Panamerican Mathematical Journal
ISSN: 1064-9735
Vol 35 No. 2s (2025)

A Comparative Analysis of Large Language Models for English-to-Tamil Machine Translation with Performance Evaluation

SyedKhaleel Jageer[1]*, Priyaradhikadevi.T[2], Madhan.K[3], Prasanna.S[4]

[1]PG Scholar, Department of Computer Science and Engineering Mailam Engineering College,Mailam,Tamilnadu,India. jskcse4@gmail.com

[2]Professor, Department of Computer Science and Engineering, Mailam Engineering College,Mailam,Tamilnadu,India. hodcse@mailamengg.com

[3]Assistant Professor, Department of Information Technology, St.Joseph's College of Engineering,Chennai,Tamilnadu,India. madhanckn@gmail.com

[4]Associate Professor, Department of Computer Science and Business System, Mailam Engineering College,Mailam,Tamilnadu,India. prasannacse@mailamengg.com

Corresponding Author: SyedKhaleel Jageer*

Article History:

***Received:** 28-09-2024*

***Revised:** 19-11-2024*

***Accepted:** 30-11-2024*

Abstract:

Machine translation is essential for cross-linguistic communication, especially for low-resource languages like Tamil, which presents challenges due to complex morphology, syntax, and cultural nuances. This study compares three advanced Large Language Models (LLMs) Claude, ChatGPT, and Gemini in translating English to Tamil, focusing on culturally sensitive, political, technical, and idiomatic content. Claude enhances coherence with prompt tuning, batch processing, and temperature control, proving effective in document-level translations. ChatGPT, using a decoder-only architecture and Reinforcement Learning from Human Feedback (RLHF), aligns translations with human cultural and contextual expectations. Gemini's transformer-based framework integrates advanced tokenization and attention mechanisms, preserving tense, gender, and politeness markers in Tamil, excelling in idiomatic and technical translations. The study employed a diverse corpus, including poetry, technical documentation, and conversational text. Translation performance was evaluated using BLEU, BERT-based metrics, METEOR, and Translation Edit Rate (TER). Results show that Gemini ranks highest in accuracy and precision, especially for complex syntax and idioms. Claude demonstrates fluency but has a relatively high TER. ChatGPT scores well in semantic alignment, with high BERT and METEOR metrics. This analysis highlights the capabilities and limitations of each model in translating Tamil, providing insights into LLM performance for low-resource languages.

Keywords: Machine Translation, Large Language Models.

1. Introduction

Machine translation (MT) is all about the automated process of translating a text, paragraph, or document from one language to another using a computer algorithm. Since the 20th century, this technology has witnessed rapid growth [1]. It evolved from basic rudimentary systems to AI-driven models. The Georgetown-IBM experiment conducted in 1950's, set the stage of decades of research. This was the eye opener in MT research, but it faced many challenges in reducing the complexities of natural language. In past few, MT has seen many major breakthroughs, now the latest is neural machine translation (NMT) [2]. Today NMT has made a revolution by providing high accuracy and context-aware translation by utilizing transformer architecture. In particular, Large language models -(LLM) such as Claude (Anthropic), ChatGPT (OpenAI), and Gemini (Google) have expanded the possibilities of MT by allowing multilingual translation, and it achieve near-human performance. India is a multilingual nation, so however, the challenge remains in structural and cultural aspects. Tamil is one of the most widely spoken classical languages in India. It is well-known for its intricate morphology and syntax [3]. In all sectors like health care, legal documentation, education and government agencies, efficient and accurate translation is essential. Still the need for a more sophisticated MT system which can bridge the linguistic gap between English and Tamil is raising. Also, the need to address ethical concerns, mitigate biases, and improve the handling of low-resource languages is necessary, and the MT is integrated into daily communication and global business [4].

The objectives of the research are to compare the 3 powerful LLM models in terms of accuracy to translate English to tamil text. The models must be able to maintain both fluency and fidelity to the original meaning. Also to investigate how efficiently each model manages complex grammatical structures, idiomatic expressions and culturally embedded references specific to Tamil. This is difficult task because there may not be equivalents in English. Next is to identify the strength and weakness of each LLM model. This helps to specify where each model lack and excel. The last is to specify the limitations of each LLM model in translating Tamil and a few insights into how these findings can help in future advancements.

2. LLM MODELS

2.1 Claude (Anthropic)

The Claude model is used for the task of English-Tamil translation. It is sophisticated LLM which is specially designed for high-performance multilingual translation. The experiment carried out involves multiple strategies to optimize the translation quality and efficiency. Among all, prompt tuning is the crucial one in improving the overall model. It is a technique that allows adaption of large LLM to specific tasks. When eight in-context exemplars for sentence-level and single exemplar for document-level translation is utilized, the model efficiently maintains the meaning and contextual coherence [5]. The inclusion of examples within the prompt is one of the best practices in machine translation. This helps the model understand the nuances of specific linguistic structures, any domain specific words or idiomatic expression. In addition to this, a new batch prompt strategy was also introduced which can perform the translation of multiple sentences or paragraphs from the same document in a cohesive manner [6]. This facilitates fluency of the translation and narrative flow and tone, particularly when dealing with large. This approach is very essential in a document level machine translation. Another is batching method which can reduce API cost, by minimizing the number of individual translation requests [7]. This helps in streamlining the computational process without compromising translation quality.

