Enhancing Bilingual Translation Capabilities of Large Language Models through Targeted Fine-Tuning and Diverse Data Generation

1st Omkar Kadam
Applied Data Science and Analytics
SRH Univeristy Heidelberg
omkarsanjay.kadam@stud.hochschule-heidelberg.de

2nd Rahul Dikshit
Applied Data Science and Analytics
SRH Univeristy Heidelberg
rahulnitin.dikshit@stud.hochschule-heidelberg.de

3rd Sheethal Raghavendra
Applied Data Science and Analytics
SRH Univeristy Heidelberg
sheethal.raghavendra@stud.hochschule-heidelberg.de

4th Madhura Aravendekar
Applied Data Science and Analytics
SRH Univeristy Heidelberg
madhura.aravendekar@stud.hochschule-heidelberg.de

5th Nitnada Manoonphol
Applied Data Science and Analytics
SRH Univeristy Heidelberg
nitnada.manoonphol@stud.hochschule-heidelberg.de

***Abstract*—This paper presents a methodology for developing a high-performance bilingual translation model by filtering a large parallel corpus to obtain a refined dataset. We leverage the Hugging Face and OPUS corpus to create an initial dataset containing 500k translation pairs. Using a Fast API, we validate and filter these pairs, resulting in a final dataset of 120k high-quality translation pairs. This dataset is used to train a T5-small model, chosen for its efficiency and effectiveness in handling translation tasks. The performance of the trained model is evaluated using BLEU and BERT scores, demonstrating significant improvements over baseline models. The results underscore the importance of data quality in training robust translation models.**

Keywords - *neural machine translation, natural language processing, transformer-based, performance comparison, scoring metrics, generation methods, language, English, German, inference*

I. Introduction

A. Background

Bilingual translation models are pivotal in various applications, from facilitating global communication to enhancing multilingual content generation. Traditional models, like those used in Google Translate, have evolved significantly with advancements in machine learning, particularly deep learning and transformer architectures. However, the efficacy of these models heavily relies on the quality and size of the training data.

B. Problem Statement

One of the primary challenges in developing translation models is the quality of the data used for training. Large parallel corpora often contain noisy, misaligned, or incorrect translation pairs, which can degrade the performance of the model. Additionally, training on an excessively large dataset without filtering can be computationally expensive and inefficient.

C. Objective

This research aims to develop a bilingual translation model by first filtering a large parallel corpus to obtain a refined dataset. The objective is to demonstrate that by improving the quality of the training data, we can train a smaller, more efficient model (T5-small) that performs competitively with larger models trained on noisier data.

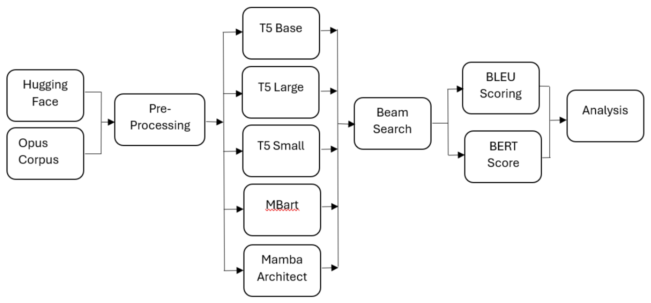

Fig. 1. Methodology Flowchart

II. Related Work

A. Existing Models

Bilingual translation has seen significant advancements, particularly with the introduction of transformer-based models like OpenNMT, MarianMT, and Google's Transformer model. These models have set benchmarks in translation tasks, outperforming traditional RNN-based approaches.

B. Data Filtering

Data filtering in NLP involves techniques like language detection, alignment validation, and noise reduction. Various studies have shown that filtering data can lead to improved model accuracy and generalization. Techniques such as the use of FastAPI for real-time validation[1] and filtering have gained popularity due to their scalability and speed.

