Word-Level Machine Translation for English-Telugu Using Encoder-Decoder and Transformer-Based Models

Introduction

Problem Statement

Machine translation plays a key role in bridging the gaps between languages, especially for resource-rich languages like English and low-resource, morphologically complex languages like Telugu. Indeed, developing effective translation systems for such a language pair is challenging since there are significant differences in grammar, vocabulary, and syntactic structures.

This project focuses on the development and improvement of MT at English↔Telugu. First, an encoder-decoder architecture at the word-level tokenization was implemented. The performance for this approach is analyzed with the BERTScore metric in detail, especially with respect to sentence length. After finding the deficiencies of the observed approach, a transformer-based mBART model was used.

Literature Review

1.Word-Level Tokenization

Word-level tokenization represents one of the most salient approaches in NLP tasks, especially in machine translation. As opposed to character-level tokenization, which segments text into single characters, in word-level tokenization, each word is treated as a unit of representation. The semantic and syntactic contexts are preserved better this way, and performance in sentence-level understanding tasks improves significantly.

- Advantages:
  1. Semantic Representation: Words naturally possess richer semantic meaning than individual characters, and models can pick up the context more effectively.
  2. Reduced Sequence Length: Tokenization at the word level generates shorter sequences compared to character-level tokenization, reducing computational overhead.
  3. Language-Specific Context: In the case of morphologically rich languages like Telugu, word-level tokenization is more efficient in representing suffixes and prefixes compared to character-level tokenization.
- Challenges
  1. OOV Issues: The word-level tokenization performs poorly for unseen words or rare words, especially in low-resource languages like Telugu.
  2. Data Requirements: Pre-training requires large and diverse datasets to cover the vocabulary well enough.

Works such as S. Hochreiter and J. Schmidhuber. [1] have shown that word-level tokenization does much better in translation tasks compared to character-level since the former is more capable of encoding better linguistic contexts.

2.Encoder-Decoder Models

The encoder-decoder architecture, introduced in sequence-to-sequence models, has been the cornerstone of neural MT. It consists of two components:
1. Encoder: It takes in an input sequence and generates a fixed-length context vector.
2. Decoder: It produces the output sequence based on the context vector.

Bidirectional LSTMs in Encoder:
Bidirectional LSTM layers in the encoder capture the contextual information from both the historical (left-to-right) and prospective (right-to-left) directions as in S. Hochreiter and J. Schmidhuber. [1]. It enhances the model's capture of word dependencies by such a bidirectional approach.
LSTMs in the Decoder:
The decoder consists of a unidirectional LSTM which generates the target sequence. The output at each step is dependent on the context vector and the previous word.
Applications in MT:
Encoder-decoder architectures have been widely used in MT, especially for low-resource languages. For example:

- Luong et al. [4] proposed attention mechanisms for encoder-decoder models, enabling better focus on relevant input words during translation.
- Bahdanau et al. [5] introduced an attention-based encoder-decoder model that alleviates the fixed-length context vector bottleneck, improving translation quality for longer sentences.

Limitations:
1. Fixed-Length Context Vector: Limits performance on long input sequences due to information loss.
2. Training Complexity: This involves the proper tuning of hyperparameters to avoid overfitting and vanishing gradients.
Other recent developments in overcoming these limitations have involved attention mechanisms, transformer layers, and pretraining.

Transformer Models

Transformers, introduced by Vaswani et al. [3], revolutionized sequence-to-sequence learning by replacing recurrent architectures with self-attention mechanisms. These models are especially good at capturing long-range dependencies, making them very suitable for machine translation.

- **Key Components**:
    1. **Self-Attention**:
        - Facilitates the model's capacity to evaluate the significance of distinct words within the input sequence in relation to one another, thereby capturing dependencies regardless of their separation.
    2. **Multi-Head Attention**:
        - Performs attention computation concurrently, enhancing the model's capability to concentrate on multiple sections of the input at the same time
    3. **Feedforward Neural Network**:

