Comparing LLM-Based Translation Approaches for Extremely Low-Resource Languages

Jared Coleman[1], **Ruben Rosales**[2], **Kira Toal**[2], **Diego Cuadros**[1]
Nicholas Leeds[1], **Bhaskar Krishnamachari**[3], **Khalil Iskarous**[3]
[1]Loyola Marymount University, [2]Independent Researcher, [3]University of Southern California
jared.coleman@lmu.edu, ruben.rosales@ieee.org, kira.toal@lmu.edu, dcuadros@lion.lmu.edu
nleeds@lion.lmu.edu, bkrishna@usc.edu, kiskarou@usc.edu

Abstract

We present a comprehensive evaluation and extension of the LLM-Assisted Rule-Based Machine Translation (LLM-RBMT) paradigm, an approach that combines the strengths of rule-based methods and Large Language Models (LLMs) to support translation in no-resource settings. We present a robust new implementation (the Pipeline Translator) that generalizes the LLM-RBMT approach and enables flexible adaptation to novel constructions. We benchmark it against four alternatives (Builder, Instructions, RAG, and Fine-tuned translators) on a curated dataset of 150 English sentences, and compare them across translation quality and runtime. The Pipeline Translator consistently achieves the best overall performance. The LLM-RBMT methods (Pipeline and Builder) also offer an important advantage: they naturally align with evaluation strategies that prioritize grammaticality and semantic fidelity over surface-form overlap, which is critical for endangered languages where mistranslation carries high risk.

1 Introduction

Recent advances in Large Language Models (LLMs) like OpenAI's GPT series have led to dramatic improvements in machine translation. However, these models still perform poorly on languages with little or no representation in their training data (Chowdhery et al., 2022; Robinson et al., 2023). This is a critical limitation, especially for endangered language revitalization efforts where reliable tools are urgently needed but parallel corpora are unavailable.

To address this, recent work introduced a new paradigm called **LLM-Assisted Rule-Based Machine Translation (LLM-RBMT)** (Coleman et al., 2024), which enables translation into a no-resource language using zero weight training or fine-tuning. Instead, it combines linguistic knowledge with LLM capabilities to guide sentence construction via grammar-aware tooling. This approach was successfully used to develop the first English-to-Owens Valley Paiute (OVP, a critically endangered Indigenous language) translator. Unlike well-resourced languages (and even many other low-resource settings) OVP has *no* publicly available parallel corpora and only a handful of fluent speakers. Figure 1 illustrates the LLM-RBMT pipeline: the English input is decomposed into simpler, translatable sentences using an LLM-based segmenting tool. Each simple sentence is translated into OVP using sentence-building tools, and then translated back into English to evaluate semantic fidelity. This process guarantees grammatical correctness while maintaining flexibility through LLM support.

In this work, we significantly expand the capabilities of the original LLM-RBMT system. We redesign the Pipeline Translator to support more complex sentence structures, introduce a more modular architecture, and improve transparency through structured validation. These enhancements enable broader grammatical coverage while preserving the core strengths of the LLM-RBMT paradigm. We also provide the first systematic evaluation of this framework by benchmarking the Pipeline translator against four alternative approaches: a more autonomous LLM-RBMT system (Builder), a prompt-only translator that relies solely on in-context instructions (Instructions), a Retrieval-Augmented Generation approach (RAG), and a fine-tuned LLM trained on a small parallel corpus (Fine-tuned). Our results show that the Pipeline translator achieves the highest overall translation quality, though at greater implementation cost. The Builder and Instructions approaches offer more lightweight and flexible solutions, trading off some accuracy for ease of use. In contrast, both the Fine-tuned and RAG methods consistently underperform, suggesting that in extremely low-resource scenarios, structured rule-based strategies may outperform more data-driven ones.

Proceedings for the Ninth Workshop on Technologies for Machine Translation of Low Resource Languages (LoResMT 2026), pages 49–68

The techniques proposed in this paper are designed for endangered languages like OVP, where the constraints differ fundamentally from mainstream MT research and even from settings typically described as "low-resource." Evaluation datasets must be constructed through careful, time-intensive collaboration with community members, and the pool of speakers available to verify translations is extremely limited. As a result, evaluation scale is necessarily much smaller than in well-resourced settings, and smaller still than in many "low-resource" scenarios where hundreds or thousands of parallel sentences may still exist. Our work demonstrates that rigorous, comparative MT research is both possible and valuable under these extreme constraints, providing a replicable framework for other endangered language efforts.

To our knowledge, this is the first work to rigorously evaluate the LLM-RBMT machine translation paradigm. We also introduce an evaluation framework aligned with the unique needs of endangered language communities, where grammatical correctness and semantic fidelity is often more important than literal, word-for-word accuracy. In settings where fluent speakers are scarce, even minor mistranslations or ungrammatical outputs can erode trust and misinformation can quickly propagate, undermining revitalization efforts. Our evaluation strategy prioritizes grammaticality and semantic fidelity over surface-level lexical overlap, rewarding translations that preserve intended meaning even if they involve simplification or restructuring.

A key advantage of the LLM-RBMT paradigm is that it guarantees grammatically correct output, even when vocabulary gaps require placeholder substitutions. This is critical for endangered language communities: with few fluent speakers available to catch and correct errors, mistranslations risk propagating through educational materials and learner communities, potentially undermining the very linguistic heritage we hope to preserve.

1.1 Contributions

We present the first systematic evaluation and extension of LLM-Assisted Rule-Based Machine Translation (LLM-RBMT). Specifically, we implement and compare five English-to-Owens Valley Paiute (OVP) translators using OpenAI's gpt-4o and gpt-4o-mini models:

1. **Pipeline**: A reengineered LLM-RBMT system with richer grammatical support.
2. **Builder**: An LLM-RBMT variant that constructs sentences step-by-step within a constrained vocabulary.
3. **Instructions**: A prompt-only baseline using in-context grammar (Tanzer et al., 2024).
4. **Fine-tuned**: An LLM fine-tuned on a small parallel corpus generated via LLM-RBMT.
5. **RAG**: A retrieval-augmented system grounded in dictionary lookups.

Our evaluation methodology is designed to be rigorous under the constraints of endangered language research. All translations were verified for grammaticality by intermediate-level speakers (native speakers being extremely rare), and back-translations for the LLM-RBMT systems were produced by these same speakers. Rather than relying on large-scale human evaluation (which is infeasible given speaker scarcity) we propose an evaluation framework based on automatic semantic similarity metrics, baseline contextualization, and qualitative examples that together provide meaningful insight into system behavior across a range of sentence complexities. This methodology offers a practical template for future work in settings where traditional evaluation approaches are impossible.

We evaluate these systems on a curated dataset of 150 English sentences spanning six distinct sentence types and compare them across translation quality and runtime. Our analysis combines both automatic metrics and qualitative observations, surfacing tradeoffs in structure, reliability, cost, and ease of implementation. While the evaluation is focused on a single endangered language, the sentence set was carefully designed to cover a broad range of grammatical constructions, allowing us to draw meaningful and generalizable conclusions.

2 Related Work

There is decades of research in machine translation for low-resource languages (Ranathunga et al., 2023), but these methods often do not perform well for endangered or "no-resource" languages, which typically lack sufficient parallel corpora for any training. As a result, traditional paradigms like rule-based machine translation (RBMT) and statistical machine translation (SMT) are still relevant for endangered languages (Torregrosa et al., 2019; Khanna et al., 2021; Pirinen, 2019), despite their reliance on expert-crafted rules. The LLM-RBMT

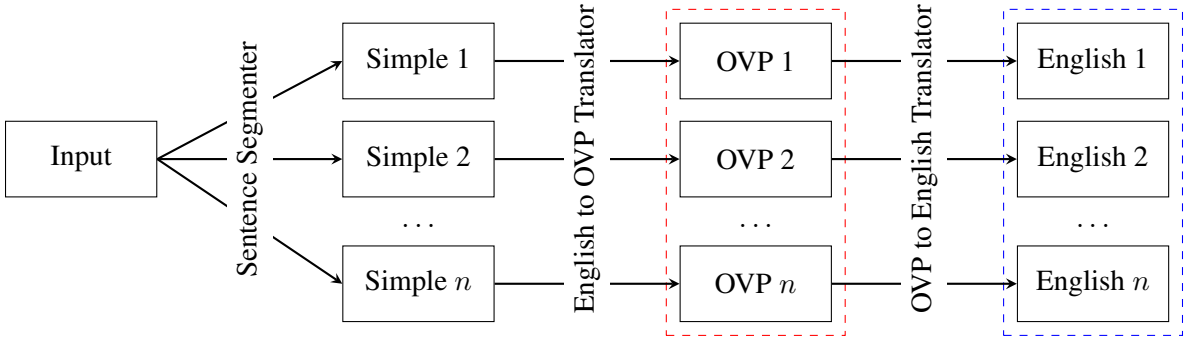

Figure 1: The "Pipeline" English to OVP translation process originally proposed in (Coleman et al., 2024). The box with a red, dashed border indicates the set of sentences in Owens Valley Paiute (the target language) and the box with a blue, dashed border indicates the set of English sentences they translate to. Ideally, the input sentence, simple sentences, and English output sentences will have high semantic similarity.

paradigm prototyped in (Coleman et al., 2024) offers a novel alternative that combines the determinism of RBMT with the flexibility of modern LLMs.

LLM-RBMT relies on two common in-context learning techniques: *prompt engineering* and *few-shot learning*, which supply task-relevant information without modifying model weights (Brown et al., 2020). There is growing interest in using such techniques for low-resource machine translation (Zhang et al., 2024; Court and Elsner, 2024; Elsner and Needle, 2023; Tanzer et al., 2024; Aycock et al., 2024), although these approaches are not rule-based and rely on LLMs to generate translations directly from in-context linguistic hints. Recent work, however, suggests state-of-the-art LLMs still struggle in these settings (Court and Elsner, 2024). One of the translators presented in this paper is based on the "Machine Translation from One Book" method (Tanzer et al., 2024).

