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 [7], 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 an OpenSubtitles2024 test set, 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. In a separate preliminary Apple M2 Metal profiling run, LocalSubs shows a 1.63× speedup over a 151k-vocabulary baseline. The code is available on https://github.com/aiden1020/localsubs.

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 inference 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 [7]. 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 on Apple M2 Metal. 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 Related Work

Subtitle translation and context. Subtitle translation differs from general sentence-level MT because dialogue cues are short, temporally constrained, and often ambiguous in isolation. Matusov et al. [2] combined subtitle-domain adaptation with preceding-sentence context and learned segmentation, while Vincent et al. [3] found that contextual information reduced context-related post-editing errors in a professional subtitling workflow. LocalSubs shares the motivation for contextual translation but targets a narrower interactive workload: one current cue, at most three preceding cues, batch-size-one inference, and output constrained to the current cue. Its primary optimization target is therefore the quality–latency trade-off on consumer hardware rather than document-level context modeling or professional subtitle segmentation.

Subword vocabularies and vocabulary adaptation. Subword tokenization addresses open-vocabulary translation by representing rare words compositionally [1], but vocabulary design also determines embedding size, output-projection cost, and the number of decoding steps. Recent analysis shows that simply trimming rare BPE entries does not consistently preserve NMT quality and can cause substantial degradation [4]. LocalSubs does not apply threshold trimming to the original tokenizer. It trains a new subtitle-domain vocabulary, migrates every new token embedding by direct copy or original-subtoken averaging, and then adapts the changed representation space through calibration and full SFT. This connects vocabulary reduction to the measured token distribution and batch-size-one decode workload rather than treating model size alone as the objective.

Pairwise and anchor-based evaluation. Pairwise evaluation preserves the fact that systems are compared on the same examples and can reveal distinctions obscured by independently averaged scores [5]. LLM judges make such evaluation scalable, but their judgments may exhibit position, style, and self-preference biases [11]; anchor choice can also materially affect the reliability and statistical power of anchor-based rankings [6]. Accordingly, this report randomizes candidate order, retains ties, and uses the same Google Translate anchor across systems. The resulting win rates support anchor-relative comparisons under one judge protocol, not a direct head-to-head preference estimate between LocalSubs and GPT-4o mini.

On-device browser translation. Client-side systems such as Bergamot demonstrate that machine translation can run locally from a browser while keeping source text on the user's device [14]. LocalSubs adopts a different implementation boundary: a Chrome extension captures and overlays subtitle cues, while Chrome Native Messaging forwards requests to a native host backed by a local inference engine. This extension–native-host split is deployment infrastructure for the optimized model, not a methodological contribution.

3 Task and Design Goals

3.1 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.

3.2 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.

4 Dataset Construction

Dataset quality is central to subtitle SFT. Raw subtitle corpora contain alignment errors, OCR noise, advertisements, simplified Chinese, and translations unsuitable for Taiwan subtitles. We construct a quality-filtered dataset from 14,237,823 English–Chinese subtitle pairs. The final dataset contains 233,088 examples.

4.1 Rule-based cleaning

We remove empty or malformed records, extreme lengths, abnormal source–target length ratios, OCR artifacts, release-group advertisements, lyric metadata, duplicated pairs, and targets dominated by unintended languages.

4.2 Traditional Chinese filtering

Targets containing simplified-only characters or excessive non-Chinese content are rejected. This stage reduces simplified-Chinese leakage but does not determine whether the wording is natural for Taiwan.

4.3 LLM-assisted quality filtering

The remaining pairs are evaluated using a Qwen3-family model [10]. The filter considers semantic consistency, natural Taiwan Traditional Chinese usage, and concise subtitle style. Approximately 30% of the evaluated pairs are retained.

