Translating Japanese into English with the Transformer Model Architecture

Zhertovskaia Elizaveta Vladimirovna

Introduction

Abstract

This paper is devoted to the analysis of a transformer's model architecture quality to translate the text from the Japanese to the English language. This study project's main goal is to thoroughly examine the transformer model components, presented in "Attention is All You Need" (Vaswani et al., 2017) paper and empirically assess the capacity of a classic transformer model to translate from an Asian to a European language. By building the transformer model and training it on a small amount of Japanese-English data from the Tatoeba Project's database, the paper hypothesizes that a transformer model is a decent and sophisticated choice for a translation task.
The model's architecture, as it was mentioned above, was built following the classic "Attention is All You Need" research paper from Google, where the translation task was implemented only for European languages.
Then, the built model was pre-trained on bilingual data.
The project measures effectiveness by applying such metrics as average of a loss function, perplexity, and BLEU.
Methologically, the project includes the training on a small Japanese-English Anki dataset containing approximately 115 thousand Japanese-English sentence-pairs, generating the translation results of a training procedure, and their measurement via some key NLP metrics.
Results demonstrate a great performance of a transformer model in translation tasks, yet reveals that, at the current stage of NLP, the architecture is difficult to build and train due to its instability and high computational costs.

Key Concepts

Machine Translation - is a use of ML and AI, which main goal is to translate the text from one language to another taking idiomatic, lexical, contextual and other complex language aspects into consideration.

Transformer - a neural network architecture that is based on encoder and decoder and relies on self-attention mechanism.

Accuracy - a metric that measures how well the model predicts the correct outputs.

Perplexity - a metric that measures the uncertainty of a model with predicting an accurate output.

BLEU - a metric that evaluates a machine translation by comparing the quantity of n-grams in a generated text to the quantity in the reference example.

“Attention is All You Need” - a 2017 revolutionary paper from Google scholars first published in arxiv.

Pre-training - a process during which the model is trying to learn the data patterns from scratch and then use them to generate unique outputs.

Dataset - a set of data used as an input for a model training, pre-training, or fine-tuning.

Encoder - a component of a transformer architecture processes input data and extracts meaningful features for further learning and translation.

Decoder - another crucial component of a transformer architecture that generates output based on the encoded features.

Attention Mechanism - a transformer-model component that compares the input tokens to one another during processing.

Layer Normalization - a stabilization technique used in deep learning to prevent the model from an unstable training process by normalizing all the features independently, rather than a mini-batch.

Data

The open-source Tab-delimited Bilingual Sentence Pairs dataset from the Tatoeba Project was used to train the model. It is a bilingual Japanese-English dataset consisting of over 115 thousand samples, 68 thousand of which were used during the model training. The database sentences are really diverse: they contain many grammar constructions, and range from simple to more complex examples. The data vocabulary is rather simple and general, yet, because of that, it does not have any noisy and useless syntaxis or english words, which makes data preprocessing and pretraining much easier.

None of the data was cleaned or processed, in order to keep original punctuation. The samples, as it was said before, do not contain any complex syntaxis or punctuation that would require additional cleaning.

The Project Structure

Research and Preparation
Purpose: Investigate a transformer model architecture, analyze its components, and explore the specific aspects of processing asian languages and the process of a model pretraining on the data in those languages. Find the available datasets.
Challenges: Lack of the datasets without noisy text, like links, tags, emojis, meaningless, language-wise, symbols, numbers, and words written in English.

Data Preprocessing
Purpose: Preprocess the chosen dataset in such a way that the model could grasp the language patterns on different levels and stably learn how to generate a translation using the dataset sentence pairs.
Challenges: Specific preprocessing for asian characters, hiragana, and katakana. Choosing a bpe-tokenizer specifically for the Japanese language.

Model Building and Training
Purpose: Learn the peculiarities of the transformer and its components, understand the mathematics behind it, implement the basic model example from the paper, and, then, adapt it to the Japanese-English translation task. Choose the training parameters and train the model on the found data.
Challenges: Difficulties with understanding the mathematical concepts, troubles with implementing the components of a model from the paper. Adapting the architecture and parameters to an asian language.