        - Implemented autonomously at each position, guaranteeing that the model acquires positional information via learned embeddings.
- **mBART (Multilingual Bidirectional and Auto-Regressive Transformers)**:
    - mBART [7] is a pre-trained transformer model for multilingual MT tasks. During pretraining, the model uses a denoising autoencoder objective that can effectively handle noisy or incomplete input sequences [6].
    - **Advantages**:
        - Utilizes pre-trained multilingual embeddings to reduce the need for large datasets.
        - Works very well on low-resource languages such as Telugu due to its cross-lingual training.
        - Self-attention mechanisms handle long sequences with ease.
- **Comparisons with Encoder-Decoder Models**:
    1. **Handling Long Sequences**:
        - Transformers outperform LSTMs in information retention across long sequences because of their attention-based architecture.
    2. **Parallelization**:
        - Unlike LSTMs, which process sequences sequentially, transformers allow parallel computation, reducing training time dramatically.
    3. **Pretraining Benefits**:
        - Pre-trained transformers, such as mBART, generalize better across languages, making them ideal for tasks with limited resources.
- **Challenges**:
    1. **Computational Resources**: Transformers require significant GPU memory and processing power.
    2. **Fine-Tuning**: Requires careful adaptation to domain-specific tasks for optimal performance.

Dataset(s) Used

Github Dataset

- **Size**: 0.14 million English-Telugu sentence pairs.
- **Domains**: General text, informal speech, and formal literature.
- **Preprocessing**: Lowercasing, punctuation removal, and word-level tokenization.

Preprocessing Steps:

Effective preprocessing is a very critical process in building accurate machine translation models. In this project, a preprocessing pipeline was designed that could handle both English and Telugu datasets with great care. The detailed steps are as follows:

Loading and Combining Datasets

The datasets were loaded from text files containing English and Telugu sentences. An English sentence was paired with its corresponding Telugu sentence to form a bilingual dataset. To ensure that the number of English and Telugu sentences matched, an assertion was used. Then, all the data was combined into a list of sentence pairs.

Cleaning the Data
Data cleaning for English and Telugu sentences was done separately by the following steps:

Cleaning of English Sentences:

- Convert all text to lowercase to maintain uniformity.
- Replace contractions using a predefined contraction mapping: "don't" → "do not".
- Remove special characters and punctuation marks to simplify input for the model.
- Removed numbers from sentences because they generally are not part of the semantics of the language.
- Removed extra spaces to maintain consistency in spacing.

Cleaning of Telugu Sentences:

- All text was converted to lowercase.
- Special characters and punctuation marks were removed.
- Stripped both English and Telugu numerals from the sentences.
- Extra spaces were removed to maintain consistency.
- Filtered out empty sentences after cleaning to retain only valid sentence pairs.

Dataset Statistics

After cleaning and filtering, the statistics of the dataset were as follows:

- **Number of valid sentence pairs**: **148441**
- **Maximum sentence length**:
  - English: **101** words.
  - Telugu: **27** words.

Exploring Sentence Lengths

To understand the distribution of sentence lengths, histograms were plotted for both English and Telugu sentences. The analysis showed that English sentences were generally longer than Telugu sentences, with the following maximum sentence lengths:

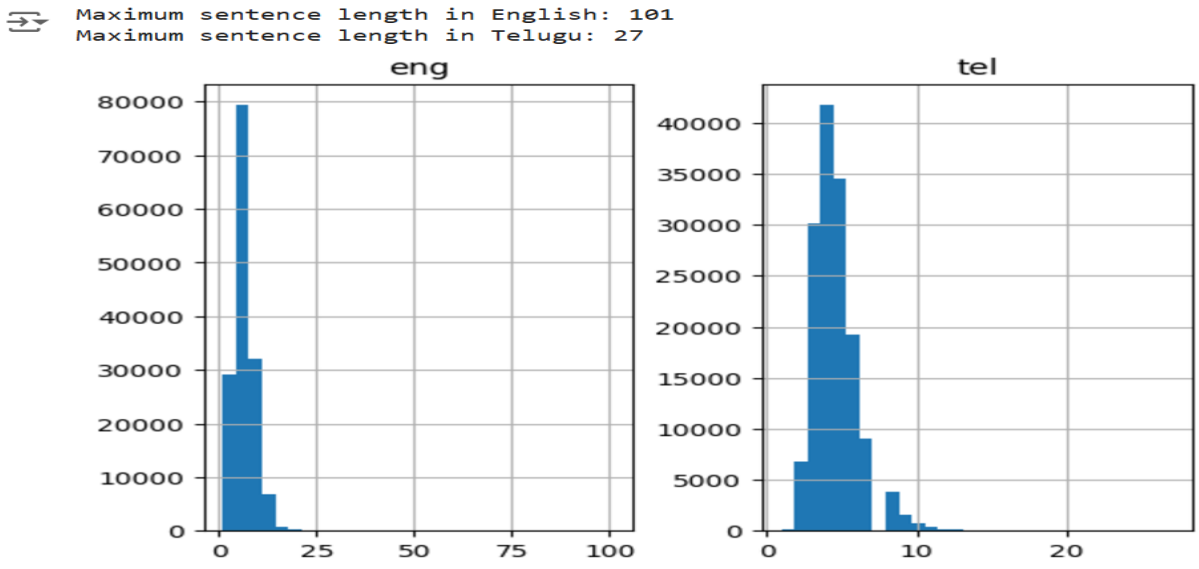


Fig.1 distribution of sentence's length

Tokenization

To prepare the text for model input, word-level tokenization was applied to both English and Telugu sentences:

1. **Creation of Tokenizer**: For both English and Telugu datasets, tokenizers were created using the Tokenizer class from Keras. The tokenizers were trained on the respective datasets for generating unique word indices.
2. **Vocabulary Size:**
   English Vocabulary: **13620** words.
   Telugu Vocabulary: **38055** words.

Sequence Encoding and Padding

After tokenization, the sentences were converted as sequences of integers or word indices. For the model to take input in the same dimension,

1. Sentences were padded with zeros to the maximum sentence length
2. Padding was applied at the end (post-padding) to maintain sequence integrity.

Dataset Split

The dataset was then split into training and testing in a 90:10 ratio. The training set would contain the data on which the model would be trained, while the test set would be kept for evaluation. For both English and Telugu sentences, encoded and padded sequences were created for splits:

- **Training Set**: 90% (Validation Set: 20% of Training Set)
- **Testing Set**: 10%

Methodology

Encoder-Decoder Architecture with Word-Level Tokenization

The first model uses an encoder-decoder structure with two Bidirectional LSTM layers in the encoder and two LSTM layers in the decoder. The model operates on word-level tokens.

Encoder:

- Two layers of Bidirectional LSTMs.
- Outputs a context vector summarizing the input sequence.

Decoder:

- Two layers of LSTMs.
- Utilizes the context vector to predict translations, word by word.

Hyperparameters:

- **Learning Rate**: 0.001

- **Batch Size**: 64
- **Embedding Dimension**: 300
- **Epochs**: 50
- **Optimizer**: Adam