4.4 Context-aware example construction

Each example contains up to three preceding English cues and one current cue:

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

CUR:
<current English subtitle cue>
```

The target contains only the translation of `CUR`. Context supports disambiguation and tone continuity but is not translated into the output.

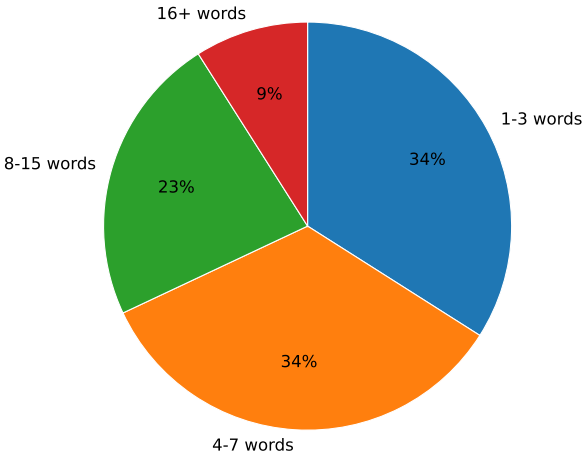

Figure 1: Source-cue length distribution of the final SFT dataset.

4.5 Length rebalancing

The cleaned corpus is dominated by short cues. We rebalance the dataset by source-cue length to increase the coverage of longer inputs.

The dataset is shuffled before the training/validation split to reduce correlations caused by movie, episode, or source-file ordering.

Table 1: Summary of the dataset construction pipeline.

Item	Setting
Initial pairs	14,237,823
Final SFT examples	233,088
Translation direction	English to Taiwan Traditional Chinese
Context	Up to three preceding English cues
Target	Translation of the current cue only
Processing	Rule-based cleaning, Traditional Chinese filtering, LLM-assisted filtering, length rebalancing, and shuffling

5 Method

5.1 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 an operation-level trace was not retained, this report treats vocabulary projection as a supported hypothesis rather than a fully isolated bottleneck.

5.2 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 2: 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.

5.3 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 training partition of the subtitle SFT corpus.

6 Deployment Architecture

LocalSubs is designed to translate streaming subtitles entirely on the user's device while remaining responsive to rapidly changing cues. Its deployment architecture therefore separates subtitle interaction in the browser from model execution in a native process. As shown in Figure 2, a translation request passes through four functional stages: subtitle capture, extension-level coordination, browser-to-native communication, and local inference. The translation is returned

along the same path and rendered as an overlay on the streaming page. This separation keeps page-facing code lightweight, confines native execution to a privileged boundary, and allows the inference runtime to be changed without modifying subtitle capture and presentation.

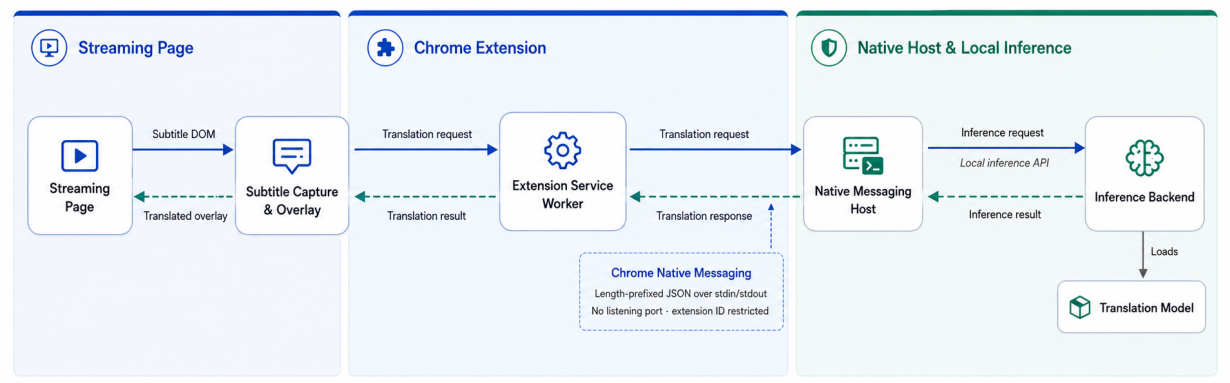

Figure 2: End-to-end LocalSubs translation workflow: subtitle cues are captured from the streaming page, routed through the Chrome extension and Native Messaging host to the local inference backend, and returned for overlay rendering. Solid, dashed, and gray arrows indicate requests, responses, and model loading, respectively.

6.1 Browser integration

The content script connects LocalSubs to the subtitle interface exposed by the streaming page. It observes subtitle-DOM changes, normalizes each detected cue, and retains up to three preceding English cues as translation context. Only the current cue is translated; the preceding cues provide disambiguating context and help preserve conversational tone. This browser-side context construction matches the input format used during fine-tuning and avoids requiring the native backend to maintain page-specific state.

Subtitle playback is asynchronous, so a result may arrive after the page has already advanced to a newer cue. Before updating the translated overlay, the content script checks that the response still corresponds to the active cue. This suppresses stale results and preserves temporal consistency between the source subtitle and its translation.

The extension service worker coordinates communication beyond the page. It associates requests with their responses and forwards them to the native host, while the content script remains limited to subtitle capture and display. This division establishes a clear privilege boundary: page-facing code cannot invoke the local inference runtime directly.

6.2 Browser-to-native boundary

Chrome Native Messaging provides the controlled bridge between the extension and local model execution. Only the extension service worker can establish this connection, and the registered host restricts access to the LocalSubs extension origin. The streaming page therefore has no direct route to the native process.

This design was chosen over a browser-accessible local service because it does not require a listening network endpoint. It reduces the exposed interface of the native component and preserves Chrome's extension-origin checks at the boundary. Native Messaging also decouples the browser protocol from the inference API, enabling the native host to validate and mediate requests before they reach the model runtime.

6.3 Local inference boundary

The native host separates extension communication from inference execution. It validates incoming requests, exposes model readiness to the extension, and manages the inference backend as a replaceable component. Consequently, the browser integration depends on a stable translation interface rather than on a particular model server or acceleration framework.

