Workload-Driven Optimization for On-Device Real-Time Subtitle Translation

Tsz-To Wong
National Yang Ming Chiao Tung University, Hsinchu, Taiwan
tsztowong.cs13@nycu.edu.tw

Abstract

This report studies on-device English-to-Traditional-Chinese subtitle translation for Taiwan under short inputs, short outputs, batch-size-one inference, low latency, and privacy constraints. These conditions limit the value of optimizations designed for long-context or high-throughput language-model serving.

Starting from LMT-60-0.6B [2], preliminary profiling suggests that vocabulary projection becomes a more important decode-time cost after GGUF quantization reduces the relative cost of Transformer blocks. We replace the original 151k-token vocabulary with a 64k-token subtitle-domain tokenizer, migrate the embedding space, and adapt the model through embedding calibration followed by full supervised fine-tuning.

On a fixed 500-example subset of the OpenSubtitles2024 test set, the LocalSubs achieves a 59.2% tie-excluded win rate against Google Translate under GPT-4o pairwise judging. Performance is strongest on short cues and declines as cue length increases. Preliminary Apple M2 Metal measurements on a 64k-vocabulary model show a 1.63× speedup over a 151k-vocabulary profiling baseline. The raw benchmark configuration is incomplete, so the latency result is treated as preliminary.

1 Introduction

Real-time subtitle translation imposes a different set of constraints from conventional machine translation and general-purpose language-model serving. Each request typically contains one current subtitle cue and only a small amount of preceding context. Both the input and output are short, inference is effectively performed at batch size one, and the translation must be produced before the subtitle display window expires. For privacy-sensitive applications, the complete pipeline must also run on local consumer hardware while generating natural Traditional Chinese as used in Taiwan.

These characteristics change the inference bottleneck. Techniques designed for long contexts, repeated prefixes, or large batches offer limited benefit when prompts are short and requests arrive sequentially. In this setting, fixed per-request overhead, decode-time computation, vocabulary projection, and output token count can have a greater effect on end-to-end latency. A large multilingual tokenizer may further increase the output-projection cost while providing inefficient tokenization for domain-specific Chinese subtitle text.

This work investigates a workload-driven optimization path based on LMT-60-0.6B [2]. The original 151k vocabulary is replaced with a 64k subtitle-domain tokenizer trained on English and Taiwan Traditional Chinese subtitle text. Because tokenizer replacement changes token identities and invalidates the original embedding matrix, the model is adapted through embedding migration, an embedding-calibration stage, and full supervised fine-tuning. The resulting system is evaluated using pairwise preference judgments against a fixed Google Translate anchor, while deployment performance is examined through preliminary Apple M2 Metal measurements. The main contributions are:

- A 64k-vocabulary subtitle tokenizer that reduces the output-projection dimension and improves Chinese token density.
- An embedding migration and two-stage adaptation procedure for replacing the tokenizer of a pretrained translation model.
- A translation-quality evaluation based on pairwise preference judgments, including context ablation and cue-length analysis.

2 Task and Design Goals

2.1 Cue-level context-aware translation

The model receives up to three previous English subtitle cues as context and one current cue as the translation target. It must output only the Taiwan Traditional Chinese translation of the current cue. Context is used for disambiguation and tone continuity, not as additional translation content.

The task is

$$y_t = f(x_{t-k:t-1}, x_t), \qquad 0 \le k \le 3,$$

where x_t is the current English cue and y_t is its translation.

The main quality requirements are semantic correctness, current-cue-only output, natural Taiwan usage, concise subtitle style, and stable handling of short ambiguous utterances.

2.2 On-device constraints

The target environment has four practical constraints:

- **Low latency:** each cue should complete within the subtitle display window.
- **Low batch:** interactive playback is effectively batch size one.
- **Limited hardware:** the model must run on consumer devices such as Apple M-series systems.
- **Privacy:** subtitle content should remain on the local device.

3 Method

3.1 Workload profile and decode cost

This workload uses short prompts, short outputs, and little reusable prefix. Optimizations aimed mainly at long attention sequences or large-batch inference therefore offer limited benefit.

Preliminary profiling suggests that, after GGUF Q5_K_M quantization reduces the relative cost of Transformer blocks, the output projection becomes a more important part of decode time. Each decode step computes

$$\text{logits} = hW_{\text{vocab}}^{\top},$$

with cost proportional to $|\mathcal{V}|d_{\text{model}}$. The original vocabulary contains 151,936 tokens with hidden size 1024, so every decode step projects to more than 150,000 logits.