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) is another emerging strategy in which models access external tools or corpora at inference time. RAG has been applied to low-resource translation by incorporating dictionaries or example sentences and has gained traction more broadly as a technique for enhancing LLMs without retraining (Bubeck et al., 2023; Frieder et al., 2023; Park et al., 2023; Wang et al., 2023). We implement a RAG-style translator using OpenAI's function-calling capabilities to ground translations in trusted linguistic sources.

Fine-tuning has improved LLM performance across many tasks (Han et al., 2024), including translation for moderately low-resource languages (Lankford et al., 2023; Cruz and Cheng, 2019). However, in extremely low-resource settings, fine-tuning on limited data is unlikely to match the reliability of rule-based or prompt-based methods. We include a fine-tuned translator trained on a small parallel corpus to directly compare this strategy with LLM-RBMT and prompt-based alternatives.

To our knowledge, this is the first work to directly compare LLM-RBMT, prompt-only, RAG, and fine-tuned translation strategies in an extremely low-resource, endangered language setting.

3 Approaches

We implemented five translators: (1) Pipeline, (2) Builder, (3) Instructions, (4) Fine-tuned, and (5) RAG. In this section, we describe each translator in more detail.

3.1 Pipeline Translator

The Pipeline Translator builds on the LLM-Assisted Rule-Based Machine Translation (LLM-RBMT) paradigm introduced in (Coleman et al., 2024), but with a rearchitected implementation to support richer linguistic structures and greater modularity. At its core is a redesigned sentence builder that leverages OpenAI's structured output functionality and a formal schema defined using `pydantic`. This allows the system to capture fine-grained grammatical features such as **proximity**, **person**, **plurality**, **inclusivity**, and **reflexivity** with structured validation.

Each sentence is a structured object with three components: a **subject** (pronoun or noun, optionally verb-derived, annotated with proximity, plurality, and possessive markers), a **verb** (marked for tense and aspect, optionally prefixed with an object pronoun), and an optional **object** (noun or

nominalized verb with possessive determiners and agreement suffixes). This supports constructions like "sawa-dü-ii hukaw̃ia-doka ui-buni-ku" ("The cook saw the walkers"), where both subject and object are nominalized verbs. Newly supported features include agentive nominalization, possessive determiners (including reflexive forms marked differently in OVP), and pronoun disambiguation distinguishing "you and I" (`taa`) from inclusive/exclusive "we." The validation-first design improves robustness and transparency. For vocabulary, logic, and implementation details, see Appendix 6.

3.2 Builder Translator

The Builder Translator is a new LLM-RBMT approach that gives the LLM access to *only* the sentence-building tool and asks it to build sentences one choice at a time. This is unlike the Pipeline Translator, which uses various tools (see Figure 1) to build sentences. While the Pipeline approach never interacts with the target language, the Builder approach is given words in the target language and their English translations at each step of the sentence-building process. The potential advantage of this approach is that it is more aware of the vocabulary available and relies on fewer expert-designed tools. For this reason, the Builder approach might be more flexible and easier to implement for new languages and sentence constructions. We suspect it will, however, be slower and more expensive since it requires more interactions with the LLM (making one call to the model for every word it selects). The prompts and examples used for the Builder Translator can be found in Appendix 7.

3.3 Instructions Translator

The Instructions Translator is a new approach inspired by (Tanzer et al., 2024). In this approach, the LLM has no access to any tools or sentence-building capabilities. Instead, it is given a set of instructions on how to build sentences in the target language (i.e., a small "grammar book") and asked to use them to generate translations. Because this approach requires no tooling, it is the easiest to implement. Unlike the Pipeline and Builder approach, however, the translations it produces are not guaranteed to be grammatically correct. The prompts and examples used for the Instructions Translator can be found in Appendix 8.

3.4 Fine-tuned Translator

The Fine-tuned Translator is built using a standard fine-tuning approach. We fine-tune the gpt-4o and gpt-4o-mini models on 393 parallel sentences generated using the sentence builder tool described in Section 3.1. To ensure the dataset is representative of the vocabulary and sentence constructions supported by the other translators, we follow a structured approach to generate the parallel data. We start by selecting subjects and verbs from the vocabulary to guarantee that every verb, noun, and pronoun is represented in the dataset at least three times. Other parts of speech (objects, suffixes, etc.) are then randomly selected to complete each sentence. This results in 243 unique sentence pairs.

We also included an additional 125 sentence pairs with unknown vocabulary, to allow the model to learn to handle partial translations. We generated sentences with either subjects, objects, verbs, subjects and objects, or all three (subjects, objects, and verbs) that are missing from the vocabulary. The last 25 sentence pairs represent examples where the translator must simplify the sentence before translating. For example, the English sentence "She laughed quietly at his silly joke" should be simplified to something like "He told a joke. She laughed." before translating since the dataset contains no examples of prepositional phrases. Similarly the other translators are not given the information or tools to handle complex sentence structures like this and so must rely on simplifying the sentence before translating. The prompts and examples used for the Fine-tuned Translator can be found in Appendix 9.

3.5 RAG Translator

The RAG Translator uses a Retrieval-Augmented Generation approach to assist the LLM in generating translations grounded in real language data. Specifically, it leverages OpenAI's function-calling capabilities to interface with an external OVP dictionary. When translating, the system identifies key content words (e.g., nouns, verbs) from the English input and issues semantic search queries to the dictionary API. The returned results include matching OVP words, definitions, and usage examples. These are injected into the prompt as context, which the LLM then uses to produce a translation.

This approach requires no prior training or prompt-specific tuning for the target language. Instead, the dictionary serves as the only source of linguistic information, and the model is prompted

to adhere as closely as possible to the forms and grammar evident in the retrieved examples. The advantage of this method is that it can work with any language for which even a small dictionary and set of example sentences exist. However, because it does not use structured tools like LLM-RBMT or explicit grammar rules like the Instructions Translator, its output can vary in grammaticality and fluency depending on the coverage and quality of the retrieved material. The prompts and examples used for the RAG Translator can be found in Appendix 10.

4 Evaluation

We evaluate the translators using a dataset of 150 sentence pairs, evenly distributed across six sentence types: subject-verb, subject-verb-object, two-verb, two-clause, complex, and nominalization.

4.1 Evaluation Metrics

Translation quality is assessed using five metrics, which fall into two categories: lexical overlap metrics and semantic similarity metrics.

Lexical Metrics. We report BLEU and chrF++ scores for completeness, though these metrics are known to be limited in settings like ours. They rely on surface-level overlap with a reference and penalize semantically faithful but lexically divergent translations.

Semantic Metrics. To better evaluate meaning preservation, we also report:

- **BERTScore** (Zhang et al., 2020), which compares contextualized embeddings of sentences to assess alignment at the token level.
- **COMET** (Rei et al., 2020), a learned metric trained to estimate human judgments of translation quality.
- **Semantic Similarity (MiniLM)**, computed as the cosine similarity between `all-MiniLM-L6-v2` model (Reimers and Gurevych, 2020) sentence embeddings.

Although all five metrics are reported, the main results in this paper focus on the MiniLM-based semantic similarity score, as it most effectively captures the intended behavior of our translation system: preserving meaning, even if surface forms differ. Results for BLEU, chrF++, BERTScore, and COMET are consistent with Semantic Similarity and are included in the appendices.

4.2 Contextualizing Evaluation Scores

Because our translation task emphasizes semantic alignment (and because traditional ground-truth references are unavailable) we contextualize the evaluation scores using a baseline analysis. Specifically, we compute each metric over all 11,175 distinct pairs of unrelated sentences in the dataset. These scores represent a background distribution of unrelated sentence comparisons and serve as a proxy for what a "typical" low similarity score looks like in our setting.

Table 1 shows the mean and standard deviation for each metric under this null hypothesis. This allows us to define conservative quality thresholds. For example, a translation that yields a Semantic Similarity score above $\mu + 3\sigma$ (e.g., > 0.746) is very unlikely to be unrelated to the input sentence and therefore likely preserves core meaning.

Table 1: Baseline scores computed over all pairwise comparisons of unrelated sentences in the dataset.

Metric	Mean	Standard Deviation
Semantic Similarity	0.569	0.059
BLEU Score	0.041	0.022
chrF++ Score	13.571	6.007
BERTScore (F1)	0.887	0.018
COMET Score	0.452	0.093

Baseline histograms for all metrics are provided in Appendix 11. We believe that this kind of contextual analysis is especially valuable in low/no-resource translation, where traditional reference-based evaluation is impractical. We encourage other researchers working in this space to adopt similar baselining strategies to interpret metric scores more meaningfully.

To apply these metrics in the absence of ground-truth references, we follow the indirect evaluation strategy from (Coleman et al., 2024). For each translation, we compute similarity scores between the original English input and two English reconstructions: the *backwards translation*, generated by translating the output OVP sentence back into English, and the *comparator translation*, which is the backwards translation with any vocabulary not available to the LLM-RBMT translators replaced by placeholders (e.g., `[VERB]`, `[OBJECT]`). The comparator is important because the backwards evaluation does not penalize the LLM-RBMT sys-

tems for using English placeholders.

A good translation, then, should score high on both the backwards and comparator translations. The following translation, for example, is perfect:

They are climbing.

Target (translator/model: Instructions/gpt-4o-mini)
Uhuw̃a tsibui-ti.
Backwards
They are climbing. 1.00
Comparator
They are climbing. 1.00

Another interesting case, though, is when the backwards translation is good but the comparator translation is bad, as in the following example:

The king wore a crown.