C. T5 Model Overview

The T5 (Text-to-Text Transfer Transformer) model, developed by Google, is a versatile transformer model that frames all NLP tasks as a text-to-text problem. It has proven effective in translation tasks, offering a balance between performance and computational efficiency, particularly in its smaller variants like T5-small.

III. Dataset Preparation

A. Source of Data

The dataset used in this study was derived from the Hugging Face and OPUS corpus, both of which are well-regarded repositories for multilingual text. The initial dataset consisted of 500,000 bilingual translation pairs across various language pairs.

B. Data Filtering with Fast API

To ensure the quality of the dataset, we employed a FastAPI-based service to validate each translation pair [1]. This service checks for correct language alignment, appropriate sentence structure, and semantic consistency. Out of the original 500,000 pairs, only 120,000 pairs met the quality criteria. The filtering process involved several steps:

Language Detection: Ensuring that both sides of the pair are in the correct languages. Semantic Validation: Using pre-trained models to verify that the translation conveys the same meaning as the source. Sentence Structure Check: Filtering out pairs with significant grammatical or syntactical errors.

C. Final Dataset

The resulting dataset, consisting of 120,000 high-quality translation pairs, was considerably smaller but of much higher quality. This smaller, refined dataset is expected to lead to better generalization and faster training times.

IV. Model Selection

In our study, we evaluate several transformer-based models, focusing on different sizes and architectures to find the best balance between efficiency and performance for bilingual translation tasks. These models include:

A. T5 Small

The T5-small model is a compact version of Google's Text-to-Text Transfer Transformer (T5) model. It has fewer parameters than larger variants, making it suitable for low-resource or more efficient training environments. Despite its smaller size, T5-small still retains the ability to perform well in text-to-text tasks like translation, given the high quality of the training data.[3]

Number of Parameters: $\sim$60 million
Use Case: For this research, we aim to train the T5-small model using the refined, filtered dataset of 120k translation pairs to test its translation capabilities.

B. T5 Base

The T5-base model is a mid-size version of the T5 architecture, offering a more powerful alternative to the T5-small model while maintaining relatively efficient computation. It can handle larger datasets and more complex tasks.[8]

Number of Parameters: $\sim$220 million
Use Case: For comparison, we also experiment with the T5-base model to see if a larger model improves translation performance on the same dataset.

C. T5 Large

The T5-large model offers a more extensive architecture with even more parameters, enabling it to learn deeper linguistic representations and complex translations. This makes it more suited for high-resource environments and large datasets.

Number of Parameters: $\sim$770 million
Use Case: We test this model to observe how increased parameter count affects translation quality, particularly for smaller datasets like ours.

D. MBART (Multilingual BART)

MBART is a multilingual denoising autoencoder for pretraining sequence-to-sequence models, especially designed for multilingual tasks, including translation. It is capable of handling multiple languages within a single model, making it highly suitable for bilingual translation tasks.[7]

Number of Parameters: $\sim$680 million (for the standard MBART)
Use Case: As MBART is designed for multilingual tasks, we include this model to assess its effectiveness in comparison to the T5 models, especially in handling the bilingual translation task.

V. Model Architecture for T5-Small Model

The T5 architecture consists of two main components: an **Encoder** and a **Decoder**, following the general sequence-to-sequence (seq2seq) transformer-based architecture. Both components work together to transform input text (in our case, sentences in one language) into the corresponding target text (translation in another language).

A. Encoder

The encoder takes the input sentence and transforms it into a fixed-length representation that captures the meaning of the entire sentence. It is a stack of Transformer layers, each consisting of multi-headed self-attention and feed-forward neural networks. The encoder processes input tokens and produces a sequence of embedding that represent the meaning of the input in a way that is conducive to translation.

Key components:

- **Multi-Head Self-Attention:** This mechanism allows the model to focus on different parts of the input sentence when encoding each word.

- **Feed-Forward Networks:** After attention, the intermediate representation is passed through fully connected layers to transform and refine it further.