Architecture Diagram:
Encoder-Decoder Model

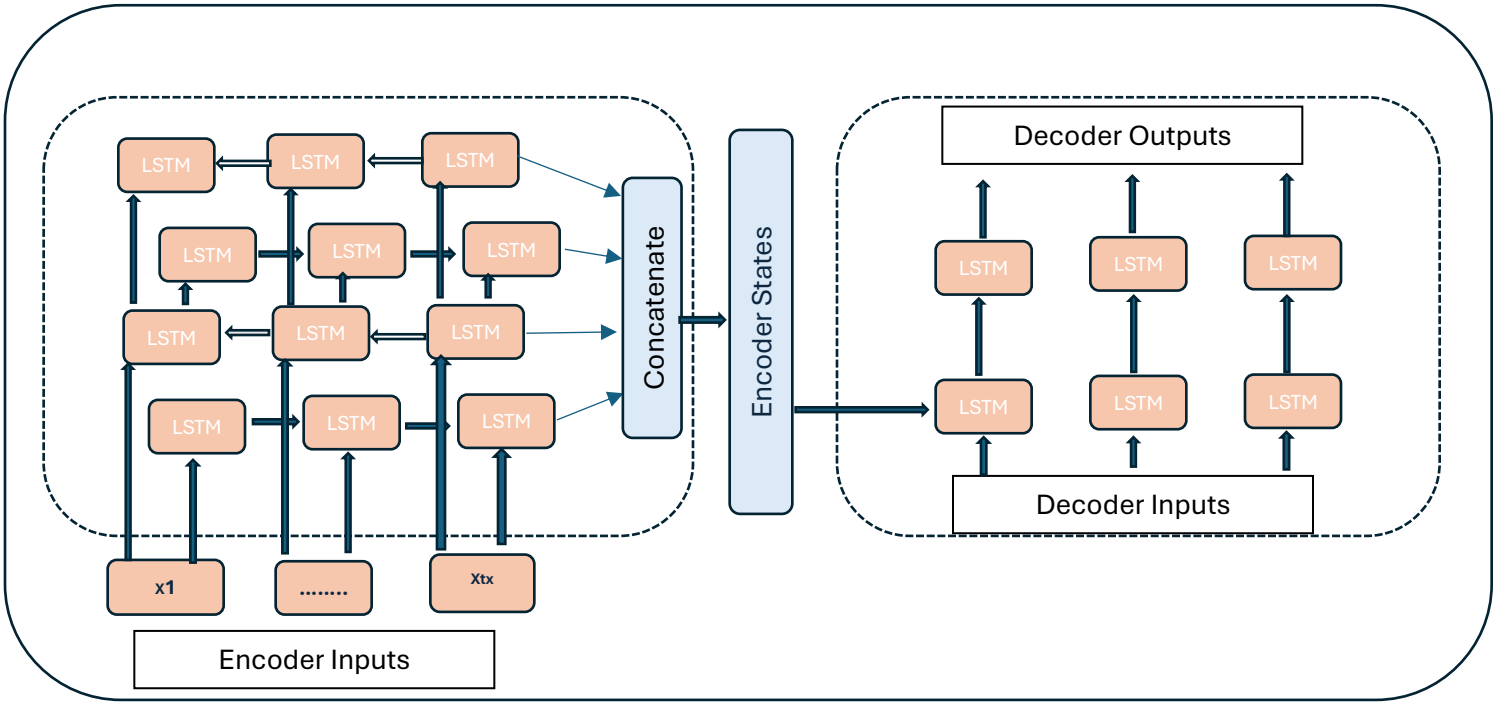


Fig.2 Architecture of Encoder-Decoder Model

Transformer-Based mBART Model

The mBART model was fine-tuned for English↔Telugu translation with six trainable layers:

- **Pretrained Weights**: Initialized with mBART's pretrained parameters.
- **Fine-Tuning**: Focused on the decoder layers for domain adaptation.

Advantages of mBART:

- Captures long-term dependencies efficiently.
- Requires less task-specific training data due to pretraining.