The evaluated deployment uses `llama-server` with the GGUF Q5_K_M model and Metal acceleration on Apple Silicon. The model is loaded once and reused across subtitle cues, avoiding repeated initialization on the latency-critical translation path. Communication with the backend remains local to the device, while process ownership is assigned to the native host so that the inference runtime follows the extension session lifecycle.

Across the complete path, subtitle text remains within the streaming page, Chrome extension, native host, and local inference process. No remote translation API participates in per-cue inference. This property is central to the deployment rather than incidental to the implementation: it provides a privacy-preserving execution path while allowing latency to be measured under the same on-device conditions in which LocalSubs is intended to operate.

7 Experimental Setup

7.1 Model and Training Configuration

We use NiuTrans/LMT-60-0.6B as the base model, a compact multilingual translation model [7, 8]. LocalSubs is fine-tuned on the training partition of the 233,088-example subtitle SFT corpus described in Section 4.

Table 3: Core experimental configuration.

Item	Setting
Base model	NiuTrans/LMT-60-0.6B
Translation direction	English to Taiwan Traditional Chinese
SFT corpus before splitting	233,088 examples
Input format	Up to three preceding cues and one current cue
Original vocabulary	151,936 tokens
Adapted vocabulary	64,024 tokens
Evaluation set	Fixed 500-example subset of OpenSubtitles2024

7.2 Evaluation data and protocol

The OpenSubtitles2024 benchmark contains bilingual subtitle alignments held out for machine-translation development and evaluation [9]. 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 [11]. 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 and verbatim judge prompt were not retained; the experiment record identifies the model alias and judging criteria. This is a reproducibility limitation. GPT-4o mini is used as a cloud baseline [15, 16].

7.3 Deployment benchmark

The deployment benchmark uses the inference backend described in Section 6: llama.cpp with GGUF Q5_K_M quantization and Apple Metal acceleration. llama.cpp provides local inference, integer quantization, and optimized Apple Silicon support [12]. The reported latency covers the measured inference request, including tokenization, prefill, and autoregressive decoding. It excludes browser-side subtitle detection, Native Messaging transport, and overlay rendering, and is therefore reported as inference latency rather than user-facing end-to-end latency. The recorded summary uses the 64k-vocabulary LocalSubs model.

8 Results

8.1 Pairwise translation quality

Table 4: Pairwise preference results against Google Translate

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

LocalSubs achieves a 59.2% tie-excluded win rate against Google Translate, compared with 64.6% for GPT-4o mini under the same Google-anchored evaluation protocol. The resulting 5.4-percentage-point gap places the local 0.6B model close to the cloud baseline on this shared-anchor metric despite substantially stricter deployment constraints, including on-device execution, batch-size-one inference, low-latency requirements, and no reliance on a remote API. Because both systems were compared separately with the same anchor, this gap is an anchor-relative comparison rather than a direct estimate of the head-to-head win rate between LocalSubs and GPT-4o mini.

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. LocalSubs therefore represents a practical quality–efficiency trade-off rather than a direct replacement for a larger cloud model. 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.

8.2 Context ablation

Table 5: Context ablation against the same Google Translate anchor on the short-response subset ($n = 260$)

Condition	W/L/T	Win rate
w/o context	129/59/72	68.6%
w/ context	128/47/85	**73.1%**

On the independently defined 260-example short-response subset, the with-context estimate is 4.5 percentage points higher. This difference is descriptive because a paired confidence interval for the difference was not computed, and it is not presented as statistically significant.

8.3 Performance by cue length

Table 6: LocalSubs 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.

8.4 Embedding calibration comparison

Table 7: 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

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.

8.5 Latency profiling results

Table 8: Apple M2 Metal latency profiling results

Metric	151k baseline	LocalSubs	Change
Average inference latency	92.7 ms	56.8 ms	1.63× speedup
P95 inference 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 2 is the cleaner tokenizer-efficiency measurement.

The average inference latency of 56.8 ms and P95 latency of 88.4 ms both fall below the commonly cited 100-ms threshold at which an interactive system can be perceived as responding instantaneously [13]. These measurements suggest that model inference introduces little perceptible waiting relative to this general HCI guideline. They do not, however, establish that translated subtitles are imperceptibly delayed during viewing, because the benchmark excludes the remaining browser-to-display path and no formal user study was conducted.