This observation motivates a combined strategy:

1. quantization reduces per-step Transformer cost; and
2. tokenizer redesign reduces the projection dimension and may reduce the number of Chinese decode tokens.

Because the operation-level profiling log is not yet complete, this report treats vocabulary projection as a supported hypothesis rather than a fully isolated bottleneck.

3.2 64k-vocabulary subtitle tokenizer

We train a ByteLevel BPE tokenizer on English and Taiwan Traditional Chinese subtitle text. BPE provides an open-vocabulary subword representation while allowing the vocabulary to be tuned to the target domain [1]. The base tokenizer contains 64,020 tokens; four structural tokens increase the final vocabulary to 64,024.

Table 1: Effect of tokenizer replacement

Metric	Original tokenizer	64k subtitle tokenizer
Vocabulary size	151,936	64,024
Chinese characters per token	1.05	1.40
Model parameters	∼596M	∼506M

The smaller vocabulary reduces the output-projection dimension by approximately 57.9%. On the same Chinese text, the subtitle tokenizer also increases character density, which reduces the number of tokens required to represent the output. This fixed-text measurement isolates tokenizer efficiency from differences in model generation behavior.

3.3 Embedding migration and two-stage adaptation

Tokenizer replacement changes token identities and prevents direct reuse of the original embedding matrix. We initialize the new embedding space using string-level correspondence:

- copy the original embedding when the token string already exists;
- otherwise, tokenize the new token with the original tokenizer and average the corresponding embeddings; and
- use a mean fallback only when decomposition is impossible.

All 64k base-vocabulary tokens were initialized by direct copy or sub-token averaging. The four structural tokens were initialized separately from their original string decompositions. No mean fallback was required.

The model is then adapted in two stages:

$$\text{embedding migration} \rightarrow \text{embedding calibration} \rightarrow \text{full SFT}.$$

During embedding calibration, Transformer layers are frozen while embeddings, the output head, and RMSNorm parameters are updated. Full supervised fine-tuning then updates the complete model on the final subtitle dataset.

4 Experimental Setup

4.1 Model and training data

The base model is NiuTrans/LMT-60-0.6B, a compact multilingual translation model [2, 3]. The LocalSubs is trained on a quality-filtered, length-rebalanced subtitle SFT dataset.

The final SFT dataset combines rule-based cleaning, Traditional Chinese filtering, LLM-assisted subtitle-pair quality filtering, shuffling, and increased coverage of medium-to-long cues. The filtering model belongs to the Qwen3 family [5]. Detailed construction rules are provided in Appendix B.

Table 2: Core experimental configuration

Item	Setting
Base model	NiuTrans/LMT-60-0.6B
Translation direction	English to Taiwan Traditional Chinese
Final SFT dataset size	233,088 examples
Input format	Up to three context cues plus one current cue
Original vocabulary	151,936 tokens
Adapted vocabulary	64,024 tokens
Primary evaluation set	Fixed 500-example subset of the OpenSubtitles2024 test set

4.2 Evaluation data and protocol

The OpenSubtitles2024 benchmark contains bilingual subtitle alignments held out for machine-translation development and evaluation [4]. The primary quality evaluation uses a fixed 500-example subset.

The main protocol is pairwise preference against a fixed Google Translate anchor. GPT-4o judges the candidate and anchor translations with randomized A/B order and temperature zero. The metric excludes ties:

$$\text{win rate} = \frac{\text{wins}}{\text{wins} + \text{losses}}.$$

LLM-based pairwise judging is scalable but can exhibit position and other biases, so randomized candidate order and a future human audit are important [6]. All results are reported with sample size and win/loss/tie counts. Character-level F1, simplified-Chinese rate, and English echo rate are used only as surface-form diagnostics.

The exact GPT-4o API snapshot was not retained; the experiment record identifies only the model alias. This is a reproducibility limitation. GPT-4o mini is used as a cloud baseline [8, 9].

4.3 Deployment benchmark

The deployment path uses llama.cpp, GGUF Q5_K_M quantization, and the Apple Metal backend. llama.cpp provides local inference, integer quantization, and optimized Apple Silicon support [7]. The recorded latency summary uses a separate 64k-vocabulary profiling model, not the final model evaluated for translation quality.