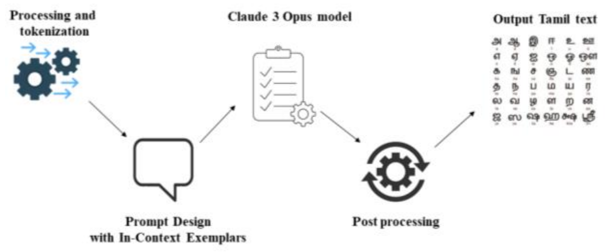
Processing and tokenization
Claude 3 Opus model
Output Tamil text
Prompt Design with In-Context Exemplars
Post processing

Fig.1 Process flow of Claude 3 Opus model

The next important factor in the Claude 3 opus model is, it involves temperature tuning. Here temperature refers to a parameter which used to control the degree of randomness in model's output as shown in figure 1. Higher temperatures lead to more creative and diverse output, while lower results more deterministic, consistent translations [8]. After several tuning, it's found that setting 0.7 is suitable to provide the best balance between translation accuracy and natural fluency. This improved the model to give translation that are both smooth and appropriate. This avoids the pitfalls of excessively creative translations that distort the exact meaning. The enhancement made in this Claude model can tackle many inherent challenges in translating between English to Tamil [9]. Tamil has high morphological complexity and syntactic nuances. This makes the language to be difficult while translating. The integration of in-context learning exemplars, batch processing and temperature tuning made the model to mitigate these challenges. The model also resulted in ensuring that translation retained both cultural relevance and linguistic fidelity. The following algorithm represents the Claude model used for English-Tamil translation used in this analysis.

Algorithm 1: Claude model for English-Tamil translation

Step 1: Prompt formulation

For each sentence s_i to be translated in $S = \{s_1, s_2, \dots s_n\}$, the prompt

$$P(s_i) = \{s_i, E_{sent}, E_{doc}\}$$

Sentence level exemplar $E_{sent} = \{e_1, e_2, \dots e_s\}$

Document level exemplar E_{doc}

Step 2: Batch translation

$$T_j = Translate\left(P\left(B_j\right)\right)$$

Batch prompt $P\left(B_j\right) = \{B_j, E_{doc}\}$ where $B_j = \{s_{j1}, s_{j2}, \dots s_{jk}\}$ represents the batches of sentences such that $j = 1mB_j = S$.

Step 3: Cost minimization

$$Cost = C(k) withk < |S|$$

C is the API call cost function and k is the number of batches. Sentences and paragraphs are grouped in batches reducing API calls.

Step 4: Temperature setting

$$T = Translate(P(x), \tau = 0.7)$$

Temperature parameter $\tau = 0.7$ controls the randomness in translation output.

Step 5: Final output

$$T = j = 1mT_j$$

Translation output T is the union of all batch translations T_j.

2.2 ChatGPT (Open AI)

LLM is categorized into two main architectures, one is encoder-decoder and another is decoder- only. An example of encoder-decoder architecture is the original transformer. It consists of two separate stacks; one is for the encoder, and another is for the decoder. But the decoder-only architecture have gained significant attention in LLM like ChatGPT.

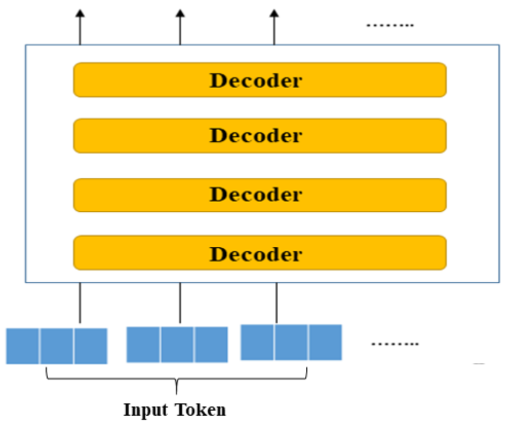

Fig.2 Decoder-only architecture

Gpt-2 and Gpt-3 are built on casual decoder architecture, handles language tasks as prediction problems and offer the advantage of implicitly learning how to accomplish several task in a unified framework [10]. 96 layers of decoder is used to generate output for GPT-3. This decoder can also use for translation, as it can generate the target sequence autoregressively, processing and predicting tamil output tokens based on the English input sequence as shown in figure 2. This is different from encoder-decoder structure, where there is a need to create two different stacks. Techniques like data parallelism, pipeline parallelism and tensor parallelism are necessary for training large scale models [11]. While translating from a high resource language to low resource language handling the complexity is crucial.

