-bit checkpoints are produced by MLX's `convert` pipeline at its default group size. We do not quantize activations and do not calibrate. The embedding layer *is* quantized alongside the linear layers in the 8-bit and 4-bit checkpoints, and the output projection shares the embedding weights (`tie_word_embeddings=True`), so no separate lm-head exists. The tokenizer is unmodified.

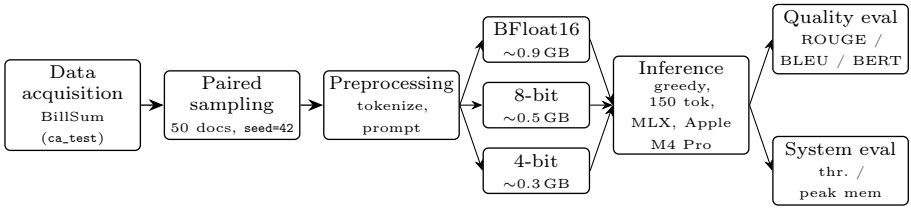


Fig. 1. End-to-end implementation pipeline for the on-device summarization study. The same 50 BillSum documents (`seed=42`) flow through three quantization branches under identical prompt and decoding configurations.

3.6 Hardware and Software Stack

Experiments are run on a 2024 MacBook Pro with the Apple M4 Pro system-on-chip (20-core GPU, 24 GB unified memory shared between CPU and GPU), accelerated by Apple Metal through the MLX framework [4] with the Python `mlx-lm` bindings.

3.7 Pipeline Overview

Figure 1 summarises the end-to-end pipeline.

3.8 Evaluation Metrics

Quality metrics compare a candidate summary $\hat{y}$ against the BillSum reference y. **ROUGE-N** [8] measures clipped n-gram overlap F1. We report ROUGE-1, ROUGE-2, and ROUGE-L (longest common subsequence), computed with the `rouge-score` package with stemming enabled. **BLEU** [10] is the corpus-level geometric mean of modified n-gram precisions ($n \leq 4$) multiplied by the brevity penalty $\mathrm{BP} = \min(1, \exp(1-r/c))$, computed via `sacrebleu` on the 0 to 100 scale. The brevity penalty matters here because our outputs (mean 126 to 130 words) are shorter than the BillSum references (mean 320 words). **BERTScore** [14] reports the F1 of greedy-matched cosine similarity between contextual embeddings of $\hat{y}$ and y with a RoBERTa-large backbone.

Efficiency metrics. Throughput is generation-only summaries per second over the 50-document set, excluding model load time (0.80, 0.86, and 0.97 s for BFloat16, 8-bit, and 4-bit). **Peak MLX memory** is the maximum MLX-allocated memory during inference (in GB) queried via `mx.get_peak_memory()`, which is specific to the model under test and unaffected by other resident processes.

4 Results

4.1 Quality

We treat BFloat16 as the quality baseline and compare 8-bit and 4-bit against it on the *same* 50 documents with identical decoding. Table 1 reports quality across

Table 1. Summarization quality across quantization regimes (paired evaluation on 50 BillSum samples, Qwen2.5-0.5B-Instruct on Apple M4 Pro under MLX). R-1, R-2, R-L denote ROUGE-1, ROUGE-2, and ROUGE-L F1; BLEU is reported on the 0 to 100 scale (`sacrebleu`); BERT denotes BERTScore F1. Best value per column in **bold**.

Quant.	R-1	R-2	R-L	BLEU	BERT
BFloat16	0.3498	0.1248	0.2048	3.21	0.8460
8-bit	**0.3551**	**0.1269**	0.2055	**3.49**	**0.8473**
4-bit	0.3495	0.1233	**0.2159**	3.30	0.8413

Table 2. Paired per-document contrasts for the declared twelve-test family. Cells give the mean difference Δ and 95% bootstrap confidence interval, both in units of 10^{-3}, followed by the two-sided sign-flip permutation p-value. No contrast survives Holm correction ($m = 12$).