The exact raw log, llama.cpp commit, converted-model checksum, warm-up count, and run count are incomplete. The latency measurements are therefore profiling evidence rather than final deployment claims.

5 Results

5.1 Pairwise translation quality

Table 3: Pairwise preference results against Google Translate

System	n	W/L/T	Win rate	Interpretation
GPT-4o mini	500	211/116/173	64.6%	Cloud baseline evaluated with the same anchor
Ours	500	229/158/113	59.2%	Main local-model result

The LocalSubs achieves a 59.2% tie-excluded win rate against Google Translate, compared with 64.6% for GPT-4o mini under the same anchor-relative evaluation protocol. The 5.4-percentage-point difference suggests that the local 0.6B model approaches the translation quality of a commercial cloud model despite operating under substantially stricter constraints, including on-device execution, batch-size-one inference, low-latency requirements, and no reliance on a remote API.

This result is particularly relevant to the target application. The objective is not to maximize translation quality without deployment constraints, but to achieve competitive subtitle translation while preserving privacy and enabling real-time local inference. The final model therefore represents a practical quality–efficiency trade-off rather than a direct replacement for a larger cloud model. Because both systems were evaluated independently against Google Translate, the reported win rates are anchor-relative and do not constitute a direct pairwise comparison between the final model and GPT-4o mini. Surface-form diagnostics further show a simplified-Chinese rate of 0.3% and an English echo rate of 0.0%. Character-level F1 is reported only as a secondary diagnostic and is not used as the main measure of translation quality.

5.2 Context ablation

Table 4: Context ablation against the same Google Translate anchor with short cue subset

Condition	n	W/L/T	Win rate
With context	260	128/47/85	73.1%
Without context	260	129/59/72	68.6%

Context changes the full-set point estimate by 0.6 percentage points. On the independently defined 260-example short-response subset, the with-context estimate is 4.5 points higher. The intervals in Table 4 are marginal bootstrap intervals for each condition rather than a paired interval for the difference. The 4.5-point difference is therefore descriptive and is not presented as statistically significant.

5.3 Performance by cue length

Table 5: Final-model win rate by source-cue length

Cue length	n	W/L/T	Win rate
1–3 words	171	91/20/60	82.0%
4–7 words	224	95/81/48	54.0%
8–15 words	97	42/50/5	45.7%
16+ words	8	1/7/0	12.5%

The model is strongest on short cues and falls below the anchor for cues of eight words or more. This pattern is consistent with qualitative observations that longer cues are more vulnerable to omission and incomplete meaning preservation. The 16+ word result is descriptive only because the subset contains eight examples.

5.4 Embedding calibration comparison

The comparison supports embedding calibration before full fine-tuning. However, char-F1 is only a surface metric, and the archived experiment record does not establish that calibration was the only difference between the two training runs. The result should therefore be treated as supporting evidence rather than a fully isolated ablation.

Table 6: Embedding calibration comparison on 105 aligned examples

Initialization	Average char-F1
Cold start	0.6227
Calibration followed by full SFT	0.7138
Difference	+0.0911

5.5 Preliminary latency results

Table 7: Preliminary Apple M2 Metal profiling results

Metric	151k profiling baseline	64k profiling model	Change
Average end-to-end latency	92.7 ms	56.8 ms	1.63× speedup
P95 end-to-end latency	148.2 ms	88.4 ms	1.68× speedup
Decode throughput	63.3 tokens/s	96.0 tokens/s	1.52× higher
Avg. generated output tokens	6.67	4.00	40.0% lower

The latency difference is consistent with a smaller output projection and denser Chinese tokenization. However, the generated-output token counts compare two systems and may reflect both tokenizer efficiency and differences in translation length or omission. The fixed-text characters-per-token result in Table 1 is the cleaner tokenizer-efficiency measurement.

End-to-end speedup is smaller than the decode-only estimate because prefill, tokenization, runtime overhead, and a higher English prompt-token count remain. These measurements require a reproducible rerun before final publication.

6 Discussion

6.1 Tokenizer redesign as a serving decision

Quantization and tokenizer redesign address different parts of the latency problem. Quantization reduces Transformer-block cost. A smaller vocabulary then reduces output-projection work, while subtitle-domain tokenization represents Chinese text with fewer tokens. In this workload, tokenizer design is therefore part of the serving architecture rather than only a preprocessing choice.