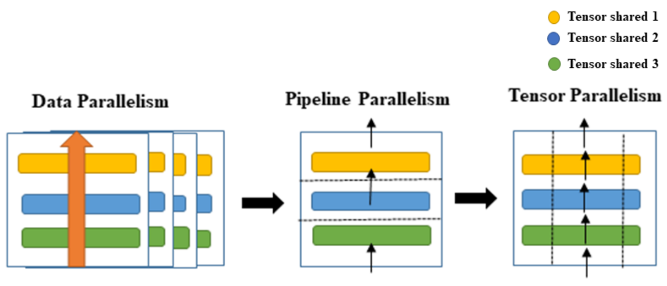

Fig.3 Parallel Processing Methods in OpenAI's LLM

Figure 3 showcases a decoder-only architecture, an elegant and powerful neural network design that underpins language models such as GPT-3. This architecture is composed of multiple stacked decoder layers, each carefully processing the input sequence comprised of tokens that represent words or phrases. As the input flows through each layer, the decoder generates an output sequence token-by-token, with each new token being informed by both the input sequence and the tokens previously generated. This distinctive structure excels in tasks like text generation, translation, and summarization, where creativity and contextual relevance are paramount. The decoder-only approach stands out by ensuring that each generated token builds on the context established by its predecessors, creating coherent and nuanced outputs as shown in the following algorithm.

```
Algorithm 2: ChatGPT's decoder-only architecture and RLHF for translation
Step 1: Autoregressive decoding
$X = \{x_1, x_2, \dots, x_n\}$ is the English input sequence tokens and Y$\{y_1, y_2, \dots, y_n\}$ are the target tamil translation tokens generated by the model.
For each t from 1 to m,
    $y_t = Decoder(y_t, X)$
Step 2: Parallel processing
M is the model with L layers. For each layer l, processing is distributed as:
    $M_l = \left(M_l^{Data}, M_l^{Pipeline}, M_l^{Tensor}\right)$
Step 3: Supervised fine-tuning
Minimize cross entropy loss over the dataset $D = \{(X_i, Y_i)\}$, where each $(X_i, Y_i)$ pair represents a supervised example.
    $L_{supervised} = \sum_{(X_i,Y_i)\in D} \sum_{t=1}^{m} logP\left(y_{i,t} \vee y_{i,<t}, X_i\right)$
Step 4: Reward model training
Create reward score $R(Y)$ based on human feedback for each output Y.
Step 5: PPO update
Optimize $L_{PPO}$ using rewards for human-aligned translations
    $L_{PPO} = E\left[min\left(\frac{\pi_\theta(Y|X)}{\pi_{\theta_{old}}(Y|X)} R(Y), clip\left(\frac{\pi_\theta(Y|X)}{\pi_{\theta_{old}}(Y|X)}, 1-\in, 1+\in\right) R(Y)\right)\right]$
where $\pi_\theta(Y|X)$ is the updated policy and $\pi_{\theta_{old}}(Y|X)$ is the previous policy. The parameter $\in$ controls the extent of clipping.
```

While LLM excels at many tasks, there are chances to provide incorrect translations or biased content, especially in a language like Tamil. Hence alignment with human values is critical. To ensure model's alignment in translation task Reinforcement learning with human feedback (RLHF) can be employed. Here human feedback is used to fine tune the model [12]. This ensures that the translation is not only grammatically correct but also contextually correct and appropriate. As a first step in RLHF, human annotators provide high-quality translation from English to Tamil. The model uses supervised data to learn the correct translation patterns. Next is reward model training. In this phase the human annotators compare various outputs generated by the model. The accuracy and appropriateness are evaluated [13]. Thus this feedback can train the model on which output generated is most suitable with human preferences. The last is reinforcement Learning, using proximal policy optimization (PPO). Based on the feedback collected the model is further fine-tuned. This iterative process ensures that the model reaches its better version and match human expectation. All these integrated approaches allow the model understand the intricate structure of the Tamil language, including its verb-final sentence structure.

2.3 Gemini (Google)

The Gemini LLM for English to Tamil translation shows some significant advancements in machine translation, especially for a low-resource language like Tamil [14]. The model uses transformer which adds more power to the model, by handling complex dependencies and contextual relationships in sequences.

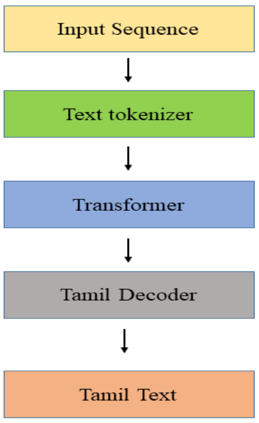

Fig.4 Pipeline for Translating Text to Tamil Using Transformer Model

Figure 4 illustrates a pipeline designed for translating text into Tamil using a transformer model. The translation process starts with the Input Sequence, which is the text to be translated. This text is then processed by a Text Tokenizer that breaks it down into smaller units, or tokens, making it easier for the model to handle. These tokens are fed into a Transformer, a sophisticated neural network model that processes and understands the context and semantics of the input text. The output from the transformer is then passed to a Tamil Decoder, which converts the processed tokens into a sequence of Tamil language tokens. Finally, these tokens are reassembled into coherent Tamil Text, producing the translated output. This pipeline leverages the transformer model's capabilities to ensure accurate and contextually appropriate translations into Tamil as represented in algorithm 3.

Algorithm 3: Gemini LLM algorithm for English-to-Tamil translation

Step 1: Tokenization

$$T_X = Tokenizer(X)$$

where $X = \{x_1, x_2, \dots, x_n\}$ is the English input sequence tokenized into smaller units like words, sub words or characters resulting in tokens $T_X = \{t_1, t_2, \dots, t_k\}$.

Step 2: Transformer encoding

For each token t_i in T_X, multi head self-attention and feed forward layers are applied by the transformer for the generation of contextual embeddings.

$$H = TransformerEncoder(T|X)$$

Where $H = \{h_1, h_2, \dots, h_k\}$ is the encoded representation for each token T_X.