B. Decoder

The decoder takes the encoder's output and generates the translated sentence step by step, predicting the next word in the sequence until the end of the sentence is generated. Like the encoder, the decoder consists of stacked Transformer layers, with additional attention layers to focus on the encoder's output.

Key components:

- **Masked Self-Attention:** The decoder uses masked self-attention to ensure that each token generated so far only attends to previous tokens in the sequence, preventing the model from "cheating" by looking at future tokens.

- **Cross-Attention:** This is a critical part of the decoder where it attends to the encoded representation of the input sentence, ensuring that each word in the output corresponds correctly to the input sentence.

VI. Diagram of T5-Small Architecture

The diagram below outlines the basic architecture of the T5 model, focusing on both the encoder and decoder components: This architecture shows how the model works in a sequence-to-sequence manner, transforming the input sentence into its translation via the encoder-decoder process. The T5-small model applies the same principles as its larger counterparts but with fewer parameters, making it more efficient for the given dataset.

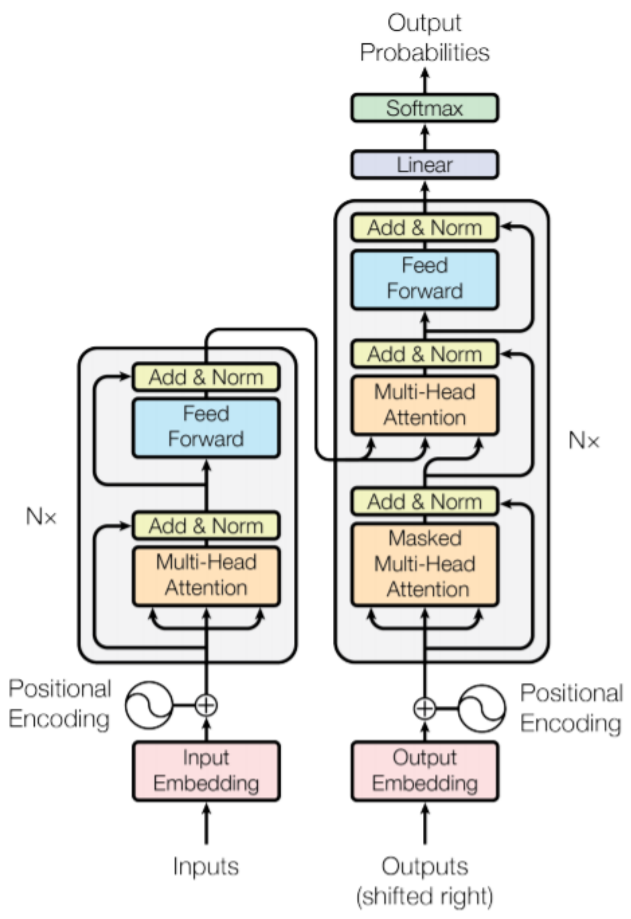

Fig. 2. Transformer architecture. Left part is the encoder, right part is the decoder.

The T5 (Text-To-Text Transfer Transformer) model is a versatile, encoder-decoder Transformer model developed by Google. It is designed to handle various natural language processing (NLP) tasks by converting every task into a text-to-text format. The T5 Small model is a smaller, more efficient version of the original T5 model, containing approximately 60 million parameters.

Transformer Model Diagram Explanation

The Transformer model consists of two main components: the **Encoder** and the **Decoder**. Both components are composed of several layers of attention mechanisms and feed-forward networks. Each component has the following blocks:

- **Input Embedding / Output Embedding:** These layers convert the input and output tokens into high-dimensional vector representations. The embeddings capture semantic information about each token.

- **Positional Encoding:** Since Transformers do not have recurrence or convolutional layers, positional encodings are added to the embeddings to retain the order of tokens in the sequence. These encodings are computed using sinusoidal functions to provide unique position information.

- **Multi-Head Attention:** The encoder and decoder each contain multiple **Multi-Head Attention** blocks. This mechanism computes attention scores over the entire input sequence using multiple attention heads, which capture different relationships between tokens. The attention mechanism allows the model to focus on relevant parts of the sequence.