	8-bit − BF16	4-bit − BF16	8-bit − 4-bit
ROUGE-1	$+5.3\ [-4.0, +15.4]\ .305$	$-0.3\ [-16.6, +18.2]\ .977$	$+5.5\ [-14.1, +23.5]\ .574$
ROUGE-2	$+2.1\ [-10.2, +15.5]\ .761$	$-1.5\ [-16.5, +14.1]\ .852$	$+3.6\ [-15.7, +22.6]\ .728$
ROUGE-L	$+0.7\ [-8.2, +9.5]\ .888$	$+11.1\ [-1.4, +24.4]\ .106$	$-10.4\ [-22.1, +1.0]\ .095$
BERTScore	$+1.2\ [-1.7, +4.1]\ .427$	$-4.7\ [-9.8, +0.6]\ .076$	$+6.0\ [+0.6, +11.0]\ .035$

the three configurations. The differences are small in absolute terms: ROUGE-1 spans 0.3495 to 0.3551 (spread 0.0056) and BERTScore spans 0.8413 to 0.8473 (spread 0.0060). The 8-bit configuration leads on four of five metrics (ROUGE-1, ROUGE-2, BLEU, BERTScore), 4-bit tops ROUGE-L, and the BFloat16 baseline does not lead on any. BLEU values (≈ 3.2 to 3.5) are dominated by the brevity penalty: our outputs (mean 126 to 130 words) are much shorter than the BillSum references (mean 320 words), giving BP = 0.205, 0.210, and 0.216 for BFloat16, 8-bit, and 4-bit. The underlying n-gram precisions are nearly identical across configurations (unigram precision 53.9, 54.7, and 53.8), so the penalty-free BLEU is 15 to 17 in every arm. BERTScore is ≈ 0.84 throughout. However, raw BERTScore is not calibrated to a meaningful zero: scoring our outputs against deliberately mismatched references still yields 0.808, so we interpret BERTScore only as a relative comparison across precisions, not as evidence of absolute semantic preservation.

We define the test family as all twelve paired contrasts (four metrics, three configuration pairs each) and report every contrast in Table 2 as mean per-document differences with 95% bootstrap confidence intervals and two-sided sign-flip permutation p-values. To account for multiple testing, p-values are corrected with the Holm step-down procedure [5], which controls the family-wise error rate across all $m = 12$ tests. BLEU is excluded because it is a corpus-level statistic with no meaningful per-document value at these lengths; a corpus-level paired bootstrap over the same 50 documents leaves all three BLEU contrasts non-significant as well (smallest $p = 0.10$). No contrast survives the correction. The only contrast below 0.05 before correction is BERTScore for 8-bit vs. 4-bit

Table 3. System-level efficiency across quantization regimes (paired evaluation, 50 BillSum samples). Throughput is generation-only summaries per second (model load excluded); mean latency is per-document generation latency in seconds; peak memory is MLX-allocated memory queried via `mx.get_peak_memory()`. Best value per column in **bold**.

Quant.	Throughput (sum/s)	Latency (s)	Peak mem (GB)
BFloat16	0.88	1.14	1.66
8-bit	1.10	0.91	1.39
4-bit	**1.34**	**0.75**	**1.23**

($p = 0.035$). At $n = 50$ the 95% confidence intervals on the paired differences bound any effect at 0.024 ROUGE-1 and 0.011 BERTScore.

Output length does not vary meaningfully with precision: mean 126.5, 127.4, and 130.2 words (SD 16.8, 16.9, 12.8) for BFloat16, 8-bit, and 4-bit, with no paired difference approaching significance ($p \geq 0.13$). Every output in every configuration terminates at the 150-token generation cap (retokenized lengths of 149 to 150 tokens), and 46 to 49 per configuration end without terminal punctuation, so the generated texts are prefixes of longer summaries rather than naturally terminated ones. Because the cap is identical across configurations and truncation is symmetric, the paired comparison is not confounded by the cap. Absolute quality figures are conservative under a fixed 150-token output budget.

4.2 System Efficiency

In contrast to quality, system metrics are monotonic in precision (Table 3). Moving from BFloat16 to 4-bit yields a 52% increase in throughput (0.88 → 1.34 summaries/s) and a 26% reduction in peak MLX memory (1.66 → 1.23 GB). Mean per-document latency drops by 34% (1.14 s → 0.75 s). The 8-bit configuration sits between the other two on all axes. The configurations ran sequentially in a single pass in the fixed order BFloat16, 8-bit, 4-bit, so thermal throttling would slow the later, faster configurations and understate the 4-bit speedup. Excluding the first (warm-up) document leaves the throughput ratio at 1.52× (within a single pass), and the paired per-document latency difference between 4-bit and BFloat16 is −0.393 s (95% CI $[-0.412, -0.381]$).