Target (translator/model: Pipeline/gpt-4o)
[crown]-neika [king]-uu ma-[wear]-ku.
Backwards
The king wore the crown. 0.995
Comparator
[SUBJECT] [VERB]-ed [OBJECT]. 0.541

This is a case where the translation is correct, but the LLM-RBMT translators do not have the vocabulary for an exact translation. When the backwards *and* comparator translations are bad, the translation is likely incorrect. For example:

My brother and I went hiking.

Target (translator/model: Pipeline/gpt-4o)
mia-ku [brother]-ii. nüü mia-ku.
Backwards
My brother went. I went. 0.837
Comparator
[SUBJECT] went. I went. 0.634

This is a case where the Pipeline translator performed poorly: a lot of information was lost in simplifying the input sentence.

Because the Pipeline and Builder Translators are based on the LLM-RBMT paradigm, they always produce grammatically correct sentences. This is not the case, however, for the Instructions and Fine-tuned Translators. Whenever the target sentence was ungrammatical, we assigned backwards and comparator semantic similarity scores of 0.0. For example, the following translation is not grammatical. Subject pronouns like "uhu" (he/she/it, distal) cannot take subject determiner suffixes like "-uu":

He works.

Target (translator/model: Instructions/gpt-4o-mini)
waakü-dü uhu-uu.
Backwards
N/A 0.00
Comparator
N/A 0.00

4.3 Results and Discussion

In this section, we present the results of evaluating the four translators on the dataset of 150 sentences. Figures 2 and 3 show the backwards and comparator semantic similarity scores, respectively. Some interesting observations can be made from these results. First, the Pipeline Translator is the most consistent across sentence types and models, likely due to it having the most structure and tooling to guide the translation process. Interestingly, the gpt-4o Instructions Translator performs relatively well, especially given its ease of implementation and quick translation time. It does have much higher variance, however, which suggests that it isn't as reliable as the Pipeline and Builder Translators. The Fine-tuned Translator performs the worst, likely due to the small number of parallel sentences used to train the model. It's important to note that both the Instructions and Fine-tuned Translators' performance depends greatly on prompt design and the quality of the parallel data used to train the model (respectively). Evaluating different versions of these translators with different prompts and training data is an important area for future work. Perhaps surprisingly, the RAG Translator also performs poorly, despite having access to significantly more vocabulary and example sentences than the other systems. A closer inspection reveals the reason for this: many of its translations are *nearly* correct, but are ungrammatical. For example:

Romeo wrote a letter.

Target (translator/model: RAG/gpt-4o)
Romeo i-dü-mui-pü piponibü.
Backwards
N/A 0.000
Comparator
N/A 0.000

This sentence attempts to translate "Romeo wrote a letter" by composing "i-dü-mui-pü" (something like "has written me") with "piponibü" ("paper"). While semantically aligned with the input, the sen-

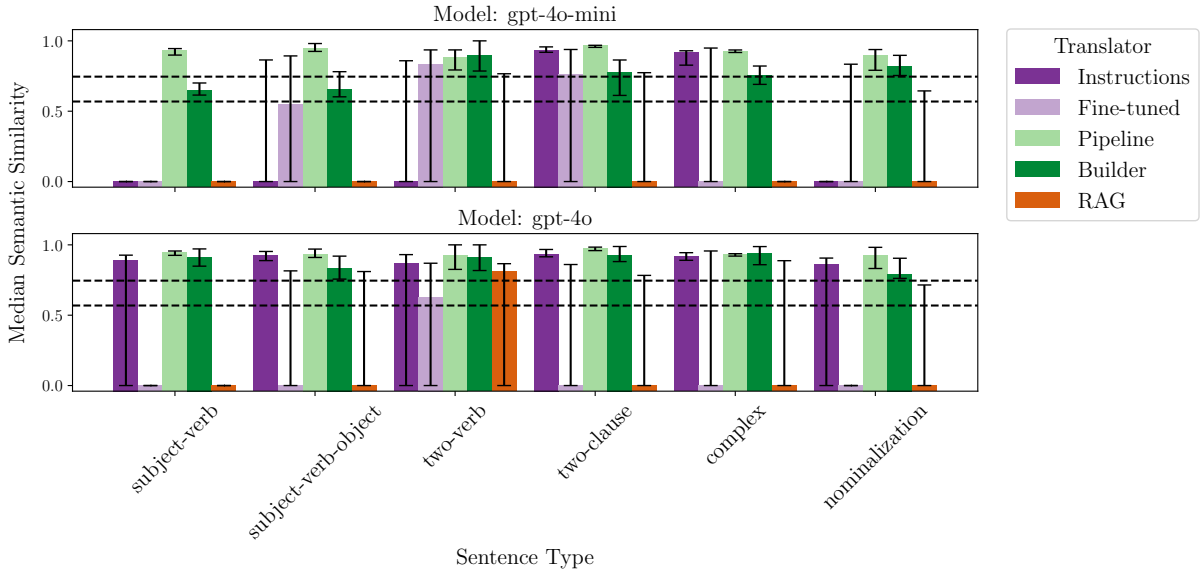

Figure 2: Backwards similarity scores: semantic similarity between target and backwards sentences. The colored bars indicate the median for each translator and error bars indicate the 25th and 75th percentiles. The dashed lines represent the baseline semantic similarity scores for unrelated sentences μ and $\mu + 3\sigma$ (see Section 4).

tence is not grammatically correct, and so cannot be reliably back-translated. Thus, it receives a semantic similarity score of 0 under our evaluation framework. This suggests that the RAG Translator may benefit from more explicit guidance such as access to grammatical rules, partial sentence builders, or controlled generation mechanisms to turn these "almost right" outputs into correct and reliable translations. Exploring such hybrid strategies is a promising direction for future work.

The two LLM-RBMT approaches (Pipeline and Builder) are particularly interesting to compare. The Pipeline Translator generally performs better than the Builder Translator on the backwards metric, but the Builder Translator performs equally well and sometimes better on the comparator metric. In Section 3.2, we hypothesized that the Builder Translator might be more capable of producing good translations when the exact vocabulary used in the input sentence is not available. The results suggest this is sometimes the case. Consider the following translation:

The chef prepared a meal.

Target (translator/model: Builder/gpt-4o)
[chef]-uu tuunapi-neika a-zawa-ku.
Backwards
The chef cooked the food. 0.926
Comparator
[SUBJECT] cooked food. 0.766

and the same sentence by the Pipeline Translator:

The chef prepared a meal.

Target (translator/model: Pipeline/gpt-4o)
[meal]-neika [chef]-uu a-[prepare]-ku.
Backwards
The chef prepared the meal. 0.997
Comparator
[SUBJECT] [VERB]-ed [OBJECT]. 0.529

Because the Builder Translator is aware of the vocabulary available in the system, it chooses to use the words "cook" and "food" (available in the vocabulary) instead of "prepare" and "meal" (not available in the vocabulary) to produce a translation that doesn't rely on so many English placeholders.

Table 2 shows average cost per translation. The Instructions and Fine-tuned translators are cheapest since they make only one model call; Builder is most expensive due to many calls. Pipeline is fairly inexpensive and balances cost with quality well. RAG is cheaper than Builder but costlier than Pipeline, likely due to function-calling overhead and larger context windows.

Table 3 shows average translation time per sentence, with results consistent with cost: Instructions and Fine-tuned are fastest, Builder is slowest (due to multiple model calls), and RAG falls between Builder and Pipeline.

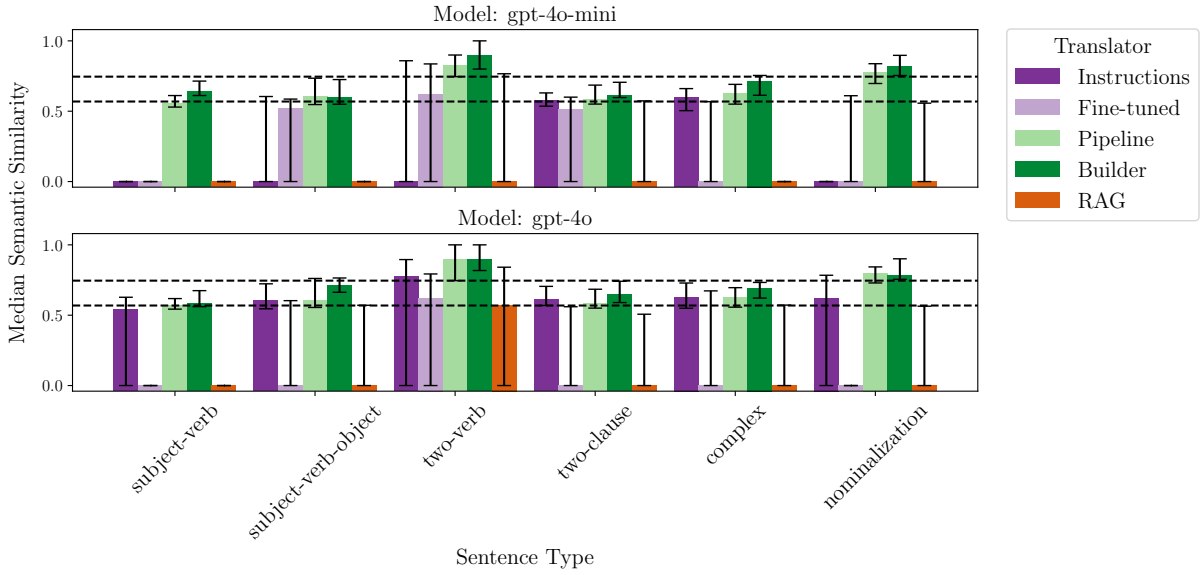

Figure 3: Comparator similarity scores: semantic similarity between target and comparator sentences. The colored bars indicate the median for each translator and error bars indicate the 25th and 75th percentiles. The dashed lines represent the baseline semantic similarity scores for unrelated sentences μ and $\mu + 3\sigma$ (see Section 4).