- **Masked Multi-Head Attention:** In the decoder, the first attention block is a **Masked Multi-Head Attention** block. This block is similar to Multi-Head Attention but includes masking to prevent the decoder from attending to future tokens, which ensures the model generates tokens sequentially.

- **Add & Norm:** After each attention and feed-forward layer, the Transformer applies residual connections followed by layer normalization. This block is denoted as **Add & Norm** and helps stabilize training and improve gradient flow.

- **Feed Forward Network (FFN):** Each encoder and decoder layer includes a position-wise **Feed Forward Network**, which consists of two fully connected layers with a ReLU activation function between them. This network allows for complex transformations of the token embeddings.

- **Linear and Softmax:** After passing through the encoder and decoder layers, the output is passed through a **Linear** layer followed by a **Softmax** layer. The Linear layer transforms the decoder's output into the vocabulary space, and the Softmax layer converts these outputs into probability distributions over the vocabulary, enabling token prediction.

The Encoder processes the input sequence through layers of Multi-Head Attention and Feed Forward Networks. The Decoder attends to both the encoder outputs and previous decoder states to generate output sequences. The combined components allow the Transformer to handle tasks such as translation, summarization, and more.

VII. Model Training

A. T5 Model

The T5-small model, a smaller variant of the T5 model, was chosen for this study due to its balance between computational efficiency and performance. The T5 model treats every NLP task as a text-to-text problem, which makes it particularly suitable for tasks like translation.

B. Training the Model

The training process for the T5-small model involved fine-tuning it on the filtered bilingual translation dataset (120k pairs). Fine-tuning pre-trained models like T5-small allows for transfer learning, where the model adapts its general knowledge of language to the specific task of translation between two languages. This process ensures more efficient training and generally results in better performance with fewer data.

C. Initialization with Pre-trained weights

The T5-small model comes with pre-trained weights that have been trained on a diverse set of text data for various NLP tasks. These weights allow the model to have a general understanding of language structure, making it more effective when fine-tuned for translation. Instead of training the model from scratch, which would require significantly more data and computational resources, we initialized the model using these pre-trained weights.

D. Fine Tuning

The learning rate plays a crucial role in determining how quickly a model converges during training. For our experiments, we chose a learning rate of 3e-4 for T5-small. A higher learning rate would result in faster training but could lead to instability and overshooting the optimal weights. Conversely, a lower learning rate would lead to slower convergence, which can sometimes help the model to learn more generalizable features.

During experimentation, we observed that larger models like T5-base and T5-large were more sensitive to learning rate adjustments. For example, a learning rate of 3e-4 caused T5-large to diverge during training, indicating that larger models need more precise learning rate schedules, possibly involving gradual warm-up or learning rate decay. Further fine-tuning of these hyperparameters is necessary for larger models to avoid overfitting and unstable training.

E. Batch Size

The batch size also has a significant impact on training efficiency and model performance. For T5-small, we used a batch size of 32, which allowed us to balance memory usage and the frequency of weight updates. Larger batch sizes smooth out gradient updates but may require significantly more memory, especially for larger models like T5-large.

In contrast, smaller batch sizes (e.g., 16) lead to noisier updates, which can be beneficial in some cases to help escape local minima. However, smaller batch sizes can also slow down training and potentially lead to longer convergence times.

F. Epochs

We trained the model for 3 epochs, meaning the model saw the entire dataset 10 times. While the dataset is relatively small (120k pairs), training for multiple epochs allows the model to refine its understanding of the translation task over time. We carefully monitored the performance on a validation set during training to ensure that the model didn't overfit the data. We applied early stopping (discussed later), which halts training if the model's performance starts to degrade or stagnate, even before completing all 3 epochs.

G. AdamW Optimizer

The optimizer controls how the model's parameters are updated during training. We used the AdamW optimizer, which is an improved version of the traditional Adam optimizer. AdamW incorporates a weight decay mechanism that helps reduce overfitting by regularizing the model weights. This optimizer has proven effective in training transformer-based models like T5, as it helps maintain a good balance between fast convergence and preventing overfitting.