This interpretation remains provisional because the operation-level profiler record is incomplete. A controlled 151k/64k/32k comparison under the same model, data, and deployment settings is needed to isolate vocabulary size from output-token count and other changes.

6.2 Role of embedding calibration

Embedding averaging avoids random initialization but does not guarantee alignment with the pretrained semantic space. Direct full fine-tuning asks the model to adapt to both a new embedding distribution and a new task format. The calibration stage separates these problems and is supported by the archived comparison.

An early run also showed that calibration must cover enough of the data distribution. Ordered subtitle data and a very short calibration stage produced a train/evaluation loss inversion when training reached later, less familiar content. Detailed diagnostics are provided in Appendix E.

6.3 Quality trade-offs

The observed advantage is concentrated in subtitle style and short-cue translation rather than broad performance across cue lengths. The final model often produces concise, natural Taiwan subtitle phrasing on short cues. Its main weakness is over-compression on longer cues, where missing details and semantic errors become more frequent.

Qualitative inspection also suggests unstable handling of names, transliterations, and some region-specific lexical choices. These error categories have not yet been systematically counted.

7 Limitations and Future Work

This study has several limitations. First, the Apple M2 latency experiment does not yet have a complete reproducibility record and was conducted with a profiling model rather than the final evaluated model. The vocabulary-projection interpretation is also based on incomplete profiling evidence because an operation-level trace was not retained. Second, the pairwise evaluation still requires human auditing, and the exact GPT-4o snapshot used for judging was not recorded. In addition, the final model and GPT-4o mini were evaluated independently against the same Google Translate anchor rather than compared directly across the full evaluation set. The automatically scenario-tagged benchmark has not yet been evaluated with a complete aligned scenario-stratified protocol, and important error categories, including omission, named-entity translation, and Taiwan-specific lexical choice, have not been manually quantified.

Future work should therefore prioritize rerunning the latency and profiling experiments with complete logs and the final evaluated model, conducting controlled comparisons across tokenizer sizes, and validating the pairwise judgments through human review. It should also include a full scenario-stratified evaluation of the tagged test set and the construction of a manually verified error taxonomy.

8 Conclusion

This work presents a workload-driven optimization path for on-device real-time subtitle translation. Short inputs, short outputs, and batch-size-one inference make fixed decode costs more relevant than optimizations designed for long-context or high-throughput serving. Preliminary profiling suggests that vocabulary projection becomes increasingly important after quantization reduces Transformer-block cost.

A 64k-vocabulary subtitle tokenizer reduces the output-projection dimension, increases Chinese token density, and decreases model size. Embedding migration and calibration provide a practical adaptation path before full supervised fine-tuning.

The LocalSubs achieves a 59.2% tie-excluded win rate against Google Translate on 500 aligned examples. It performs best on short cues and remains weaker on longer inputs. A separate reduced-vocabulary profiling model shows a preliminary 1.63× average-latency speedup on Apple M2 Metal, but this result requires a controlled and reproducible rerun.

A Task Format and Structural Tokens

The training and inference input format is:

```
CTX:
<0--3 previous English subtitle cues>

CUR:
<current English subtitle cue>
```

The output contains only the translation of `CUR`. Context must not be copied or translated into the output.

The tokenizer includes four structural tokens corresponding to the user role, assistant role, `\nCTX:\n`, and `\nCUR:\n`. The latter two delimit context and current-cue content. Their embeddings are initialized from the average embeddings of their original sub-token decompositions.

B Dataset Construction

The source pipeline begins with 14,237,823 English–Chinese subtitle pairs. The final SFT dataset contains 233,088 examples.

The filtering pipeline includes:

1. removal of empty records, extreme lengths, abnormal length ratios, OCR corruption, release-group advertisements, and lyric metadata;
2. Traditional Chinese filtering, including rejection of targets containing simplified-only characters;
3. LLM-assisted subtitle-pair quality filtering for semantic correctness, natural Taiwan usage, and subtitle readability; and
4. length rebalancing followed by shuffling before the training/evaluation split.

The observed acceptance rate of the LLM-assisted filtering stage was approximately 30%.