Table 2: Avg. Cost/Translation and Standard Deviation

Translator	Model	Avg. Cost ($)	Std. Dev. ($)
Builder	4o	0.581	0.186
Builder	4o-mini	0.024	0.011
Fine-tuned	4o	0.001	0.000
Fine-tuned	4o-mini	0.000	0.000
Instructions	4o	0.003	0.000
Instructions	4o-mini	0.000	0.000
Pipeline	4o	0.015	0.001
Pipeline	4o-mini	0.001	0.000
RAG	4o	0.107	0.003
RAG	4o-mini	0.006	0.000

Table 3: Avg. Translation Time and Standard Deviation

Translator	Model	Avg. Time (s)	Std. Dev. (s)
Builder	4o	18.855	8.066
Builder	4o-mini	10.991	5.264
Fine-tuned	4o	1.317	2.411
Fine-tuned	4o-mini	2.047	10.797
Instructions	4o	1.262	0.383
Instructions	4o-mini	0.848	1.347
Pipeline	4o	9.391	49.176
Pipeline	4o-mini	5.862	2.498
RAG	4o	7.288	1.956
RAG	4o-mini	10.677	3.908

5 Conclusion

We presented and evaluated five machine translation approaches for extremely low-resource languages. The LLM-Assisted Rule-Based Machine Translation (LLM-RBMT) paradigm, particularly the Pipeline Translator, achieved the best overall performance, balancing grammaticality and semantic fidelity. The Builder Translator handled lexical gaps well but was more costly and the Instructions and Fine-tuned translators were faster and cheaper but less reliable. Surprisingly, the RAG Translator underperformed despite having access to more vocabulary and sentence examples, suggesting that unconstrained retrieval has limited benefit in low-resource settings. We also presented an evaluation framework that prioritizes grammaticality and semantic alignment over word-for-word accuracy.

We believe this work opens many avenues for impactful future research. One direction is to extend LLM-RBMT to support even more advanced constructions. Another is to combine LLM-RBMT with retrieval, more careful prompt-engineering, and fine-tuning. Ablation studies could also shed light on which components (tools, prompts, training data, etc.) most influence performance. Finally, there is a need for standardized benchmarks tailored to extremely low-resource languages. We view this work as a foundation toward that broader goal.

References

Seth Aycock, David Stap, Di Wu, Christof Monz, and Khalil Sima'an. 2024. Can LLMs Really Learn to Translate a Low-Resource Language from One Grammar Book?

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712.

Aakanksha Chowdhery and 1 others. 2022. PaLM: Scaling Language Modeling with Pathways.

Jared Coleman, Bhaskar Krishnamachari, Ruben Rosales, and Khalil Iskarous. 2024. LLM-assisted rule based machine translation for low/no-resource languages. In *Proceedings of the 4th Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP 2024)*, pages 67–87, Mexico City, Mexico. Association for Computational Linguistics.

Sara Court and Micha Elsner. 2024. Shortcomings of llms for low-resource translation: Retrieval and understanding are both the problem. *CoRR*, abs/2406.15625.

Jan Christian Blaise Cruz and Charibeth Cheng. 2019. Evaluating language model finetuning techniques for low-resource languages. *CoRR*, abs/1907.00409.

Micha Elsner and Jordan Needle. 2023. Translating a low-resource language using GPT-3 and a human-readable dictionary. In *Proceedings of the 20th SIGMORPHON workshop on Computational Research in Phonetics, Phonology, and Morphology, SIGMORPHON@ACL 2023, Toronto, Canada, 14 July 2023*, pages 1–13. Association for Computational Linguistics.

Simon Frieder, Luca Pinchetti, Alexis Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2023. Mathematical capabilities of chatgpt. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient finetuning for large models: A comprehensive survey. *CoRR*, abs/2403.14608.

Tanmai Khanna, Jonathan N. Washington, Francis M. Tyers, Sevilay Bayatlı, Daniel G. Swanson, Tommi A. Pirinen, Irene Tang, and Hèctor Alòs i Font. 2021. Recent advances in Apertium, a free/open-source rule-based machine translation platform for low-resource languages. *Machine Translation*, 35(4):475–502.

Séamus Lankford, Haithem Afli, and Andy Way. 2023. adaptmllm: Finetuning multilingual language models on low-resource languages with integrated LLM playgrounds. *Inf.*, 14(12):638.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pages 2:1–2:22. ACM.

Tommi A Pirinen. 2019. Workflows for kickstarting RBMT in virtually no-resource situation. In *Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages*, pages 11–16, Dublin, Ireland. European Association for Machine Translation.

Surangika Ranathunga, En-Shiun Annie Lee, Marjana Prifti Skenduli, Ravi Shekhar, Mehreen Alam, and Rishemjit Kaur. 2023. Neural machine translation for low-resource languages: A survey. *ACM Comput. Surv.*, 55(11):229:1–229:37.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Nathaniel R. Robinson, Perez Ogayo, David R. Mortensen, and Graham Neubig. 2023. ChatGPT MT: Competitive for High- (but not Low-) Resource Languages.

Garrett Tanzer, Mirac Suzgun, Eline Visser, Dan Jurafsky, and Luke Melas-Kyriazi. 2024. A benchmark for learning to translate a new language from one grammar book. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Daniel Torregrosa, Nivranshu Pasricha, Maraim Masoud, Bharathi Raja Chakravarthi, Juan A. Alonso, Noe Casas, and Mihael Arcan. 2019. Leveraging rule-based machine translation knowledge for under-resourced neural machine translation models. In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks, MT-Summit 2019, Dublin, Ireland, August 19-23, 2019*, pages 125–133. European Association for Machine Translation.

Xuena Wang, Xueting Li, Zi Yin, Yue Wu, and Jia Liu. 2023. Emotional intelligence of large language models. *Journal of Pacific Rim Psychology*, 17:18344909231213958.

Chen Zhang, Xiao Liu, Jiuheng Lin, and Yansong Feng. 2024. Teaching large language models an unseen language on the fly. *CoRR*, abs/2402.19167.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. *CoRR*, abs/1904.09675.

6 Pipeline Translator

The Pipeline Translator is a modular system that breaks down the translation process into three main stages: sentence simplification, structured translation, and backwards translation. Each stage is handled by an LLM guided via system prompts, examples, and structured outputs. This architecture enables flexibility, transparency, and better control over how grammar and vocabulary are applied.

6.1 Step 1: Sentence Simplification

The first step converts the original English sentence into one or more simple SV/SVO constructions. These simplified sentences are expressed in a structured JSON format that includes the subject, verb, object (if applicable), and associated grammatical features such as tense, aspect, and proximity.

The system prompt for this step instructs the LLM to split input sentences into semantically equivalent simple clauses, avoiding extra modifiers and maintaining grammatical simplicity:

```
You are an assistant that splits user input sentences into a set of simple SVO or SV sentences. The set of simple sentences should be as semantically equivalent as possible to the user input sentence. No adjectives, adverbs, prepositions, or conjunctions should be added to the simple sentences. Indirect objects and objects of prepositions should not be included in the simple sentences. Subjects and objects can be verbs (via nominalization) (e.g., "run" → "the runner", "the one who ran", "the one who will run").
```

For example, given the input sentence:

```
I saw two men walking their dogs while drinking coffee.
```

The model might return:

```
{ "sentences": [ {"subject": "I", "verb": "see", "verb_tense": "past", "object": "man"}, {"subject": "man", "verb": "walk", "verb_tense": "past_continuous", "object": "dog"}, {"subject": "man", "verb": "drink", "verb_tense": "past_continuous", "object": "coffee"} ] }
```

These structured representations serve as the intermediate form that feeds into the next stage.

6.2 Step 2: Structured Translation to OVP

Each simplified sentence is then translated into a grammatical OVP sentence using a deterministic set of sentence-building functions. These functions rely on explicit mappings between English lemmas and available Paiute roots (for nouns and verbs), combined with correct suffixes for tense, aspect, plurality, and proximity.

The model uses custom logic to convert structured inputs into Paiute phrases, considering:

- **Pronoun resolution**: e.g., `first-person singular → nüü`.
- **Tense and aspect suffixes**: e.g., `past + completive → -ku`, `present continuous → -ti`.
- **Object pronoun prefixes**: added before the verb for transitive verbs.
- **Word lookup and fallback**: If a word is not found in the lexicon, it is left in brackets (e.g., `[crown]`).

The resulting Paiute sentence components are ordered based on grammatical rules: subject-first for pronouns, verb-first for noun subjects.

6.3 Step 3: Backwards Translation

To enable evaluation, the structured Paiute output is reverse-engineered back into English. This step uses the same grammar logic and vocabulary definitions used during construction, and represents each sentence as a list of parts of speech (e.g., subject, object, verb) and their definitions.

These parts are passed to the LLM, which is prompted to return a natural English translation:

```
You are an assistant for translating structured sentences into natural English sentences.
```

An example input to this prompt might look like:

```
[ {"part_of_speech": "subject", "positional": "proximal", "word": "wood"}, {"part_of_speech": "object", "positional": "proximal", "word": "dog"}, {"part_of_speech": "verb", "tense": "present ongoing (-ing)", "word": "see"} ]
```

The model would respond with:

```
This wood is seeing this dog.
```

This translation is then used for semantic comparison with the original English sentence.

6.4 Comparator Sentences

In addition to the backwards translation, we also compute a *comparator translation*, in which any word not available to the translator is replaced with a placeholder (e.g., `[OBJECT]`, `[VERB]`). This lets us measure how well the system did using only known vocabulary, and is particularly useful for LLM-RBMT systems that transparently insert English placeholders for out-of-vocabulary words. It helps ensure that semantic similarity is not inflated by untranslated content.