Transformer Based Model

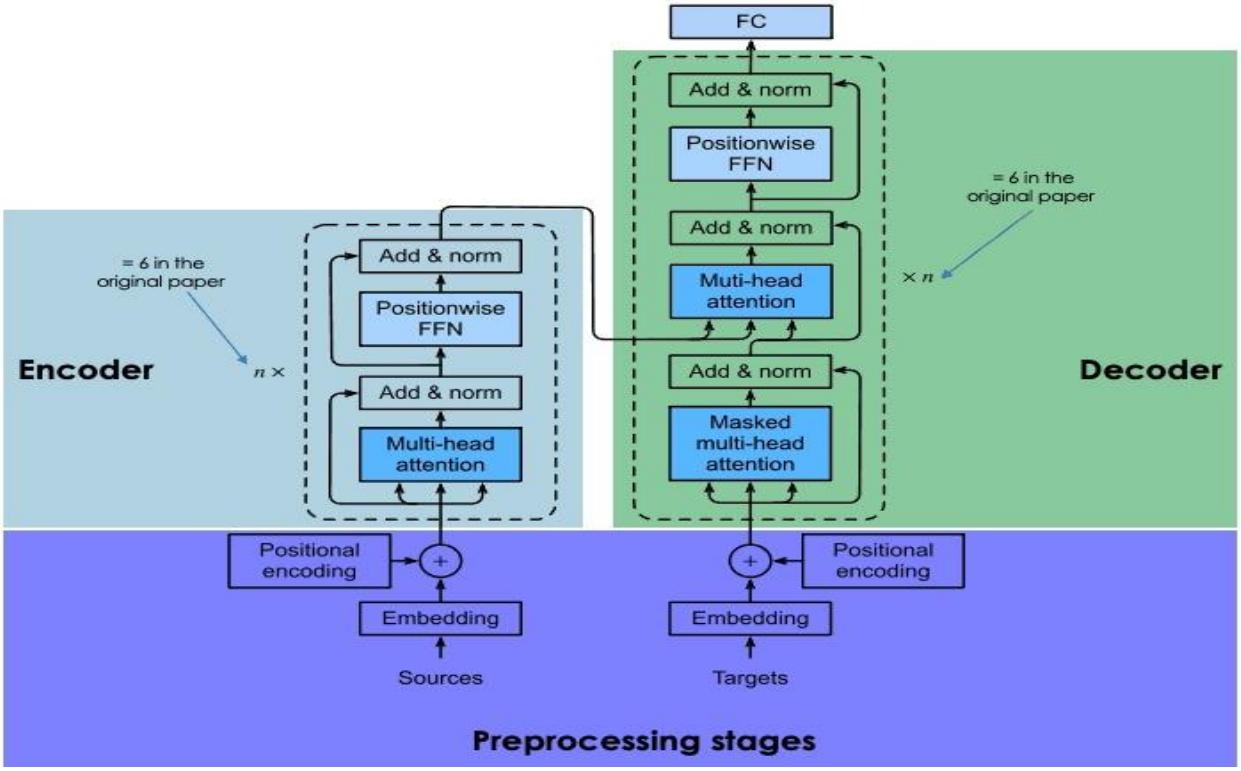


Fig.3 Architecture of mBART

Results

- **BERTScore Analysis for Word-Level Encoder-Decoder**:
    - English → Telugu:
        1. Average Precision: 0.8658
        2. Average Recall: 0.8425
        3. Average F1 Score: 0.8535
    - Telugu → English:
        1. Average Precision: 0.7180
        2. Average Recall: 0.7063
        3. Average F1 Score: 0.7126
- **Samples Test Sentence Translation along with their Bert Scores**

English → Telugu				
Prediction	Target	Precision	Recall	F1 Score
నేను నేనే చేయలేకపోయాను	నేనే చేయబోతున్నాను	0.865291	0.896360	0.880552

టామ్ నిద్రపోతున్నాడని నేను అనుకున్నాను	టామ్ నిద్రపోతున్నాడని నేను అనుకున్నాను	1.000000	1.000000	1.000000
నేను అలా నేను చెప్పాను	అలా చేయవద్దు అన్నారు	0.686422	0.671476	0.678867

Telugu → English				
Prediction	Target	Precision	Recall	F1 Score
i am going to do it myself	i am going to do it myself	1.0000	1.0000	1.0000
i wonder why you were are	i wonder why the dogs are barking	0.5037	0.4923	0.4988