The measured 26% memory reduction is below the naive 75% implied by precision alone. On-disk weights are 0.92, 0.49, and 0.26 GB for BFloat16, 8-bit, and 4-bit, while peak memory exceeds the weights by 0.74, 0.90, and 0.97 GB respectively. This runtime component (KV-cache, activations, and framework allocations) dominates once the weights shrink, so memory savings saturate quickly at lower precision.

4.3 Behavioural Shift under Quantization

We close with an exploratory analysis, formulated post hoc after inspecting the outputs. The statistical equivalence of Section 4.1 does not mean the config-

Table 4. Copy rate: percentage of generated word 4-grams appearing verbatim in the source document (mean over the 50 paired documents; exploratory post-hoc analysis).

Configuration	Copy rate (%)	SD
BFloat16	27.3	15.6
8-bit	31.2	15.5
4-bit	35.5	21.8
Human reference	*30.2*	—

urations behave identically: no 4-bit output is byte-identical to its BFloat16 counterpart (0 of 50 documents; the median pair diverges by the second word), and 8-bit matches BFloat16 on only 5 of 50. Table 4 quantifies one systematic difference. The *copy rate*, the fraction of generated word 4-grams appearing verbatim in the source document, rises monotonically as precision drops, from below the human-reference rate (27.3%) to above it (35.5%). The trend is statistically significant (within-document permutation trend test, $p = 0.012$ under three-arm exchangeability; the more conservative two-arm sign-flip test gives $p = 0.03$; uncorrected, as this exploratory test lies outside the declared confirmatory family) and uncorrelated with output length ($r = -0.02$). In this model, quantization thus appears to shift the model from abstractive toward more extractive behaviour even though no overlap metric in Table 2 registers a difference. Increased verbatim copying would also produce exactly the pattern in Table 1 in which 4-bit scores highest on ROUGE-L yet lowest on BERTScore. However, the per-document correlation between the copy-rate shift and the ROUGE-L shift is weak ($r = 0.19$), so we advance this mechanism only as a hypothesis consistent with the aggregate direction, to be tested on the full split.

4.4 Trade-Off Analysis and Recommendations

The combined picture is clear: **quality is statistically indistinguishable at this sample size, efficiency is monotonic.** The 4-bit configuration achieves 1.52× throughput and uses 0.74× the memory of BFloat16 (8-bit: 1.25×, 0.84×), while the quality spreads (0.0056 ROUGE-1, 0.0060 BERTScore) fail every significance test in Table 2. We note that the point estimates mildly favour 8-bit on quality: it leads four of the five metrics (4-bit leads only ROUGE-L), has the highest brevity-penalty-free BLEU (16.6 against 15.6 and 15.3), and the only nominally significant contrast disfavours 4-bit. None of these survives correction at $n = 50$, so we base the recommendation on the efficiency axis. 4-bit is therefore a Pareto improvement within the resolution of this evaluation and our recommended default for deploying this model on Apple Silicon under MLX. The 8-bit configuration is the conservative fallback for larger models or untested domains, and BFloat16 is reserved for situations that require reproducing reference numerics exactly. Promising follow-up work includes scaling to the full `ca_test` split, adding human evaluation, and repeating the study on other small

LLMs (Phi-3.5, Gemma-2B, Llama-3.2-1B) to test whether the quality equivalence generalises.

5 Limitations

Our paired evaluation uses 50 documents from the `ca_test` split. Because our confidence intervals scale with n, evaluating the full split (1,237 documents) would narrow them by roughly a factor of five and could render some of the small differences observed here statistically detectable. We would expect such differences to remain below the margin of practical significance for deployment. The copy-rate analysis of Section 4.3 is exploratory and post hoc. The extractiveness shift and the mechanism it suggests await confirmation on the full split. The evaluation is restricted to California bills (the `ca_test` transfer split) and to automatic metrics, so transfer to BillSum's US Congressional test split is untested. Human evaluation along factual-consistency and faithfulness axes would strengthen the quality claim, since ROUGE and BERTScore can miss certain factual errors. Our prompt is passed as a raw string without the Qwen2.5 chat template, so the model runs in text-continuation rather than instruction-following mode. This holds identically across the three configurations, but chat-formatted prompting could raise absolute quality and reduce truncation. Only Qwen2.5-0.5B-Instruct is studied on a single M4 Pro, so the quality-equivalence finding may not transfer to larger models, other instruction-tuning recipes, or non-summarization tasks, and absolute throughput and memory figures will differ on other hardware.