Step 3: Tamil decoding

$$y_t = TamilDecoder(H, y_t)$$

for each position $t = 1, \dots, m$ in the target sequence. y_t is the previously generated Tamil tokens. H is the encoded information used by the decoder to understand the English context while generating each Tamil token y_t.

Step 4: Reassembly

$$TamilText = Reassembly(Y)$$

The Tamil tokens Y are reassembled to coherent Tamil text.

The pipeline starts with an input sequence and ends with the output of the required Tamil text. After providing the input sequence, it undergoes tokenization via test tokenizer. The tokenizer plays a major role in bridging the gap between English and Tamil [15]. This break the English text into smaller words, sub words or characters ensuring that the input is in correct format for the transformer to process. The transformer is the main engine behind the process of translation. It first grasps the contextual meaning by attending different part of the sentence and then handles the intricate grammatical rules that vary between English and tamil. This makes the model to understand and preserve the tense, gender and politeness while translating. The Tamil decoder is responsible for generating the Tamil text by translating the encoded tokens into Tamil. This step verifies whether the generated output matches the meaning. The Gemini LLM is trained to work with wide variety of sentence structure, expressions and colloquialism [16]. Its ability to fine-tune improves the model's proficiency in generating accurate translations even for idiomatic expressions, complex sentences, and technical words [17]. When compared to traditional LLM model like rule based or statistical, Gemini offers best performance because of the transformer based approach [18]. This makes the model to be a cutting-edge tool in the field of bilingual translation.

3. Result and Discussion

A custom dataset was created to assess the translation of politically sensitive English content into Tamil by Large Language Models (LLMs), including Claude, ChatGPT, and Gemini. This dataset, structured as a frame, covers various political references, formal and informal tones, and culturally nuanced language to challenge each model's translation accuracy and sensitivity. The models' outputs were evaluated using Bilingual Evaluation Understudy Score (BLEU), Bidirectional Encoder Representations from Transformers (BERT) based scores, Metric for Evaluation of Translation with Explicit Ordering (METEOR), and Translation Edit Rate (TER) [19].

BLEU is a widely adopted metric that examines the overlap in n-grams between the translated text and reference translations. While this approach prioritizes precision, it doesn't always capture more nuanced aspects of meaning. It uses a modified precision formula and applies a brevity penalty to avoid favoring shorter translations.

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} \omega_n log p_n) \tag{1}$$

Here, p_n is the precision for n-grams (usually up to 4-grams). ω_n is weight assigned to different n-grams, typically equal for 1- to 4-grams. Brevity Penalty (BP) compensates for short translations, calculated as:

$$BP = \begin{cases} 1 if c > r \\ e^{\left(1-\frac{r}{c}\right)} if c \leq r \end{cases} \tag{2}$$

where c is the length of the candidate translation and r is the length of the closest reference translation.

BERT-based scores are able to assess semantic similarity in a more holistic way, going beyond just lexical matching to consider contextual factors. This can provide deeper insights into the overall coherence and fluency of a translation. These vectors (+) are then compared for similarity using cosine similarity. For a candidate translation C and a reference translation R, BERT Score computes:

$$BERTScore = \frac{1}{|C|} \sum_{c \in C} \max_{r \in R} cosine_{similarity}(c, r) \tag{3}$$

$$cosine_{similarity}(c, r) = \frac{c \cdot r}{\|c\| \|r\|} \tag{4}$$

METEOR, which builds on BLEU by also accounting for synonyms, word stems, and word order through a combination of precision and recall calculations. The result is a score between 0 and 1 that offers a more well-rounded assessment. It uses a harmonic mean of precision and recall with a higher weight on recall. F-mean is the harmonic mean of precision and recall. Penalty is estimated for incorrect word order. Chunks are contiguous matching phrases and matches are all matching words between the candidate and reference.

$$METEOR = F_{mean} \cdot (1 - Penalty) \tag{5}$$

$$F_{mean} = \frac{10 \times Precision \times Recall}{9 \times Precision \times Recall} \tag{6}$$

$$Penalty = 0.5 \times \left(\frac{chunks}{matches}\right)^3 \tag{7}$$

TER focuses on the amount of editing required to transform the machine-generated translation into an acceptable final product. Rather than n-gram overlap, TER looks at insertions, deletions, substitutions, and reorderings, with lower scores indicating higher quality. Number of edits includes the total operations (insertion, deletion, substitution, shift) needed for alignment with the reference. The average number of words in reference normalizes TER to the length of the reference.

$$TER = \frac{Number of edits}{Averag e number of words \in reference} \tag{8}$$

The dataset for this study was sourced from GitHub, selected for its high-quality, community-verified content relevant to the research objectives [20, 22]. It includes diverse text samples necessary for tasks like text normalization, tokenization, and script conversion. The dataset was preprocessed using several essential libraries. Text normalization was carried out using re for regular expressions, and script information was processed with langdetect and langid. Word tokenization and detokenization were handled by nltk and spaCy, while sentence splitting relied on nltk. Word segmentation and syllabification were performed with polyglot and syllapy. Script conversion and Romanization were done using indic-transliteration and unidecode, and Indicization was achieved through indic-nlp-library. These pre-processing steps were applied to ensure the dataset was structured and clean, facilitating high-quality input for model training and testing.