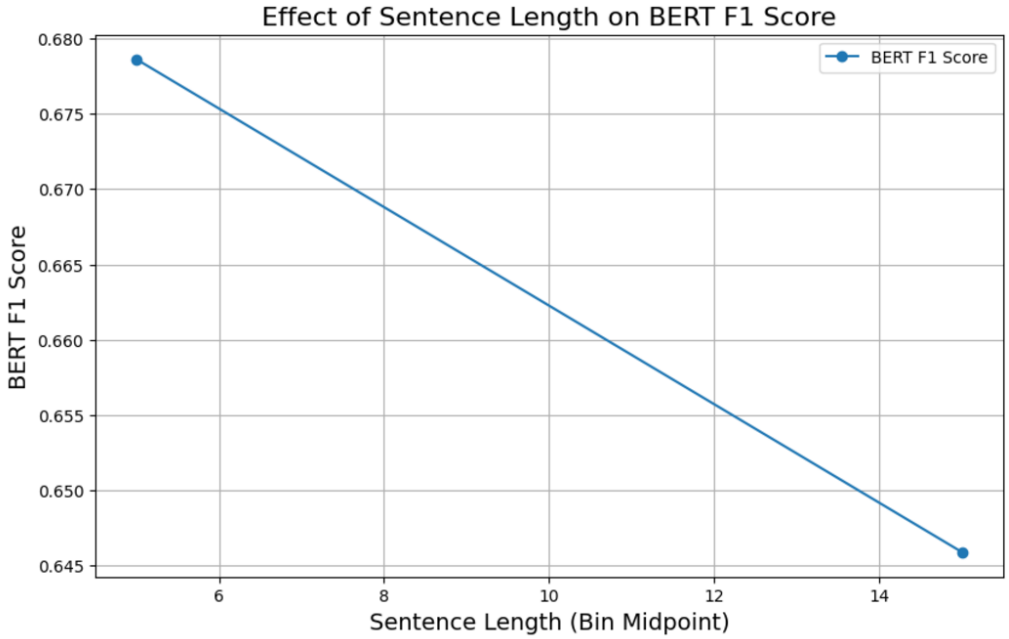


Fig 4. Effect on BERTscore with sentence Length

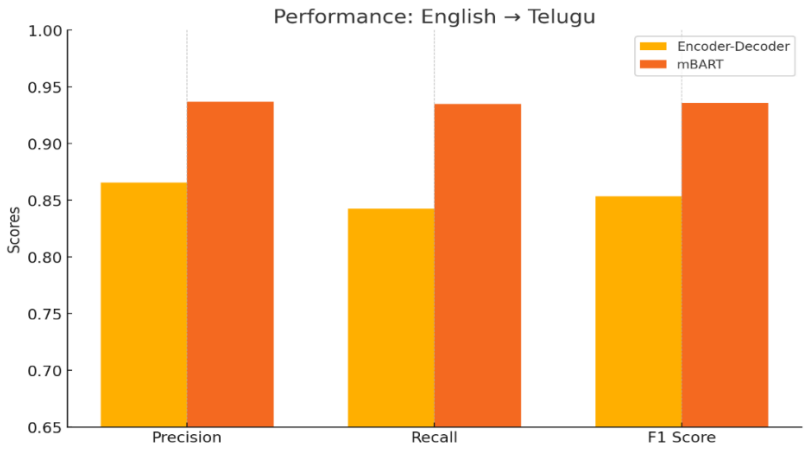
- Here we can see that as the sentence's length is increasing the average F1 score is decreasing. This can be due to the inability of LSTM's to capture long range dependencies.

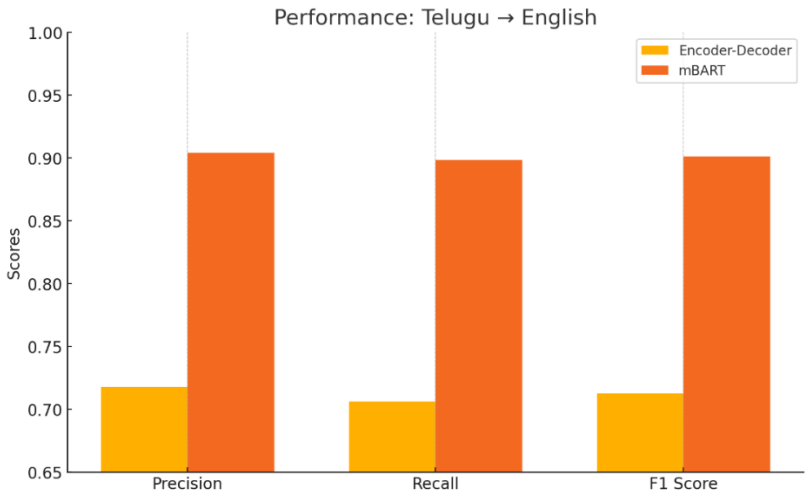
2. **BERTScore Analysis for mBART**:
    - English → Telugu:
        - Average Precision: 0.9369
        - Average Recall: 0.9350
        - Average F1 Score: 0.9358
    - Telugu → English:
        - Average Precision: 0.9043
        - Average Recall: 0.8987
        - Average F1 Score: 0.9014
3. **Samples Test Sentence Translation along with their Bert Scores**

English → Telugu				
Prediction	Target	Precision	Recall	F1 Score
'నేను ఆడబోతున్నాను'	'నేను ఆడబోతున్నాను'	1.000	1.000	1.000
మేము ఈ ప్రాజెక్ట్‌లో కొన్ని గొప్ప ఫలితాలను పొందాము...	మేము ఈ ప్రాజెక్ట్‌లో కొన్ని గొప్ప ఫలితాలను పొందాము	1.000	1.000	1.000

Telugu → English				
Prediction	Target	Precision	Recall	F1 Score
why dont you ever ask simples questions	why dont you ever ask simple questions	0.9654	0.9783	0.9718
his legs are longs	his legs are long	0.9536	0.9806	0.9669

Comparison:





- We can see that mBART performs better than the encoder-decoder model as mBART is a pretrained transformer and it can capture long range dependencies

---

Discussion

- The evaluation metrics we used is BERTscore which calculates the semantic similarity between two sentences. Sample translations provide valuable insights into the performance of the Word-Level Encoder-Decoder and mBART models. Below, we interpret the results, analyze the significance of the findings, and propose recommendations for future improvements.