6 Conclusion

We presented a paired, controlled study of 4-bit, 8-bit, and BFloat16 quantization for on-device abstractive summarization with Qwen2.5-0.5B-Instruct under the MLX framework. Quality is statistically indistinguishable across the three configurations at $n = 50$, while system efficiency is monotonic: 4-bit is 52% faster and uses 26% less peak MLX memory than BFloat16. 4-bit quantization therefore dominates on both efficiency axes at no quality cost this evaluation can detect, and should be the default for on-device summarization with this model on Apple Silicon. The implication is that privacy-preserving, offline summarization with this class of model is practical on consumer hardware, with any quality cost below what a 50-document paired evaluation can detect.

Reproducibility. The model checkpoints are publicly available through the `mlx-community` hub (quantization group size 64, `tie_word_embeddings=True`). Experiments used MLX 0.31.2 with `mlx-lm` 0.31.3. Evaluation used `rouge-score` 0.1.2, `sacrebleu` 2.6.0, and `bert-score` 0.3.13 (RoBERTa-large backbone), with `seed=42` for data sampling. Evaluation scripts, prompts, per-document outputs, and aggregated metrics are archived at `https://doi.org/10.5281/zenodo.21682325`.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Acknowledgments. This research was funded by the Faculty of Computer Science, Brawijaya University, through the *Hibah Penelitian Tugas Akhir Mahasiswa FILKOM* (Master's Thesis Research Grant for FILKOM Students) scheme. The authors gratefully acknowledge this support.

References

1. Dettmers, T., Lewis, M., Belkada, Y., Zettlemoyer, L.: LLM.int8(): 8-bit matrix multiplication for transformers at scale. In: Advances in Neural Information Processing Systems, vol. 35, pp. 30318–30332 (2022)
2. Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: GPTQ: Accurate post-training quantization for generative pre-trained transformers. In: International Conference on Learning Representations (ICLR) (2023)
3. Gerganov, G., contributors: llama.cpp: LLM inference in C/C++. `https://github.com/ggml-org/llama.cpp` (2023)
4. Hannun, A., Digani, J., Katharopoulos, A., Collobert, R.: MLX: Efficient and flexible machine learning on Apple Silicon. `https://github.com/ml-explore/mlx` (2023)
5. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2), 65–70 (1979)
6. Kornilova, A., Eidelman, V.: BillSum: A corpus for automatic summarization of US legislation. In: Workshop on New Frontiers in Summarization, pp. 48–56. ACL (2019). `https://doi.org/10.18653/v1/D19-5406`
7. Lewis, M., et al.: BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Annual Meeting of the Association for Computational Linguistics, pp. 7871–7880. ACL (2020). `https://doi.org/10.18653/v1/2020.acl-main.703`
8. Lin, C.-Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out, pp. 74–81. ACL (2004)
9. Lin, J., et al.: AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In: Proceedings of MLSys (2024)
10. Papineni, K., Roukos, S., Ward, T., Zhu, W.-J.: BLEU: A method for automatic evaluation of machine translation. In: Annual Meeting of the Association for Computational Linguistics, pp. 311–318. ACL (2002). `https://doi.org/10.3115/1073083.1073135`
11. Xu, B., Chen, Y., Wen, Z., Liu, W., He, B.: Evaluating small language models for news summarization: Implications and factors influencing performance. In: Proceedings of NAACL 2025, pp. 4909–4922. ACL (2025). `https://doi.org/10.18653/v1/2025.naacl-long.253`
12. Yadav, A., Bhargavi, R.C.: Optimizing LLMs using quantization for mobile execution. In: ICT Analysis and Applications (ICT4SD 2025). Lecture Notes in Networks and Systems, vol. 1654. Springer, Cham (2025). `https://doi.org/10.1007/978-3-032-06697-8_33`
13. Zhang, J., Zhao, Y., Saleh, M., Liu, P.J.: PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In: International Conference on Machine Learning, pp. 11328–11339. PMLR (2020)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