The inference-latency speedup exceeds the decode-throughput gain, which is consistent with the 64k system using fewer prompt and generated tokens in the recorded run. Prefill, tokenization, and runtime overhead also contribute to inference latency, so the aggregate speedup cannot be attributed to decode throughput alone.

9 Discussion

9.1 Mechanisms behind the latency improvement

The reduced-vocabulary system changes three quantities that can affect latency. First, replacing 151,936 output classes with 64,024 reduces the width of the per-step vocabulary projection by 57.9%. This change should reduce work at every decode step, while leaving the hidden size and Transformer depth unchanged. Second, the subtitle tokenizer encodes fixed Chinese text more densely, so an equivalent translation can require fewer decode steps. Third, the vocabulary replacement removes parameters from the tied embedding and output-projection matrix, reducing the model from approximately 596M to 506M parameters and potentially changing memory traffic. Quantization acts on a different part of this mechanism: by reducing the relative cost of the Transformer blocks, it can make the remaining projection and token-count costs more visible in the total decode path.

The measurements are consistent with a combination of these mechanisms rather than with projection reduction alone. Table 8 shows a 1.52× decode-throughput gain, whereas average inference latency improves by 1.63×. The larger inference-latency gain is compatible with the recorded reductions in prompt and generated-token counts, because fewer tokens reduce work outside the per-token throughput measurement. However, the generated outputs are not held fixed: a shorter output may reflect better tokenization, a more concise translation, or an omission. The profiling run also compares separately adapted systems rather than changing only the projection width. The current evidence therefore establishes a system-level difference consistent with the proposed mechanisms, but it does not identify how much of the gain comes from projection width, token density, parameter reduction, or changed generation behavior. A controlled vocabulary-size comparison with fixed prompts, decode steps, runtime settings, and stage-level timing is required for causal attribution.

9.2 Why embedding calibration may help

Tokenizer replacement creates a distribution shift at both ends of the model. Directly copied embeddings preserve tokens shared by the old and new vocabularies, but embeddings initialized by averaging old sub-tokens are only approximate representations of the new token strings. At the same time, the tied output head must learn to assign probability mass over a different set of units. If full fine-tuning begins immediately, the Transformer must adapt simultaneously to this changed interface and to the subtitle translation objective.

Embedding calibration reduces this coupling by freezing the Transformer layers while updating the embeddings, output head, and RMSNorm parameters. A plausible mechanism is that the calibration stage first aligns the new token interface with representations that the pretrained Transformer already produces; full SFT can then focus more directly on task and domain adaptation. The higher character-F1 in Table 7 is consistent with this explanation, but it is not an isolated causal estimate because the archived runs may differ in more than initialization schedule

and character-F1 measures surface overlap rather than translation preference. The early loss inversion in Appendix E further suggests that calibration depends on data coverage: with ordered data and only 0.081 epoch, later names and transliterations exposed regions of the token distribution that the interface had not yet adapted to. Extending calibration and shuffling the data address this coverage problem together, so their individual effects remain unresolved.

9.3 Cue length, context, and over-compression

The cue-length pattern in Table 6 suggests an interaction between the target workload, training distribution, and generation behavior. Short cues match the dominant application workload and make up 68% of the final SFT data in the 1–7-word bins. They also require fewer semantic relations to be preserved, allowing the model to benefit from preceding context without producing long outputs. This provides a plausible explanation for why context yields a larger descriptive gain on the short-response subset while the model remains strongest on short cues overall.

Longer cues place a different demand on the system. They contain more propositions and details that must survive a concise subtitle rendering, but examples of 16 or more words constitute only 9% of the training set. The model's preference for short outputs may be beneficial stylistically on brief cues yet become over-compression when the source contains more information. This creates an important ambiguity in the latency results: the 40% reduction in generated tokens can represent tokenizer efficiency, appropriate concision, or missing content. The decline beyond eight source words and the qualitative omission errors make all three explanations plausible.