Word-Level Encoder-Decoder Model:

- English → Telugu:
  - The Average Precision (0.8658), Recall (0.8425), and F1 Score (0.8535) indicate a reasonable performance for short to moderately complex sentences.
  - This aligns with the known limitations of LSTMs, particularly their difficulty in retaining contextual information across lengthy sequences.
- Telugu → English:
  - Performance metrics (Precision: 0.7180, Recall: 0.7063, F1: 0.7126) indicate challenges in translating from a morphologically rich language (Telugu) to a more syntactically simple language (English).
  - Sample analysis shows a notable drop in performance for sentences with significant semantic or structural differences.

Implications: The Word-Level Encoder-Decoder model relies heavily on the size and quality of the training corpus. While adequate for shorter sequences, the lack of context preservation over long ranges impacts its usability for more complex translations.

---

mBART Model:

- English → Telugu:
  - The Average Precision (0.9369), Recall (0.9350), and F1 Score (0.9358) indicate a substantial improvement over the Word-Level Encoder-Decoder.
  - The model consistently produces high-quality translations for both short and moderately complex sentences.
- Telugu → English:
  - Metrics (Precision: 0.9043, Recall: 0.8987, F1: 0.9014) indicate improved performance compared to the Encoder-Decoder model.
  - Sample translations highlight the model's capability to handle morphologically complex Telugu inputs.

Implications: The mBART model's pretraining on multilingual corpora provides it with a significant edge in handling contextual and semantic complexities. Its fine-tuning ability allows better generalization, especially for longer and semantically intricate sentences.

Key Observations:

1. **Word-Level Encoder-Decoder**:
   - Performs well on short and moderately long sentences.
   - Struggles with sentence-level dependencies in longer texts due to limitations of LSTMs.
   - Translation errors often stem from missing idiomatic or morphological nuances in Telugu.
2. **Transformer-Based mBART**:
   - Demonstrates robust performance across varying sentence lengths.
   - Excels in retaining context for longer sentences due to self-attention mechanisms.
   - Pretrained embeddings reduce the dependency on large domain-specific datasets.

Recommendations for Improvement:

1. **Hybrid Tokenization**: Experiment with sub word tokenization methods like Byte Pair Encoding (BPE) to balance sequence length and semantic retention.
2. **Attention Mechanisms in Encoder-Decoder**: Adding attention layers could help the LSTM-based architecture focus on relevant parts of the input sequence.
3. **Dataset Augmentation**: Expand the dataset with augmented or synthetic data for longer sentence pairs and add more diverse data.
4. **Unfreezing Layers**: Train the mBART by unfreezing more layers in the decoder and encoder as well as embedding layer with large dataset.

Conclusion

This project analyzed word-level tokenization for English↔Telugu translation using encoder-decoder architectures and transformer-based models:

1. **Word-Level Encoder-Decoder**: Effective for shorter sequences but limited by LSTM memory constraints.
2. **Transformer-Based mBART**: Achieved higher BERTScores and handled long dependencies effectively.

The comparison presented here makes it clear that mBART outperforms conventional Encoder-Decoder models with an especially improved performance regarding morphologically rich languages such as Telugu. These observations show that pretrained models could serve better for modern translation tasks as they have contextual and semantic understanding. However, much room is left for improvement, specifically regarding long and complex sentences and rare linguistic patterns.
With the incorporation of the recommendations proposed, even better translation quality can be achieved for future iterations, narrowing the gap between human-level and machine-level translation.

References

1. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
2. P. Koehn, "Statistical Machine Translation," Cambridge University Press, 2009.
3. A. Vaswani et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
4. Luong et al. (2015)., "*Effective Approaches to Attention-based Neural Machine Translation*".
5. Bahdanau et al. *"Neural Machine Translation by Jointly Learning to Align and Translate"*
6. Github Dataset Repository. [Online]. Available: https://github.com/SRIDEV93/Language-Translation-Using-Machine-Learning-Telugu-to-English-/blob/main/English.txt
7. mBART Pre-trained Transformer. [Online]. Available https://huggingface.co/facebook/mbart-large-50-many-to-many-mmt