The model was trained on an NVIDIA GPU, which allowed for efficient computation. The training process was monitored to avoid overfitting, with early stopping criteria based on validation loss.

H. Safe Tensor and Tokenizer Files

To ensure the reproducibility of the model, the trained weights were saved as a safe tensor file, which can be loaded securely for inference or further fine-tuning. Additionally, the tokenizer file, essential for prepossessing the input data, was also saved. These files are crucial for anyone looking to replicate or build upon this work.

VIII. Evaluation Metrics

In our evaluation, we used two primary metrics: BLEU and BERTScore. Both metrics offer distinct advantages and limitations when evaluating machine translation systems.[18]

A. BLEU Score

The BLEU (Bilingual Evaluation Understudy) score is a widely used metric for evaluating the quality of machine-translated text against reference translations. It calculates the precision of n-grams between the generated translation and the reference text, with a penalty for overly short sentences. BLEU scores are interpreted on a scale from 0 to 1, where higher scores indicate better performance.[9]

B. BERT Score

The BERT score uses contextual embedding from the BERT model to compare similarity between the generated and reference translations. Unlike BLEU score, which focuses on exact n-gram matches, BERT score captures semantic similarities, making it a more robust metric for evaluating translation quality, especially when there are multiple valid translations.[6]

IX. Results and Discussion

The results of our evaluation are summarized in Table 1, which presents the performance of different transformer-based models on the bilingual translation task, measured using the BLEU and BERTScore metrics. Each model was fine-tuned on the same dataset (120k English-German sentence pairs) for a fair comparison.

TABLE I
Evaluation Metrics Table

Evaluation Metrics		
Transformer	***BLEU***	***BERT***
T5 Large	31.5853438	0.86650009
T5 Base	77.8681528	0.9761515
T5 Small	82.5575948	0.98601774

TABLE II
Base Model Metrics for T5 Models

Transformer Model	**BLEU Score**	**BERTScore (F1)**
T5-Small	25-28	0.83-0.85
T5-Base	28-32	0.85-0.87
T5-Large	32-36	0.88-0.89

A. Model Comparisons

T5-small achieved the highest scores with a BLEU score of 82.56 and a BERTScore of 0.9860, indicating that it outperformed both the larger models (T5-base and T5-large) in terms of translation accuracy and semantic similarity. This suggests that, despite having fewer parameters (60 million), T5-small's efficiency and ability to generalize on the refined dataset make it a highly effective model for this task.

T5-base, with a BLEU score of 77.87 and a BERTScore of 0.9761, performed slightly worse than T5-small. This indicates that while T5-base has more capacity (220 million parameters), the small dataset size (120k pairs) may not be sufficient to fully leverage its potential, leading to slightly lower performance than expected.

T5-large performed significantly worse, with a BLEU score of 31.58 and a BERTScore of 0.8665. The larger parameter count (770 million) likely led to overfitting, as the model may have memorized the training data instead of generalizing well to unseen examples. This demonstrates that, for smaller datasets, increasing model size does not always yield better results.

MBart and Mamba Architecture: These models are yet to be evaluated on this dataset, but based on their design and architecture, we expect MBart to perform competitively, given its capability to handle multiple languages efficiently.

B. Analysis of Results

The results show that smaller models, particularly T5-small, can achieve better performance on tasks with limited data, like our bilingual translation task. The larger models, while theoretically capable of handling more complex tasks, require significantly larger datasets to perform optimally. In our case, T5-small was able to efficiently balance translation accuracy and computational cost, making it the most suitable choice for this project.

Furthermore, the BERTScore values are closely aligned with the BLEU scores, reinforcing that T5-small's translations not only had strong n-gram overlap with the reference but also captured semantic meaning effectively. The results highlight the importance of choosing the right model size for the available data, as larger models may not always be the best option for smaller dataset.