The corpus has text from several domains, enabling the evaluation of the models' ability to handle various genres. The text like poems, stories and literary text must be captured and translated correctly along with presence of literary style. Technical documents like manuals, academic papers and instructional content challenges the model to accurately translate the domain specific terms. The model must also adaptable to everyday language like dialogues, informal expressions, social media post and conversational text. The corpus also contains text with basic vocabulary, advanced vocabulary, idiomatic expressions and intricate syntactic structures making the model to handle the linguistic features. Idiomatic expressions with figurative meanings, which differ between English and Tamil, assess the models' ability to capture deeper meaning beyond literal translation. Furthermore, texts were designed to test the models' handling of Tamil's agglutinative structures, where words are formed by adding affixes to base words. Importantly, the selected texts were translated by the models without any fine-tuning or post-processing, ensuring an accurate evaluation of their general-purpose translation capabilities.

Table.1 Comparison of Translation Metrics for Idiomatic Expressions

Metric	Claude	ChatGPT	Gemini
BLEU N-gram			
1-gram BLEU score	0.0235	0.0215	Failed to Translate
2-gram BLEU score	0.0011	0.0010	Failed to Translate
3-gram BLEU score	0.0004	0.0004	Failed to Translate
4-gram BLEU score	0.0003	0.0002	Failed to Translate
BERT Scoring			
BERT Precision	0.7303	0.7085	Failed to Translate
BERT Recall	0.7341	0.7245	Failed to Translate
BERT F1 Score	0.7318	0.7160	Failed to Translate
METEOR Score	0.1212	0.0980	Failed to Translate
TER Score	191.00	219.00	Failed to Translate

Table.1 evaluates the three LLMs for translating idiomatic expressions from English to Tamil. the key metrics involved are BLEU scores (for n-gram precision), BERT-based metrics (precision, recall, F1), METEOR, and TER. These metrics helps in understanding how well the models can generate output in terms of accuracy, fluent, and contextually appropriate translations. Three models, Claude, ChatGPT, and Gemini, are compared to show their strength and weakness in handling bilingual translation tasks. With a METEOR score of 0.1212 and a BERT F1 score of 0.7318, Claude's translations were closer to human quality. ChatGPT, although showing reasonable performance, lagged slightly behind Claude in all metrics, particularly in fluency and semantic measures such as METEOR (0.0980) and BERT scoring. In contrast, Gemini failed to translate the idiomatic expressions entirely, yielding no measurable scores for any of the metrics and highlighting its limitations in handling complex linguistic constructs.

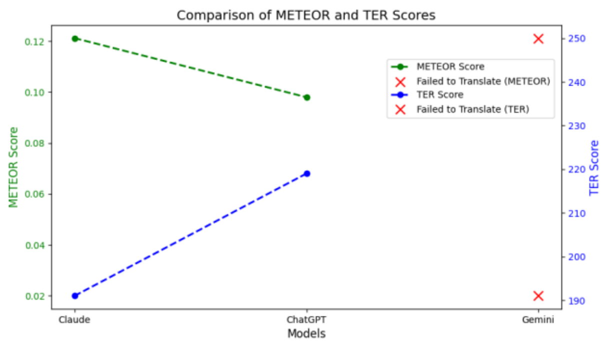

Fig.6 Combination graph for METEOR and TER Scores

Fig. 6 illustrates the comparison of two key metrics such as METEOR and TER scores, for three different LLM namely Claude, ChatGPT, and Gemini for idiomatic expressions. METEOR scores evaluates the similarity between machine-generated translations and reference translations, where higher score indicated greater performance. TER refers the number of edits made to the match, where lower score indicates the favorable condition. The TER score analysis reinforces Claude's superiority, as it required fewer edits (191) compared to ChatGPT (219). Overall, the comparison identifies Claude as the most effective model for translating idiomatic expressions, with ChatGPT showing room for improvement and Gemini needing substantial development to address such translation challenges.

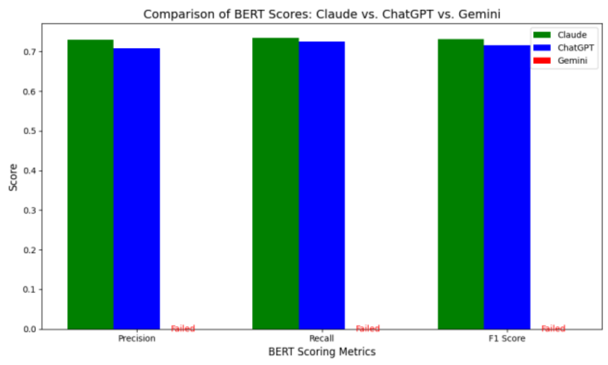

Fig.7 BERT Comparison Across Models

Fig.7 presents the comparison of BERT scores which includes precision, recall, and F1-score across three LLMs for idiomatic expressions. BERT metrics evaluate how well the models perform based on the correctness (precision), completeness (recall), and a harmonic mean of both (F1-score). Claude and ChatGPT demonstrate competitive performance on the BERT evaluation metrics, with Claude being slightly better in Precision and Recall, and ChatGPT excelling slightly in F1 Score. Gemini, however, is marked as unusable for this task, highlighting that its failure to translate idiomatic expressions makes it ineffective for this particular task.