For full implementation details, including example prompts and models used, we refer the reader to the Data and Code Appendix.

7 Builder Translator

The system prompt for the Pipeline Translator is as follows:

```
You are an assistant trying to build a sentence in Paiute. The user will provide you with parts of speech and vocabulary options one at a time. Make choices to best approximate the meaning of Input Sentence. Do not make choices that are not provided by the user. This may mean you can't build the sentence you want, but that's okay. Whenever you've chosen enough parts of speech and vocabulary to form a grammatically correct sentence, the user will ask you if you want to continue. If you're happy with the sentence you've built, you can choose to stop. If not, continue selecting optional parts of speech and vocabulary until you're satisfied.
```

The model is them prompted with messages like:

```
Input Sentence: Jared will eat an apple.
Current Translation: .
Current Choices: { 'subject_noun': None, 'subject_suffix': None, 'subject_noun_nominalizer': None, 'verb': None, 'verb_tense': None, 'object_pronoun': None, 'object_noun': None, 'object_noun_nominalizer': None, 'object_suffix': None }
Please select a required part of speech: subject_noun, verb
```

and should respond with choices like:

```
subject_noun
```

, to which the translator will then prompt the model again with something like:

```
Input Sentence: Jared will eat an apple.
Current Translation: .
Current Choices: { 'subject_noun': None, 'subject_suffix': None, 'subject_noun_nominalizer': None, 'verb': None, 'verb_tense': None, 'object_pronoun': None, 'object_noun': None, 'object_noun_nominalizer': None, 'object_suffix': None }
Please select a word for subject_noun: nüü (I), uhu (he/she/it), uhuña (they), mahu (he/she/it), mahuña (they), ihi (this), ihiña (these), taa (you and I), nüügwa (we (exclusive)), taagwa (we (inclusive)), üü (you), üügwa (you (plural)), isha' (coyote), isha'pugu (dog), kidi' (cat), pugu (horse), wai (rice), tüba (pinenuts), maishibü (corn), paya (water), payahuupü (river), katünu (chair), toyabi (mountain), tuunapi (food), pasohobü (tree), nobi (house), toni (wickiup), apo (cup), küna (wood), tübbi (rock), tabuutsi' (cottontail), kamü (jackrabbit), aaponu' (apple), tüsüga (weasle), mukita (lizard), wo'ada (mosquito), wükada (bird snake), wo'abi (worm), aingwü (squirrel), tsiipa (bird), tüwoobü (earth), koopi' (coffee), pahabichi (bear), pagwi (fish), kwadzi (tail), tüka (eat), puni (see), hibi (drink), naka (hear), kwana (smell), kwati (hit), yadohi (talk to), naki (chase), tsibui (climb), sawa (cook), tama'i (find), nia (read), mui (write), nobini (visit), katü (sit), üwi (sleep), kwisha'i (sneeze), poyoha (run), mia (go), hukañia (walk), wünü (stand), habi (lie down), yadoha (talk), kwatsa'i (fall), waakü (work), wükihaa (smile), hubiadu (sing), nishua'i (laugh), tübinohi (play), yotsi (fly), nüga (dance), pahabi (swim), tünia (read), tümui (write), tsiipe'i (chirp)
Because this is a subject_noun word, you can also choose to use a wildcard by putting the word in brackets. For example: [wildcard]
```

The model is then prompted to select a word for the subject noun, and the process continues until the sentence is complete. When the sentence is grammatically correct, the model will be asked if it wants to continue or stop:

```
Input Sentence: Jared will eat an apple.
Current Translation: [Jared]-ii tüka-wei.
Enter one of the following choices:
c: Continue building the last sentence
n: Add and build a new Paiute sentence for this translation
t: Terminate and return the current translation.
```

For more information on the specific prompts, models, examples, and more, we refer the reader to the Data and Code Appendix.

8 Instructions Translator

The system prompt for the Instructions Translator contains all of the grammar rules and vocabulary available to the model and is as follows (split into two parts for readability):

```
You use the following grammar rules to translate user input sentences from English to Owens Valley Paiute. Use the vocabulary and sentence structures available to translate the input sentence as best as possible. It doesn't need to be perfect and you can leave English words untranslated if necessary.
# Vocabulary
## Nouns: isha' (coyote), isha'pugu (dog), kidi' (cat), pugu (horse), wai (rice), tüba (pinenuts), maishibü (corn), paya (water), payahuupü (river), katünu (chair), toyabi (mountain), tuunapi (food), pasohobü (tree), nobi (house), toni (wickiup), apo (cup), küna (wood), tübbi (rock), tabuutsi' (cottontail), kamü (jackrabbit), aaponu' (apple), tüsüga (weasle), mukita (lizard), wo'ada (mosquito), wükada (bird snake), wo'abi (worm), aingwü (squirrel), tsiipa (bird), tüwoobü (earth), koopi' (coffee), pahabichi (bear), pagwi (fish), kwadzi (tail)
## Transitive Verbs: tüka (eat), puni (see), hibi (drink), naka (hear), kwana (smell), kwati (hit), yadohi (talk to), naki (chase), tsibui (climb), sawa (cook), tama'i (find), nia (read), mui (write), nobini (visit)
## Intransitive Verbs: katü (sit), üwi (sleep), kwisha'i (sneeze), poyoha (run), mia (go), hukañia (walk), wünü (stand), habi (lie down), yadoha (talk), kwatsa'i (fall), waakü (work), wükihaa (smile), hubiadu (sing), nishua'i (laugh), tsibui (climb), tübinohi (play), yotsi (fly), nüga (dance), pahabi (swim), tünia (read), tümui (write), tsiipe'i (chirp)
## Object Suffixes: eika (proximal), oka (distal)
## Object Pronouns: i (me), u (him/her/it (distal)), ui (them (distal)), ma (him/her/it (proximal)), mai (them (proximal)), a (him/her/it (proximal)), ai (them (proximal)), ni (us (plural, exclusive)), tei (us (plural, inclusive)), ta (us (dual), you and I), ü (you (singular)), üi (you (plural), you all)
```

```
## Subject Suffixes: ii (proximal), uu (distal)
## Subject Pronouns: nüü (I), uhu (he/she/it), uhuwña (they), mahu (he/she/it), mahuwña (they), ihi (this), ihiwña (these), taa (you and I), nüügwa (we (exclusive)), taagwa (we (inclusive)), üü (you), üügwa (you (plural))
## Verb Nominalizer Tenses: dü (present), pü (have x-ed, am x-ed), weidü (future (will)),
# Sentence Structure
## Simple Sentence Structure:
Subject-Object-Verb: [object noun]-[object suffix] [subject noun]-[subject suffix] [object pronoun]-[verb]-[verb tense]
Subject Pronoun-Object-Verb: [object noun]-[object suffix] [subject pronoun] [object pronoun]-[verb]-[verb tense]
Subject-Verb: [verb]-[verb tense] [subject noun]-[subject suffix]
## Verb Nominalization Sentence Structure:
Subject Nominalizer: [verb]-[verb nominalizer tense]-[subject suffix] [verb nominalizer]-[verb nominalizer tense]
Object Nominalizer: [verb]-[verb nominalizer tense]-[object suffix] [subject noun]-[subject suffix] [object pronoun]-[verb]-[verb tense]
Subject&Object Nominalizer: [verb]-[verb nominalizer tense]-[object suffix] [verb nominalizer]-[verb nominalizer tense]-[subject suffix] [subject noun]-[subject suffix] [object pronoun]-[verb]-[verb tense]
# Fortis/Lenis Transformations p->b, t->d, k->g, s->z, m->w̃
```

The model is then prompted with messages like:

```
TThis cook saw the ones who walked by the house.
```

and should respond with:

```
sawa-dü-ii hukaw̃ia-doka ui-buni-ku.
```

We use few-shot examples to demonstrate how the grammar rules and vocabulary should be used to translate the input sentence. For more information on the specific prompts, models, examples, and more, we refer the reader to the Data and Code Appendix.

9 Fine-tuned Translator

The fine-tuned translator is built using a standard fine-tuning approach (the dataset used for fine-tuning is described in Section 3.4). The system prompt for the Fine-tuned Translator is as follows:

```
You are a translator for translating text from English to OVP. For any word that does not have an equivalent in OVP, leave the word untranslated and place it inside brackets.
```

The model is then prompted with messages like:

```
The jackrabbit is eating the bread.
```

and should respond with:

```
kamü-ii tüka-wei [bread]-neika.
```

Fine-tuning was conducted using the OpenAI API, targeting two base models: `gpt-4o-2024-08-06` and `gpt-4o-mini-2024-07-18`. Each model was trained on 193,920 tokens using a dataset automatically compiled from seven translation subsets (e.g., "random_good_translations.csv," "random_no_subject_noun.csv," etc.). The data was transformed into a chat-completion format, with one user-assistant message pair per row. The training was conducted with the following hyperparameters:

- **Epochs:** 3
- **Batch size:** 1
- **Learning rate multiplier:** 1.8 (for `gpt-4o-mini`), 2.0 (for `gpt-4o`)
- **Seed:** 1296158545 (for `gpt-4o-mini`), 236502696 (for `gpt-4o`)

No validation split was used during training. The final models are identified as `ft:gpt-4o-2024-08-06:kubishi::AInyiTpj` and `ft:gpt-4o-mini-2024-07-18:kubishi::AInrzzLW`, respectively.

For more information on the specific prompts, models, examples, and implementation details, we refer the reader to the Data and Code Appendix.

10 RAG Translator

The RAG (Retrieval-Augmented Generation) Translator uses an LLM augmented with retrieval capabilities to translate English to Owens Valley Paiute. It is capable of calling external tools to supplement its knowledge during translation.