Table 8: Cue-length distribution of the final SFT dataset

Source-cue length	Share
1–3 words	34%
4–7 words	34%
8–15 words	23%
16+ words	9%

The initial cleaned subtitle SFT dataset is retained only for analysis of data ordering, evaluation splitting, and training instability. The final model uses the quality-filtered, length-rebalanced subtitle SFT dataset.

C Automatic Scenario Annotation

The 4,246-example OpenSubtitles2024 test set was automatically assigned multi-label scenario tags for future stratified evaluation. Annotation does not constitute completed model evaluation.

The context-ablation subset `short_response` is defined independently from the scenario tag `S1_short_response`. They must not be merged without row-level verification.

D Tokenizer and Embedding Migration Details

The four structural tokens are initialized separately. The parameter reduction is concentrated in the tied embedding and output-projection matrix.

Table 9: Automatic scenario-tag distribution

Scenario	Definition	Count	Share
S1 short cue	Brief response, interjection, or cue with at most three source words	2,609	61.4%
S2 context-dependent cue	May require preceding cues for disambiguation	940	22.1%
S3 tone-continuity cue	Context establishes a tone or interaction style	22	0.5%
S4 medium-to-long cue	Contains at least eight source words	1,232	29.0%
S5 Taiwan-specific lexical choice	May involve region-dependent Chinese wording	3	0.1%
S6 named-entity cue	Contains a name, place, title, organization, or other entity	631	14.9%

Table 10: Tokenizer configuration

Item	Setting
Tokenizer family	ByteLevel BPE
Training text	English and Taiwan Traditional Chinese subtitles
Target vocabulary	64,000
Base tokenizer size	64,020
Size after structural tokens	64,024
Inherited special tokens	Qwen-style special tokens

Table 11: Embedding initialization coverage for the 64k-token base vocabulary

Method	Tokens	Share
Direct copy	32,340	50.5%
Average of original sub-token embeddings	31,680	49.5%
Mean fallback	0	0.0%

Table 12: Model size before and after vocabulary replacement

Item	151k model	64k-vocabulary base model
Vocabulary size	151,936	64,024
Hidden size	1024	1024
Parameters	∼596M	∼506M
BF16 size	1.11 GB	0.94 GB

Table 13: Embedding calibration configuration

Item	Setting
Frozen parameters	Transformer layers
Trainable parameters	Embeddings, output head, and RMSNorm
Trainable parameter count	65.6M of 506M
Learning rate	5×10^{-4}
Batch size	16
Gradient accumulation	2
Training duration	0.5 epoch
Hardware	3 × RTX 3090

E Training Configuration and Diagnostic Run

E.1 Embedding calibration

E.2 Early loss inversion

The initial cleaned dataset and a short 0.07-epoch calibration stage produced increasing training loss while evaluation loss decreased.

Table 14: Loss inversion in the early training run

Epoch	Evaluation loss	Training loss	Gap
0.000	–	2.928	–
0.027	2.334	2.410	+0.076
0.054	2.279	2.667	+0.388
0.081	4.001	2.795	+0.544

A likely cause is ordered subtitle data combined with insufficient calibration coverage. Later portions of the data contained less familiar names, transliterations, and domain terms. The calibration stage was extended to 0.5 epoch, and the final dataset was shuffled before splitting.

F Evaluation Details

The judge prioritizes semantic correctness, Taiwan Traditional Chinese usage, naturalness, subtitle concision, and current-cue-only output. Simplified Chinese is treated as a hard failure. Candidate order is randomized to reduce position bias.

Google Translate is a fixed comparison anchor, not ground truth. The OpenSubtitles translation remains the dataset reference and is used only for reference-based diagnostics.

Character-level F1 is useful for internal tracking but unreliable as a standalone translation-quality metric. Valid paraphrases may have low overlap, while an incorrect translation may share many characters with a noisy reference.

The evaluation artifacts preserve sample-level judgments and A/B assignments. The exact GPT-4o snapshot, API date, and A/B randomization seed were not retained and should be recorded in future reruns.