The current aggregate metrics cannot separate these mechanisms. A targeted analysis should compare source length, reference length, generated character count, tokenizer-normalized output length, and manually labeled omissions on aligned examples. Context should also be evaluated by ambiguity type rather than only by cue length, because preceding cues are expected to help pronouns, ellipsis, speaker intent, and tone continuity, but not necessarily the content-capacity problem of long current cues. Such an analysis would determine whether additional long-cue training, an adequacy-oriented objective, or length-aware decoding can improve meaning preservation without giving up the short-cue latency advantage.

10 Limitations and Future Work

This study has several limitations. First, translation quality declines as cue length increases, and the aggregate pairwise preference score does not directly measure whether all source information is preserved. The qualitative analysis identifies omission and over-compression errors, but their frequency and severity have not yet been manually quantified. Second, the pairwise evaluation still requires human auditing, and the exact GPT-4o snapshot and verbatim judge prompt were not recorded. In addition, LocalSubs 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 other important error categories, including named-entity translation and Taiwan-specific lexical choice, have not been manually quantified. Finally, the vocabulary-projection interpretation remains based on incomplete profiling evidence because an operation-level trace was not retained.

Future work should therefore prioritize adequacy-focused human evaluation of long cues, including manual measurement of omissions and over-compression, and validate the pairwise judgments through human review. It should also include controlled profiling across tokenizer sizes with an operation-level trace, a full scenario-stratified evaluation of the tagged test set, and the construction of a manually verified error taxonomy.

11 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. Profiling evidence 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.

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. In a separate Apple M2 profiling run, LocalSubs shows a 1.63× average-latency speedup over the recorded baseline.

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/validation split.

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

The reported 233,088 examples are the total SFT corpus before the training/validation split. The initial cleaned subtitle corpus is retained only for analysis of data ordering, split construction, and training instability. LocalSubs uses the training partition of the quality-filtered, length-rebalanced corpus.

Table 9: 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%

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.

Table 10: 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%

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

Table 11: 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

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

Table 12: 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 13: 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

E Training Configuration and Diagnostic Run

E.1 Embedding calibration

Embedding calibration is the first adaptation stage after vocabulary replacement. Its purpose is to align the migrated token embeddings and tied output interface with the representations produced by the pretrained Transformer before all model parameters are updated. The Transformer layers remain frozen during this stage, while the embeddings, output head, and RMSNorm parameters are trainable.

Table 14: 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

This stage is not treated as an independent translation model. Its output serves as the initialization for full supervised fine-tuning on the subtitle task, during which the complete model is updated. Separating the two stages reduces the need for the Transformer to adapt simultaneously to both a new token interface and the subtitle-domain objective. The calibration comparison in Table 7 provides supporting evidence for this procedure, but it is not a controlled causal ablation because the archived runs may differ in more than the initialization schedule.

E.2 Early loss inversion

The initial cleaned dataset and a short 0.081-epoch calibration run produced an unstable loss pattern: evaluation loss initially decreased and then rose sharply, while training loss increased across the recorded checkpoints.

Table 15: 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 SFT corpus was shuffled before the training/validation split.

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, A/B randomization seed, and verbatim API prompt were not retained and should be recorded in future reruns.

F.1 LLM-as-a-Judge Prompt

The following reconstructed prompt template documents the recorded pairwise judging criteria; it is not claimed to reproduce the original API prompt verbatim. Translation A and Translation B are randomly assigned before the prompt is constructed.