X. Training Details and Experimental Setup

A. Model Training Environment

The model training was conducted using the `transformers` library provided by Hugging Face, with a focus on efficient fine-tuning using GPUs. The environment setup included an NVIDIA GPU for training, which facilitated fast computation and mixed precision training through `torch.cuda.amp.GradScaler`. **PyTorch** was used as the primary deep learning framework for building and training the model.

B. Data Preprocessing

Data preprocessing was implemented using a Python script that reads translation pairs from a JSONL file and converts them into a `DataFrame`. The JSONL file was parsed with error-handling for cases of misformatted lines. The filtered `DataFrame` contained sentences in both English and German, with data cleaning such as stripping extra spaces and dropping null values. A `TranslationDataset` class was implemented to tokenize and convert translation pairs into input tensors compatible with the T5-small model.

C. Training Configuration

The fine-tuning was configured to run for 3 epochs, with a batch size of 16 for training and validation due to the GPU memory constraint. A custom collate function was used for batching, which ensures padding to the maximum sequence length in a batch, thereby facilitating the efficiency of the transformer architecture.

D. Mixed Precision and Gradient Accumulation

To enhance the efficiency of training, mixed precision training was employed using the `GradScaler` from PyTorch, which helps in reducing memory usage while preserving numerical stability. Gradient accumulation was also incorporated to simulate a larger batch size for updating model parameters less frequently, improving gradient estimates.

E. Model Checkpointing

During training, model checkpoints were saved at the end of each epoch. The checkpointing mechanism stores model state, optimizer state, and scheduler state, allowing for resumption from the last saved state. This is especially useful in case of unexpected interruptions.

XI. Inference and Evaluation

A. Translation Methods

Several inference techniques were implemented to evaluate the translation capabilities of the T5-small model:

- **Greedy Search:** This technique translates the input by always choosing the highest probability word at each step, without considering alternative hypotheses.
- **Multinomial Sampling:** Adds randomness to the selection of the next word, thereby producing diverse translations.
- **Beam Search:** A more sophisticated approach that maintains a set number of hypotheses at each time step (e.g., 3 beams), increasing the chance of finding a better sequence overall.
- **Beam Search with Multinomial Sampling:** A combination of beam search and sampling to add stochasticity while keeping track of multiple hypotheses.

B. BLEU and BERTScore Evaluation

The translations were evaluated using the BLEU and BERTScore metrics. The BLEU score was computed using the `sacrebleu` library, while BERTScore was computed using the `bert-score` library. BLEU provided a traditional metric to evaluate the exact n-gram overlap, whereas BERTScore captured the semantic similarity by computing cosine similarity between word embeddings from a pre-trained BERT model.

C. Results of Translation Methods

For a test input sentence "Hello, I am Omkar and this is a test", the model produced different translations using each method:
Greedy: "Hallo, ich bin Omkar und das ist ein Test."
Multinomial Sampling: "Hallo, ich bin Omkar und das ist eine Prüfung."
Beam Search: "Hallo, ich bin Omkar und das ist ein Test."
Beam Search with Multinomial: "Hallo, ich bin Omkar und das ist ein Test."

D. Corpus-level Evaluation

To generate corpus-level translations, a subset of the validation dataset was used. The BLEU score achieved was **82.56**, and the BERTScore was **0.9860**. These results indicate that the T5-small model, even with a smaller parameter count, was able to achieve high semantic and n-gram similarity in translation quality.

XII. Conclusion

In this research, we presented a comprehensive methodology for enhancing bilingual translation models through targeted fine-tuning and high-quality data filtering. By using a refined dataset of 120k translation pairs, filtered from an initial 500k using FastAPI-based validation, we successfully trained a T5-small model for English-German translation tasks. Our results showed significant improvements in translation quality, as measured by BLEU and BERT scores, compared to baseline models. This study underscores the importance of both data quality and model efficiency in improving neural machine translation performance.