F.1 Bootstrap confidence intervals

Win-rate confidence intervals are computed by `eval/summarize_winrate.py`. The implementation uses 10,000 bootstrap resamples and random seed 13 by default. The resampling unit is one judged row. For each replicate, n rows are sampled with replacement from the original n rows, and win/loss/tie counts are recomputed from the sampled rows. Rows labeled `model_wins`, `anchor_wins`, and `tie` map to W, L, and T, respectively. Ties remain in the resampled data, while the statistic is recomputed as

$$\widehat{p}_{\text{win}} = \frac{W}{W + L}.$$

The implementation sorts the 10,000 bootstrap statistics and selects the entries at indices $\lfloor 0.025B \rfloor$ and $\lfloor 0.975B \rfloor$, where $B = 10{,}000$. This is a custom order-statistic implementation of a percentile interval; it does not use SciPy quantile interpolation. The reported intervals describe each condition separately. A paired confidence interval for the difference between with-context and without-context conditions would require resampling aligned example pairs and recomputing the difference within each replicate.

“‘latex

G Latency Benchmark Details

Table 15: Full preliminary latency summary

System	Average	P50	P95
64k-vocabulary profiling model, `Q5_K_M`	56.8 ms	53.2 ms	88.4 ms
151k-vocabulary profiling baseline, `Q5_K_M`	92.7 ms	87.5 ms	148.2 ms
Speedup	1.63×	1.64×	1.68×

Table 16: Preliminary throughput summary

Workload	64k model	151k baseline	Speedup
Prefill pp64	1586.6 tokens/s	1194.7 tokens/s	1.33×
Decode tg32	96.0 tokens/s	63.3 tokens/s	1.52×

Table 17: Average token counts in the recorded profiling run

Segment	64k tokenizer	151k tokenizer	Δ
English prompt	23.83	25.83	−7.7%
Generated Chinese output	4.00	6.67	−40.0%

The 64k tokenizer produces 7.7% fewer tokens for the English prompts in the recorded profiling set. The 64k-vocabulary model also generates 40.0% fewer Chinese output tokens than the 151k-vocabulary baseline. The prompt-token comparison uses the same source text and therefore provides direct evidence of improved input tokenization efficiency. In contrast, the generated-output comparison may reflect both tokenizer compression and differences in translation content or generation behavior.

The latency improvement is supported by three directly measured observations: the 64k-vocabulary model achieves 1.52× higher decode throughput, uses fewer prompt tokens, and generates fewer output tokens in the recorded run. Together, these factors are consistent with the

measured 1.63× improvement in average end-to-end latency. A precise attribution of the gain to prefill, decoding, tokenization, and runtime overhead would require stage-level timing measurements.

A publication-quality report should record the llama.cpp commit, GGUF filenames and checksums, Apple M2 model and memory configuration, benchmark command, input set, decoding parameters, warm-up count, measured run count, and raw per-run log.

References

[1] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016. https://aclanthology.org/P16-1162/. https://doi.org/10.18653/v1/P16-1162.

[2] Yingfeng Luo, Ziqiang Xu, Yuxuan Ouyang, Murun Yang, Dingyang Lin, Kaiyan Chang, Tong Zheng, Bei Li, Peinan Feng, Quan Du, Tong Xiao, and Jingbo Zhu. NiuTrans.LMT: Toward Inclusive and Scalable Multilingual Machine Translation with LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25151–25179, 2026. https://aclanthology.org/2026.acl-long.1153/. https://doi.org/10.18653/v1/2026.acl-long.1153.

[3] NiuTrans. NiuTrans/LMT-60-0.6B. Hugging Face model card. https://huggingface.co/NiuTrans/LMT-60-0.6B, accessed July 11, 2026.

[4] Joerg Tiedemann and Hengyu Luo. OpenSubtitles2024: A Massively Parallel Dataset of Movie Subtitles for MT Development and Evaluation. In *Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)*, pages 8897–8907, 2026. https://doi.org/10.63317/4ivg578ub2ob. Dataset available at https://huggingface.co/datasets/Helsinki-NLP/OpenSubtitles2024, accessed July 11, 2026.

[5] An Yang et al. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388, 2025. https://arxiv.org/abs/2505.09388. https://doi.org/10.48550/arXiv.2505.09388.

[6] Lianmin Zheng et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623, 2023. https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

[7] ggml-org. llama.cpp: LLM inference in C/C++. GitHub repository. https://github.com/ggml-org/llama.cpp, accessed July 11, 2026.

[8] OpenAI. GPT-4o Model. OpenAI API documentation. https://developers.openai.com/api/docs/models/gpt-4o, accessed July 11, 2026.

[9] OpenAI. GPT-4o mini Model. OpenAI API documentation. https://developers.openai.com/api/docs/models/gpt-4o-mini, accessed July 11, 2026.