Evaluation
Purpose: Assess the models training results by measuring the outputs quality via some foundational NLP metrics. Those include: Accuracy, Perplexity, and BLEU. The last one is especially paramount for this purpose, for it is often used in other research works and, therefore, convenient for a comparison between the results of this project and other Japanese-English translation task implementations.
Challenges: Ensuring fair metric representation of the generated results.

Machine Translation Task

Machine translation is a sub-field of computational linguistics that focuses on developing systems capable of automatically translating text or speech from one language to another. In Natural Language Processing (NLP), the goal of machine translation is to

produce translations that are not only grammatically correct but also convey the meaning of the original content accurately.
A neural network, inspired by the human brain, is a network of interconnected nodes functioning as an information system. Input data passes through these nodes to produce an output. Neural machine translation software utilizes neural networks to process vast datasets, with each node contributing a specific change from source text to target text until the final result is obtained at the output node.

Input Data

A batch of tokenized sequences $B = \{S_1, S_2, S_3 \ldots. S_n\}$, where B is a batch size and each sequence S_i is a vector of discrete token indices of length L: $S_i = [s_{i1}, s_{i2}, s_{i3} \ldots.. s_{iL}]$, where $s_{ij} \in \{1, 2, 3 \ldots.. V\}$. Here, V denotes the fixed vocabulary size and L is the pre-defined context window length. This integer-indexed batch is converted into its continuous representation via a lookup operation on a trainable embedding matrix $E \in R^{vxd}$, where d is the model's embedding dimension. The resulting input tensor to the first transformer layer is $H^{(0)} \in R^{BxLxd}$

Output Data

A conditional probability distribution over the whole vocabulary V for each target position in the sequence. , autoregressively conditioned on all preceding tokens. Formally, for a given input sequence $X_{1:L}$, the model parameterizes the distribution:

$P(x_t \mid X_{1:t-1}; \Theta) = \text{softmax}(W * h_t^{(L)} + b)$, where:

- Θ denotes the full set of model parameters.
- $h_t^{(L)} \in R^d$ is the final contextualized representation of the token at position t after processing through L transformer layers.
- $W \in R^{dx|V|}$ and $b \in R^{|V|}$ are the output projection matrix and bias, respectively.