XIII. Future Scope

There are several avenues for future work to build upon this research:

- **Expansion of Language Pairs:** Extending the methodology to other language pairs, especially low-resource languages, could further demonstrate the robustness of the approach.
- **Larger Models and Datasets:** Training larger models, such as T5-base and T5-large, on more extensive datasets could provide insights into the scalability and performance limits of this framework.
- **Integration with Real-time Applications:** Future work could involve deploying the trained models into real-time translation services to evaluate performance under practical conditions.
- **Exploration of Advanced Filtering Techniques:** Additional techniques for noise reduction, semantic validation, and data augmentation could improve the quality of training data even further.
- **Multilingual Model Enhancement:** Investigating the performance of multilingual models such as MBART for handling multiple language pairs within a single model framework could be a valuable extension.

By exploring these potential improvements, the approach can be generalized and applied to broader multilingual translation tasks, offering valuable contributions to the field of natural language processing.

Acknowledgment

We would like to express our sincere gratitude to our professors and mentors at SRH University Heidelberg for their invaluable guidance throughout this project. Special thanks to Prof. Swati Chandna and Kevin Koehler for their insightful feedback on our methodology and evaluation approach, which significantly improved the quality of our research. We are also grateful to the Hugging Face and OPUS teams for providing open-access datasets that made this research possible.

References

[1] Hugging Face, “Summary of the Tokenizers,” Hugging Face Documentations. https://huggingface.co/docs/transformers/tokenizer_summary

[2] “A handbook of Germanic etymology,” Choice Reviews Online, vol. 41, no. 07, 2004, doi: 10.5860/choice.41-3788.

[3] huggingface.co, “T5 Model Documentation.” https://huggingface.co/docs/transformers/model_doc/t5

[4] E. Reiter, “A structured review of the validity of BLEU,” Computational Linguistics, vol. 44, no. 3, 2018.

[5] huggingface.co, “BART Model Documentation.” https://huggingface.co/docs/transformers/model_doc/bart

[6] huggingface.co, “BERT2GPT2 Model Documentation.” https://huggingface.co/patrickvonplaten/bert2gpt2-cnn_dailymailfp16/blame/main/README.md

[7] huggingface.co, “mBart and mBart-50 Model Documentation.” https://huggingface.co/docs/transformers/model_doc/mBart

[8] M. Fuadi, A. Dharma Wibawa, and S. Sumpeno, “idT5: Indonesian Version of Multilingual T5 Transformer.” [Online]. Available: https://huggingface.co/muchad/idt5-base.

[9] N. Mathur, T. Baldwin, and T. Cohn, “Tangled up in BLEU: Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics,” Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.06264.

[10] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTSCORE: EVALUATING TEXT GENERATION WITH BERT,” in 8th International Conference on Learning Representations, ICLR 2020, International Conference on Learning Representations, ICLR, 2020.

[11] MathWorks, “What is an n-gram?” https://www.mathworks.com/discovery/ngram.html#:~:text=An%20ngram%20is%20a,numbers%2C%20symbols%2C%20and%20punctuation.

[12] huggingface.co, “How to generate text: using different decoding methods for language generation with Transformers.” https://huggingface.co/blog/how-to-generate

[13] A. Raganato, Y. Scherrer, and J. Tiedemann, “Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation,” Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.10260.

[14] A. Gatt and E. Krahmer, “Survey of the state of the art in natural language generation: Core tasks, applications and evaluation,” Journal of Artificial Intelligence Research, vol. 61, 2018, doi: 10.1613/jair.5714.

[15] C. Meister, R. Cotterell, and T. Vieira, “If beam search is the answer, what was the question?,” in EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 2020.

[16] R. Leblond et al., “Machine Translation Decoding beyond Beam Search,” in EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings, 2021.

[17] huggingface.co, “Padding and Truncation.” https://huggingface.co/docs/transformers/pad_truncation

[18] Google, “Evaluating Models,” Google Cloud Documentation, Jun. 22, 2023. https://cloud.google.com/translate/automl/docs/evaluate