Table 2: Comparison of Translation Evaluation Metrics for Technical Terminology

Metric	Claude	ChatGPT	Gemini
BLEU N-gram			
1-gram BLEU score	0.0096	0.0106	0.0104
2-gram BLEU score	0.0004	0.0004	0.0004
3-gram BLEU score	0.0001	0.0001	0.0001
4-gram BLEU score	0.0001	0.0001	0.0001

ROUGE L Scores			
F1 Score	0.0497	0.0483	0.0502
Precision	0.0497	0.0483	0.0502
Recall	0.0497	0.0483	0.0502
TER Score	786.00	842.00	778.00
BERT Scoring			
BERT Precision	0.5284	0.6996	0.5992
BERT Recall	0.5179	0.6998	0.6038
BERT F1 Score	0.5228	0.6993	0.6012
METEOR Score	0.0053	0.0048	0.0054

Table 2 provides a comparative analysis of the performance of Claude, ChatGPT, and Gemini across various translation evaluation metrics for technical terminology, including BLEU N-gram scores, ROUGE L scores, TER scores, BERT scoring, and METEOR scores. While all models perform similarly in surface-level metrics like BLEU and ROUGE, ChatGPT demonstrates better semantic understanding based on BERT scoring, and Gemini shows efficiency in producing translations with fewer necessary edits, as indicated by its lower TER score.

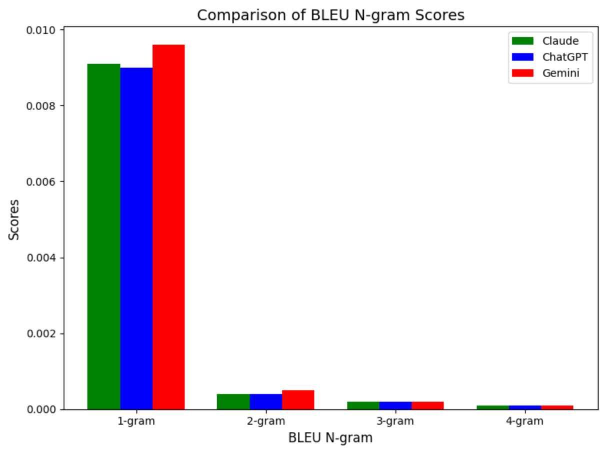

Fig.8 Comparison of BLEU scores

Fig. 8 illustrates the BLEU score comparison across Claude, ChatGPT, and Gemini models for n-gram levels from 1-gram to 4-gram when analyzing cultural references. Here, Gemini achieves the highest 1-gram BLEU score, closely followed by Claude and ChatGPT, suggesting that Gemini performs slightly better in capturing single-word matches, which may include common cultural keywords. However, as we move to higher n-grams (2-gram, 3-gram, and 4-gram), the BLEU scores drop significantly for all models, with minimal differences among them. This decline indicates a challenge in precisely matching longer cultural phrases or idiomatic expressions, which are often complex and context-dependent.

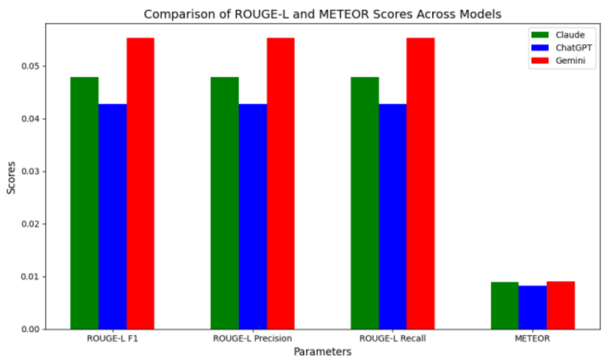

Fig. 9 ROUGE L and METEOR scores

Fig. 9 shows the comparison of ROUGE L (Recall-Oriented Understudy for Gisting Evaluation) and METEOR scores across the same models for handling cultural references. In the ROUGE L metrics (F1, Precision, and Recall), Gemini again outperforms both Claude and ChatGPT, with particularly notable gains in the F1 and Recall scores. This suggests Gemini's slight advantage in capturing broader cultural similarities in phrases and sentence structures. However, the METEOR scores remain low across all three models, implying challenges in capturing synonyms and semantic variations specific to cultural references. This low METEOR score may highlight the models' limitations in understanding culturally nuanced language, as METEOR evaluates both exact and semantically similar matches, which are critical for cultural expressions.