```
SYSTEM
You are an expert bilingual subtitle evaluator for English-to-
Traditional-Chinese translation as used in Taiwan.

Compare Translation A and Translation B for the CURRENT subtitle cue.
Use the preceding CONTEXT only to resolve ambiguity. Judge the
translations using these criteria, in order:

1. Semantic correctness and preservation of all source information.
2. Correct Traditional Chinese usage for Taiwan.
3. Naturalness and fluency.
4. Concision and readability as a subtitle.
5. Translation of the current cue only; context must not be copied
   or translated into the output.

Any candidate containing Simplified Chinese is a hard failure. Do not
prefer a candidate merely because it is more literal, longer, or shown
first. Select TIE only when neither translation is meaningfully better.
```

```
The "winner" value must be "A", "B", or "TIE".
Return only valid JSON, for example:
{"winner":"A","reason":"brief justification"}

USER
CONTEXT:
{up_to_three_preceding_english_cues}

CURRENT:
{current_english_cue}

TRANSLATION A:
{translation_a}

TRANSLATION B:
{translation_b}
```

G Inference Latency Benchmark Details

Table 16: Full inference-latency profiling summary

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

Table 17: Throughput profiling summary

Workload	LocalSubs	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 18: Average token counts in the recorded profiling run

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

The LocalSubs tokenizer produces 7.7% fewer tokens for the English prompts in the recorded profiling set. LocalSubs 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: LocalSubs 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 inference latency. A precise attribution of the gain to prefill, decoding, tokenization, and runtime overhead would require stage-level timing measurements.

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] Evgeny Matusov, Patrick Wilken, and Yota Georgakopoulou. Customizing Neural Machine Translation for Subtitling. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 82–93, 2019. https://aclanthology.org/W19-5209/. https://doi.org/10.18653/v1/W19-5209.

[3] Sebastian Vincent, Charlotte Prescott, Chris Bayliss, Chris Oakley, and Carolina Scarton. A Case Study on Contextual Machine Translation in a Professional Scenario of Subtitling. In *Proceedings of the 25th Annual Conference of the European Association for Machine Translation (Volume 1)*, pages 561–572, 2024. https://aclanthology.org/2024.eamt-1.46/.

[4] Marco Cognetta, Tatsuya Hiraoka, Rico Sennrich, Yuval Pinter, and Naoaki Okazaki. An Analysis of BPE Vocabulary Trimming in Neural Machine Translation. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 48–50, 2024. https://aclanthology.org/2024.insights-1.7/. https://doi.org/10.18653/v1/2024.insights-1.7.

[5] Maxime Peyrard, Wei Zhao, Steffen Eger, and Robert West. Better than Average: Paired Evaluation of NLP Systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2301–2315, 2021. https://aclanthology.org/2021.acl-long.179/. https://doi.org/10.18653/v1/2021.acl-long.179.

[6] Shachar Don-Yehiya, Asaf Yehudai, Leshem Choshen, and Omri Abend. Mediocrity Is the Key for LLM-as-a-Judge Anchor Selection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15491–15513, 2026. https://aclanthology.org/2026.acl-long.706/. https://doi.org/10.18653/v1/2026.acl-long.706.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

[14] browsermt. Bergamot Translator: Optimized Machine Translation on Consumer-Grade Devices. GitHub repository. https://github.com/browsermt/bergamot-translator, accessed July 13, 2026.

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

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