10.1 Available Tools

The model has access to the following tools:

- `search_english`: retrieves dictionary definitions, glosses, and example usages for English words.
- `search_sentences`: retrieves example sentence pairs from a parallel English–Paiute dataset.

10.2 Workflow

The translator proceeds in an iterative dialogue:

1. The English sentence is passed to the model.
2. The model may invoke tools to look up word meanings or similar sentences.
3. Retrieved results are inserted into the conversation.
4. The model generates a Paiute translation, optionally using retrieved examples.

All interactions and retrieval steps are included in the translation metadata.

10.3 Few-Shot Tool Use Instruction

The model is guided to use tools through few-shot prompting. Prior examples are included in the message history that demonstrate:

- Calling `search_sentences` on full input sentences to retrieve relevant phrasal translations.
- Decomposing complex inputs and calling `search_english` on key content words (e.g., "sit," "chair," "in").
- Handling tense and argument structure by querying expressions like "present continuous" or "reflexive possessive pronoun."

This few-shot strategy "teaches" the model to reason over the tools available and to apply them appropriately even on novel inputs.

For code and configuration details, see the Data and Code Appendix.

11 Evaluation Metric Baselines

To establish reference baselines for each evaluation metric, we computed pairwise scores between all distinct sentence pairs in the dataset (11,175 total comparisons). These comparisons reflect how each metric scores unrelated or semantically distant sentences, and help contextualize what constitutes a "high" or "low" value for a given metric.

Figure 4 shows the distribution of cosine similarities between sentence embeddings generated by the all-MiniLM-L6-v2 model. As reported in the main text, the mean similarity between unrelated sentences is approximately 0.569 with a standard deviation of 0.059. Similar histograms for the other metrics are provided below. These baseline distributions are used throughout the paper to interpret evaluation results and define thresholds for semantic alignment.

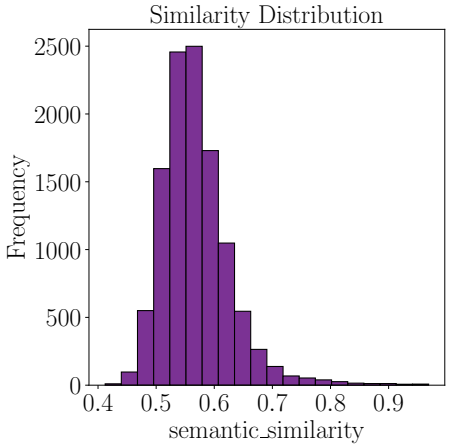

Figure 4: Distribution of MiniLM-based semantic similarity between all pairs of unrelated sentences in the dataset.

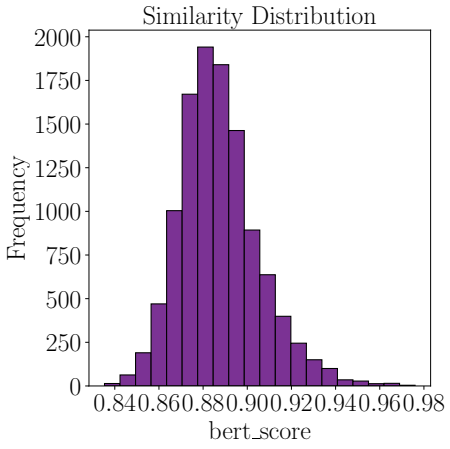

Figure 5: Distribution of BERTScore (F1) between all pairs of unrelated sentences in the dataset.

12 Vocabulary

Table 4 (on the next page) shows the vocabulary available to the Pipeline and Builder Translators.

tüka	eat
puni	see
hibi	drink
naka	hear
kwana	smell
kwati	hit
yadohi	talk to
naki	chase
tsibui	climb
sawa	cook
tama'i	find
nia	read
mui	write
nobini	visit

(a) Transitive Verbs

katü	sit
üwi	sleep
kwisha'i	sneeze
poyoha	run
mia	go
hukawĩa	walk
wünü	stand
habi	lie down
yadoha	talk
kwatsa'i	fall
waakü	work
wükihaa	smile
hubiadu	sing
nishua'i	laugh
tsibui	climb
tübinohi	play
yotsi	fly
nüga	dance
pahabi	swim
tünia	read
tümui	write
tsiipe'i	chirp

(b) Intransitive Verbs

isha'	coyote
isha'pugu	dog
kidi'	cat
pugu	horse
wai	rice
tüba	pinenuts
maishibü	corn
paya	water
payahuupü	river
katünu	chair
toyabi	mountain
tuunapi	food
pasohobü	tree
nobi	house
toni	wickiup
apo	cup
küna	wood
tübbi	rock
tabuutsi'	cottontail
kamü	jackrabbit
aaponu'	apple
tüsüga	weasle
mukita	lizard
wo'ada	mosquito
wükada	bird snake
wo'abi	worm
aingwü	squirrel
tsiipa	bird
tüwoobü	earth
koopi'	coffee
pahabichi	bear
pagwi	fish
kwadzi	tail

(c) Nouns

ku	completive (past)
ti	present ongoing (-ing)
dü	present
wei	future (will)
gaa-wei	future (going to)
pü	have x-ed, am x-ed

(d) Object Suffixes

nüü	I
uhu	he/she/it
uhuwĩa	they
mahu	he/she/it
mahuwĩa	they
ihi	this
ihiwĩa	these
taa	you and I
nüügwa	we (exclusive)
taagwa	we (inclusive)
üü	you
üügwa	you (plural)

(e) Subject Pronouns

ii	(proximal)
uu	(distal)

(f) Subject Suffixes

i	me
u	him/her/it (distal)
ui	them (distal)
ma	him/her/it (proximal)
mai	them (proximal)
a	him/her/it (proximal)
ai	them (proximal)
ni	us (plural, exclusive)
tei	us (plural, inclusive)
ta	us (dual), you and I
ü	you (singular)
üi	you (plural), you all

(g) Object Pronouns

eika	(proximal)
oka	(distal)

(h) Object Suffixes

Table 4: Vocabulary available in sentence building system.

13 BLEU and chrF++ Scores

We provide the BLEU, chrF++, BERTScore, and COMET results of the input sentences against the backwards and comparator translations (starting on the next page).

14 Reproducibility Checklist

Unless specified otherwise, each answer refers to material provided either in the main paper, in the appendix, or in the public source code and data archive (data_code_appendix.zip). That archive includes:

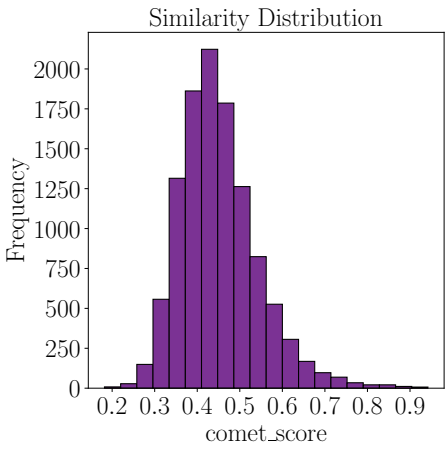

Figure 6: Distribution of COMET scores between all pairs of unrelated sentences in the dataset.

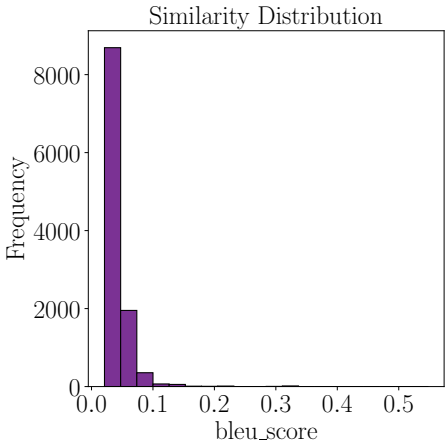

Figure 7: Distribution of BLEU scores between all pairs of unrelated sentences in the dataset.

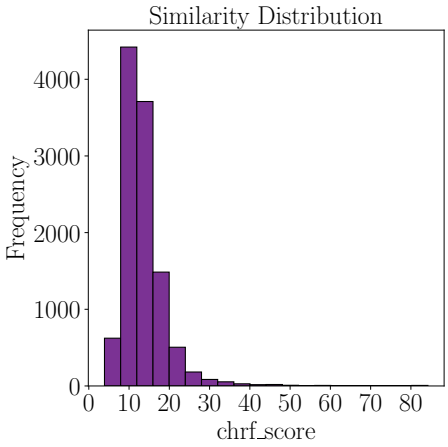

Figure 8: Distribution of chrF++ scores between all pairs of unrelated sentences in the dataset.