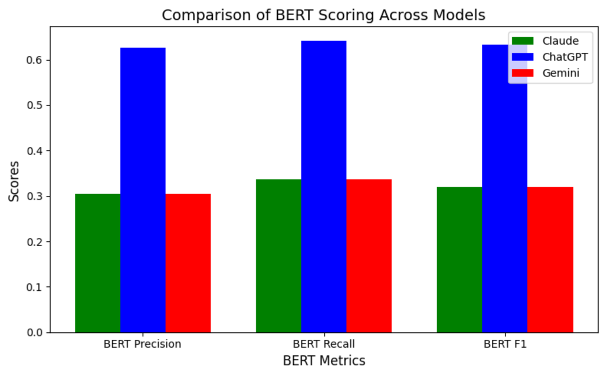

Fig. 10 ROUGE and METEOR scores comparison

Fig. 10 presents a comparison of BERT and TER scores across Claude, ChatGPT, and Gemini models for cultural references. The BERT scores (Precision, Recall, and F1) reflect each model's ability to capture semantic similarity with reference text, which is essential for understanding cultural nuances. ChatGPT shows a notable advantage in BERT scores, especially in Recall and F1, indicating its proficiency in grasping semantically similar cultural references across a broader range of phrases. Claude and Gemini perform similarly in this metric, suggesting they capture fewer nuanced or culturally specific references than ChatGPT. On the other hand, the TER score, which measures the number of edits needed to make model outputs align with reference text, shows similar values across all models. The TER scores are 490.00 for Claude, 513.00 for ChatGPT, and 488.00 for Gemini. Lower TER scores would indicate fewer discrepancies, but here, the TER scores are relatively high, implying that all models face challenges in achieving high alignment with cultural references, likely due to the variability in language and expressions tied to cultural context.

Table.3 BLEU N-gram Scores

Model	1-gram BLEU	2-gram BLEU	3-gram BLEU	4-gram BLEU
Claude 3.5 Sonnet	0.0052	0.0002	0.0001	0.0000
ChatGPT 4o	0.0057	0.0002	0.0001	0.0001
Gemini 1.5 Pro	0.0054	0.0002	0.0001	0.0000

Table 3 presents the BLEU scores for three language models—Claude, ChatGPT, and Gemini—across different n-gram levels (1-gram to 4-gram) for political references. The 1-gram BLEU score for ChatGPT is slightly higher than Claude and Gemini, indicating a marginally better match with reference translations at the unigram level. However, the scores drop significantly as the n-gram level increases, with all models showing minimal improvements at higher levels (2-gram to 4-gram). This suggests that while the models perform similarly in unigram comparison, their ability to generate longer, more accurate sequences diminishes, indicating limited fluency and context capture in multi-gram evaluation.

Table. 4 ROUGE L and TER Scores

Model	ROUGE L F1 score	ROUGE L Precision	ROUGE L Recall	TER Score
Claude 3.5 Sonnet	0.0501	0.0501	0.0501	1040.00
ChatGPT 4o	0.0476	0.0476	0.0476	938.00
Gemini 1.5 Pro	0.0473	0.0473	0.0473	1012.00

Table 4 shows the ROUGE and TER scores for Claude, ChatGPT, and Gemini models, evaluating their translation quality for political references. ROUGE measures the overlap of generated text with a reference, using metrics like F1 score, precision, and recall, to gauge similarity in content and phrasing. Claude achieves the highest ROUGE scores (0.0501) across F1, precision, and recall, indicating the closest alignment with reference content. ChatGPT and Gemini show slightly lower scores. TER scores, which assess edit distance, reveal Claude as having the highest score, suggesting more edits are needed compared to ChatGPT and Gemini.

Table. 5 BERT Scoring and METEOR Scores

Model	BERT Precision	BERT Recall	BERT F1 Score	METE0R Score
Claude 3.5 Sonnet	0.4967	0.5173	0.0564	0.0041
ChatGPT 4o	0.5436	0.5600	0.5513	0.0044
Gemini 1.5 Pro	0.4870	0.5089	0.4974	0.0042

Table 5 compares BERT-based scores (precision, recall, and F1) and METEOR scores for the Claude, ChatGPT, and Gemini models, assessing their semantic and contextual alignment with political reference text. BERT scores are derived from embeddings, measuring similarity in meaning rather than just word overlap. ChatGPT achieves the highest BERT Precision (0.5436), Recall (0.5600), and F1 Score (0.5513), suggesting it more effectively captures semantic similarity. Gemini and Claude trail behind, with Gemini slightly outperforming Claude on these metrics. The METEOR scores, which consider exact matches, stems, synonyms, and order, are low overall, with ChatGPT achieving the highest score (0.0044), though differences are minimal.

In assessing English to Tamil translation, each model excels in distinct aspects, demonstrating specific achievements. ChatGPT showcases strong semantic understanding and adaptability, making it highly capable of capturing contextual nuances and conveying meaning accurately. Its high scores in BERT-based metrics reflect its proficiency in delivering translations that closely align with the intended meaning, making it a strong choice for tasks requiring accurate cultural and contextual translation. Gemini stands out in its ability to retain content fidelity, achieving high ROUGE and METEOR scores. This achievement makes it particularly effective in scenarios where retaining original text structure and meaning is crucial, such as legal or formal translations. Although Claude requires more edits (as indicated by TER), it excels in capturing surface-level text alignment and single-word accuracy, demonstrated by its 1-gram BLEU score. This makes Claude suitable for simpler translations where exact word matches are prioritized. Overall, ChatGPT is best for nuanced context, Gemini for structured fidelity, and Claude for straightforward translation accuracy.

While manual translation relies on human expertise to accurately convey meaning, context, and cultural nuances, machine translation uses algorithms that may struggle with idiomatic expressions and cultural references. In comparing models like Claude, ChatGPT, and Gemini, Claude excels in technical terminology. ChatGPT performs better in context-based tasks with higher BERT scores, and Gemini struggles with idiomatic or cultural translations. The manual translation is more accurate for complex, context-heavy content, whereas MT works well for straightforward, technical material. Claude is ideal for technical content, ChatGPT suits general content, and Gemini performs best with structured, non-contextual material.

4. Conclusion and future work

In conclusion, each Large Language Model (LLM) exhibits unique strengths in English-to-Tamil translation, reflecting their diverse technological strategies. Claude uses prompt tuning, batch processing, and temperature control to improve coherence in document-level translations, making it suited for simpler, surface-level alignments. ChatGPT utilizes a decoder-only architecture combined with Reinforcement Learning from Human Feedback (RLHF) to capture nuanced cultural and contextual meanings, producing high BERT and METEOR scores. Gemini employs advanced tokenization, attention mechanisms, and a transformer-based framework that excels at preserving Tamil's complex syntax, tense, and politeness markers, ideal for technical and idiomatic content. Future research should focus on integrating the strengths of these models to create a hybrid system that combines ChatGPT's contextual depth and Gemini's structural accuracy, optimizing both fluency and precision. Leveraging more sophisticated fine-tuning techniques and exploring cross-lingual transfer learning could further improve performance in low-resource languages. This approach would help bridge the gap in translation quality, ensuring more accurate, culturally sensitive, and fluent translations for complex linguistic tasks.

References

[1] Wang, H., Wu, H., He, Z., Huang, L., & Church, K. W. (2022). Progress in machine translation. *Engineering*, *18*, 143-153.

[2] Stahlberg, F. (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, *69*, 343-418.

[3] Rajendran, S., Anand Kumar, M., Rajalakshmi, R., Dhanalakshmi, V., Balasubramanian, P., & Soman, K. P. (2022, November). Tamil NLP Technologies: Challenges, State of the Art, Trends and Future Scope. In *International Conference on Speech and Language Technologies for Low-resource Languages* (pp. 73-98). Cham: Springer International Publishing.

[4] Mager, M., Mager, E., Kann, K., & Vu, N. T. (2023). Ethical considerations for machine translation of indigenous languages: Giving a voice to the speakers. *arXiv preprint arXiv:2305.19474*.

[5] Cui, M., Du, J., Zhu, S., & Xiong, D. (2024). Efficiently Exploring Large Language Models for Document-Level Machine Translation with In-context Learning. *arXiv preprint arXiv:2406.07081*.

[6] Wang, L., Du, Z., Jiao, W., Lyu, C., Pang, J., Cui, L., ... & Tu, Z. (2024, August). Benchmarking and improving long-text translation with large language models. In *Findings of the Association for Computational Linguistics ACL 2024* (pp. 7175-7187).

[7] Wang, Y., Zhang, Q., Li, K., Tang, Y., Chen, J., Luo, X., & Chen, T. (2021, August). iBatch: saving Ethereum fees via secure and cost-effective batching of smart-contract invocations. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 566-577).

[8] Ouyang, S., Zhang, J. M., Harman, M., & Wang, M. (2024). An Empirical Study of the Non-determinism of ChatGPT in Code Generation. *ACM Transactions on Software Engineering and Methodology*.

[9] Ouyang, S., Zhang, J. M., Harman, M., & Wang, M. (2024). An Empirical Study of the Non-determinism of ChatGPT in Code Generation. *ACM Transactions on Software Engineering and Methodology*.

[10] Shao, M., Basit, A., Karri, R., & Shafique, M. (2024). Survey of different Large Language Model Architectures: Trends, Benchmarks, and Challenges. *IEEE Access*.

[11] Liang, P., Tang, Y., Zhang, X., Bai, Y., Su, T., Lai, Z., ... & Li, D. (2023). A survey on auto-parallelism of large-scale deep learning training. *IEEE Transactions on Parallel and Distributed Systems*, *34*(8), 2377-2390.

[12] Wang, L., & Xiao, Y. (2024). Improving Low-Resource Machine Translation Using Reinforcement Learning from Human Feedback. *Intelligent Automation & Soft Computing*, *39*(4).

[13] Howcroft, D. M., Belz, A., Clinciu, M., Gkatzia, D., Hasan, S. A., Mahamood, S., ... & Rieser, V. (2020, December). Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *13th International Conference on Natural Language Generation 2020* (pp. 169-182). Association for Computational Linguistics.

[14] Ahuja, P. (2024). Beyond Numbers: A Textual Voyage Into Finance With Gemini. *Available at SSRN 4833462*.

[15] HO, L. (2024). English of Tamil Multi-Modal Neural Machine Translation for Image Captioning. *Grenze International Journal of Engineering & Technology (GIJET)*, *10*.

[16] Hakkarainen, S., & Engelbrecht, K. (2024). Automatic Idiomatic Expression Detection. Comparison Between GPT-4 and Gemini Pro Prompt Engineering & LSTM-RNN Construction.

[17] Pang, Z., Xie, Z., Man, Y., & Wang, Y. X. (2023). Frozen transformers in language models are effective visual encoder layers. *arXiv preprint arXiv:2310.12973*.

[18] Herencia López-Menchero, A. (2024). Analysis of the Transformer Architecture and application on a Large Language Model for mental health counseling.

[19] Lavecchia, A. (2024). Advancing drug discovery with deep attention neural networks. *Drug Discovery Today*, 104067.

[20] Kunchukuttan, A. (n.d.). The IndicNLP Library [Dataset]. GitHub. https://github.com/anoopkunchukuttan/indic_nlp_library

[21] Khaleel Jagger. Tamil NLTK Ranking [Dataset]. GitHub. https://github.com/khaleeljageer/TamilNLTKRanking