- Full source code for all translators and evaluation scripts
- All datasets used in experiments
- A README.md with detailed reproduction instructions

Conceptual and Descriptive Clarity

- Includes a conceptual outline and/or pseudocode description of AI methods introduced: **Yes**
- Clearly delineates statements that are opinions, hypotheses, and speculation from objective facts and results: **Yes**
- Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper: **Yes**

Theoretical Contributions

- Does this paper make theoretical contributions? **No**

Datasets

- Does this paper rely on one or more datasets? **Yes**
- A motivation is given for why the experiments are conducted on the selected datasets: **Yes**
- All novel datasets introduced in this paper are included in a data appendix: **Yes** (included in data_code_appendix.zip)
- All novel datasets will be made publicly available upon publication with a license that allows free research use: **Yes**
- All datasets drawn from the existing literature are accompanied by appropriate citations: **Yes**
- All datasets drawn from the existing literature are publicly available: **Yes**
- All datasets that are not publicly available are described in detail: **N/A**

Computational Experiments

- This paper states the number and range of values tried per (hyper-)parameter during development, along with the criterion used for selecting the final setting: **Yes** (see Appendix 9)
- Any code required for preprocessing data is included in the appendix: **Yes**

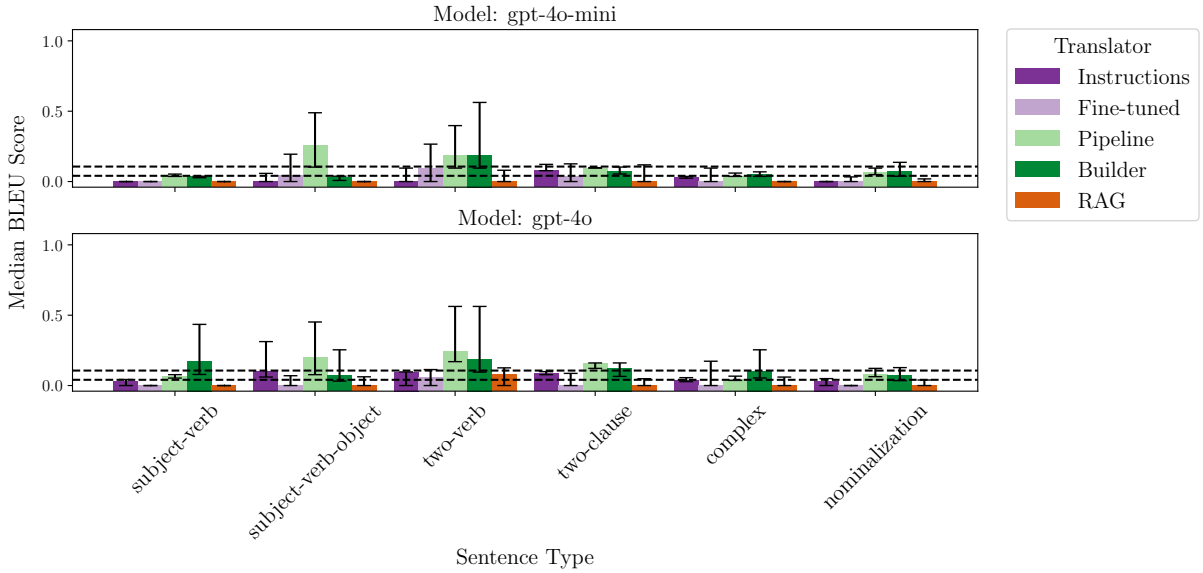

Figure 9: BLEU scores of the input sentences against the backwards translations.

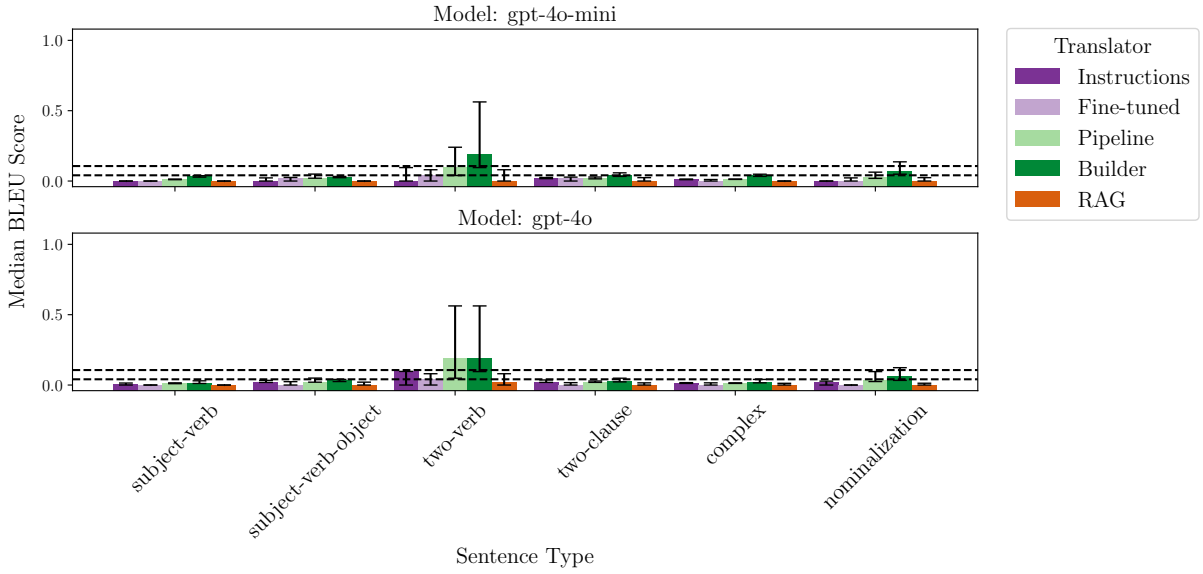

Figure 10: BLEU scores of the input sentences against the comparator translations.

- All source code for conducting and analyzing the experiments is included in a code appendix: **Yes** (data_code_appendix.zip)
- All source code will be made publicly available upon publication with a license that allows free research usage: **Yes**
- All source code implementing new methods has comments detailing the implementation, with references to the paper where applicable: **Partial** (in the Readme.md, not in the source code itself)
- If an algorithm depends on randomness, the method used for setting seeds is described: **Partial** (OpenAI API responses are nondeterministic and seeds are not controlled)
- This paper specifies the computing infrastructure used for running experiments (hardware, software libraries, OS, etc.): **Yes** (see Appendix 6)
- This paper formally describes evaluation metrics used and explains the motivation for choosing them: **Yes** (Section 4)
- This paper states the number of algorithm runs used to compute each reported result: **Yes**

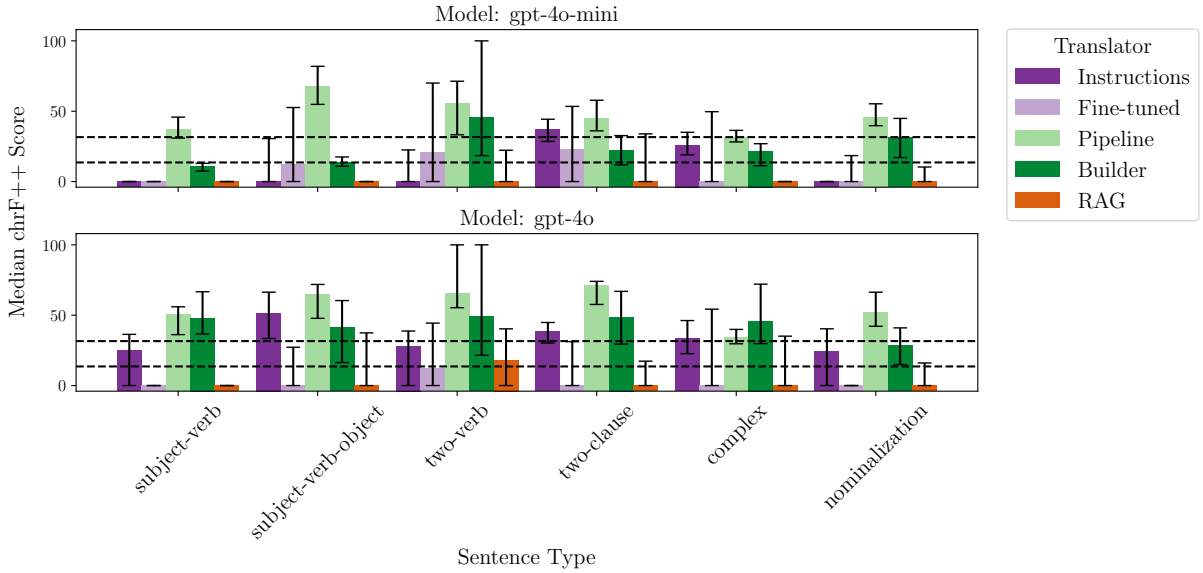

Figure 11: chrF++ scores of the input sentences against the backwards translations.

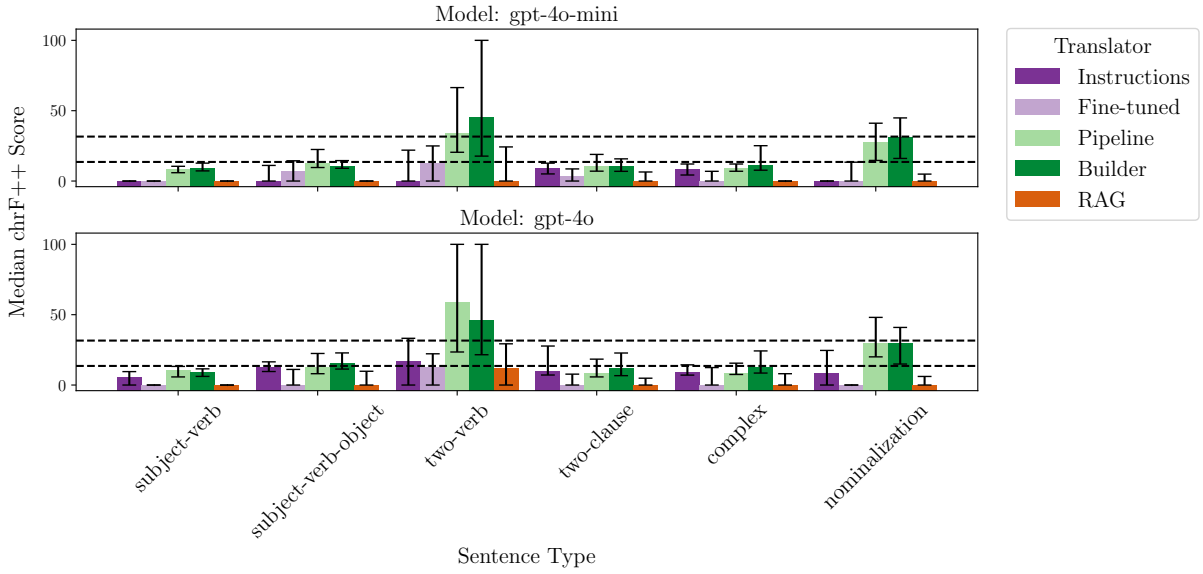

Figure 12: chrF++ scores of the input sentences against the comparator translations.

- Analysis of experiments includes measures of variation or other distributional information (e.g., error bars): **Yes**
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests: **No**
- This paper lists all final (hyper-)parameters used for each model/algorithm: **Yes** (see Appendix 9)

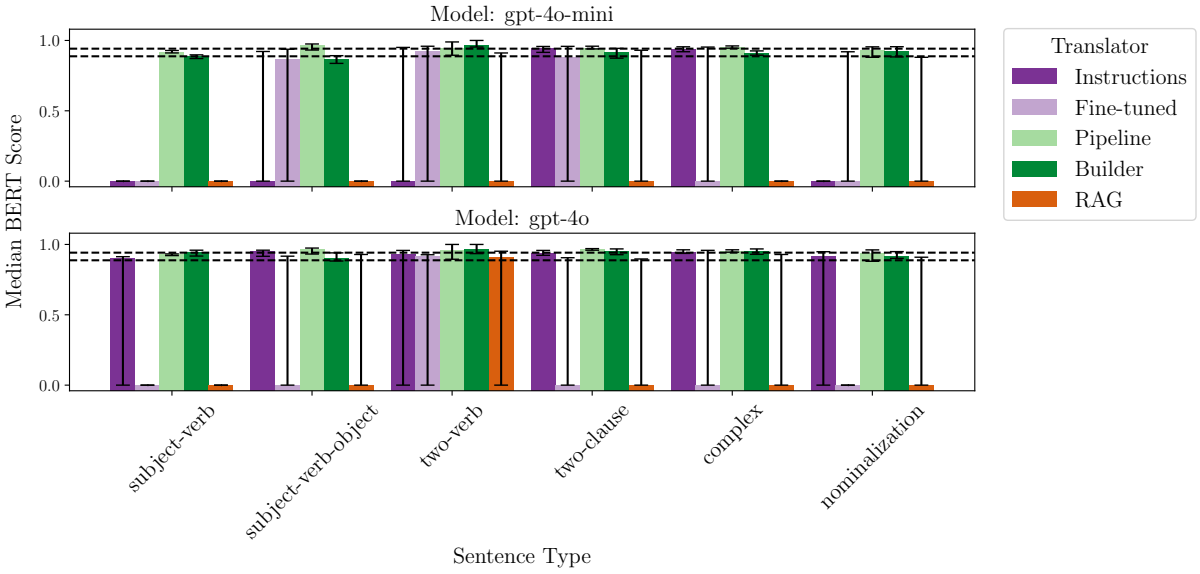

Figure 13: BERTScore of the input sentences against the backwards translations.

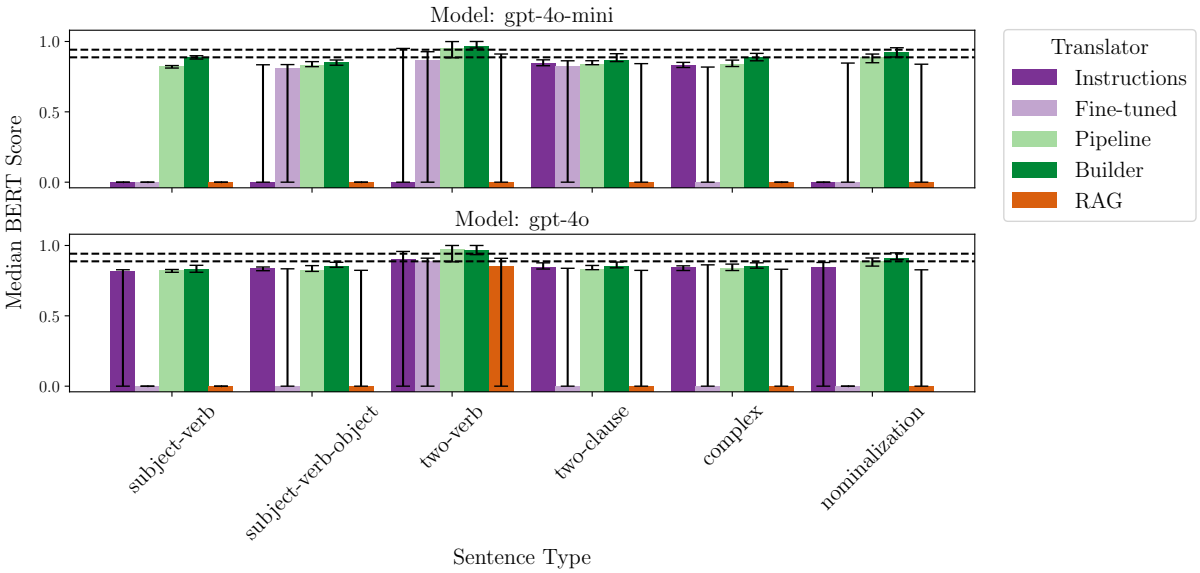

Figure 14: BERTScore of the input sentences against the comparator translations.

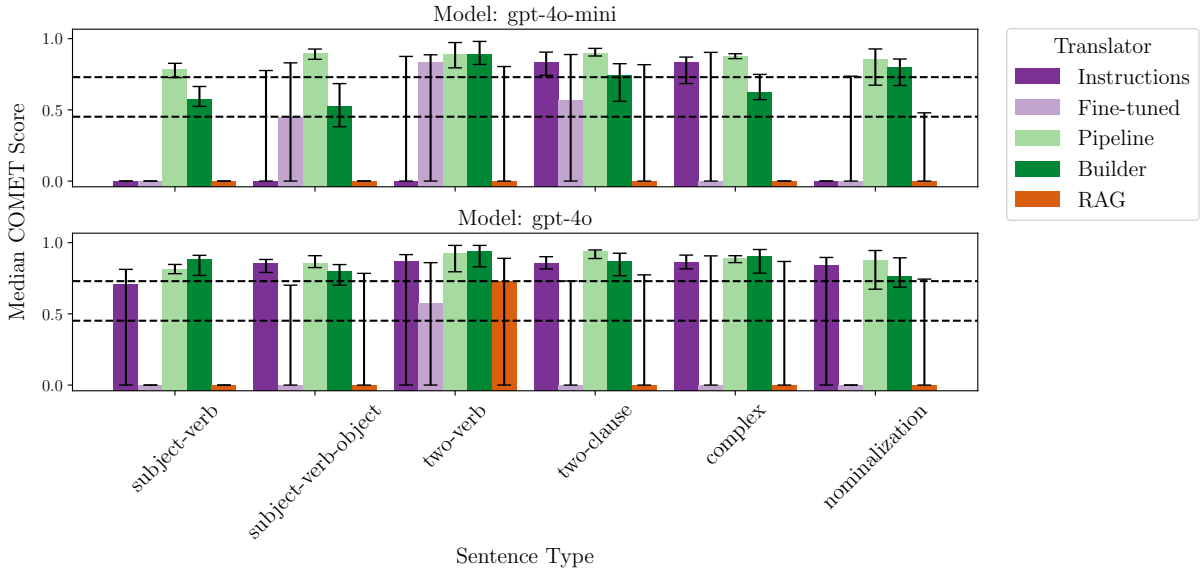

Figure 15: COMET scores of the input sentences against the backwards translations.

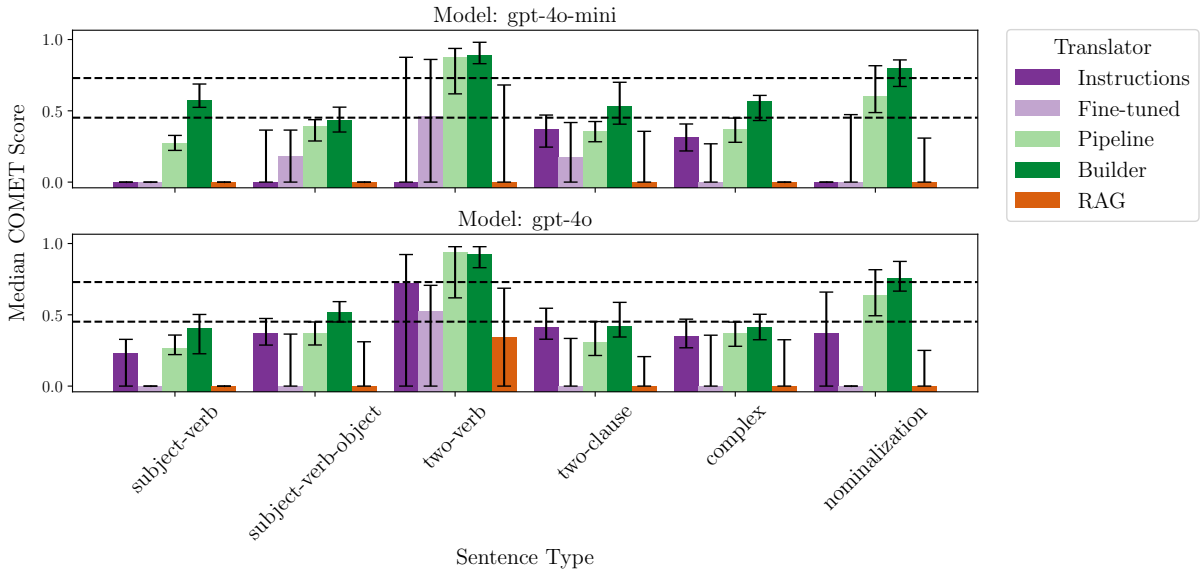

Figure 16: COMET scores of the input sentences against the comparator translations.