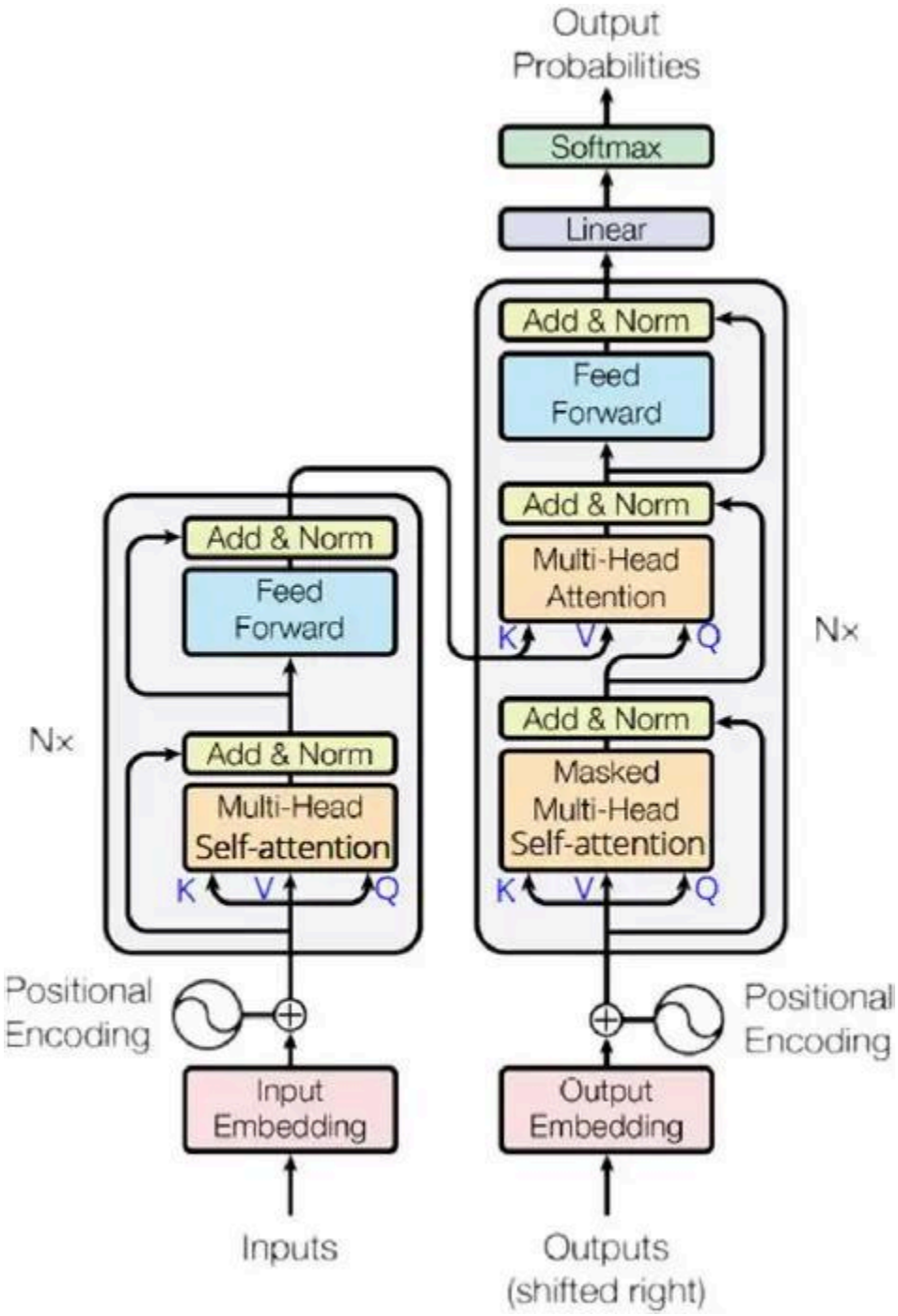
The model's fundamental output is thus a sequence of probability vectors $[P_1, P_2, \ldots\ldots, P_L]$, from which text generation proceeds via ancestral sampling, beam search, or other decoding strategies.

Transformer architecture

The transformer architecture includes:

1. Tokenizers
2. Embedding layer

3. Positional encoder
4. Encoder block
5. Decoder block
6. Self-Attention mechanism



Tokenizer

The transformer neural network operates over numbers, not the text, and because of that it is necessary to map the text dataset to a numerical representation. So, it was crucial to create the text tokens. This was done by training a BPE tokenizer on the dataset vocabulary in both English and Japanese, since there was a problem finding a proper BPE, not bert base-cased, japanese tokenizer. Apart from that, training a custom tokenizer helped ensure that the vocabulary tokens will correspond to the used data. The tokenizers were trained with SentencePiece.

Embedding Layer

Secondly, in order to get a proper form of input the vector embeddings, representing the japanese and english texts were created. This procedure was done to get a semantic representation of the words from the dataset.

Positional Encoding

Just right after that, since the transformer does not have any mechanism to capture the order of tokens, positional encoding was added. Positional encodings are vectors which encode the position of each token in a sequence. These encodings are then added to the token embeddings, enabling the model to consider both the content and position of tokens. For our case, the embedding dimension is 512 and the maximum sequence length is 80, so the positional encoding matrix has a shape of (80, 512).

Encoder and Decoder Blocks

The encoder is composed of a stack of identical encoding layers. These layers process the entire input sequence simultaneously layer-by-layer. Conversely, the decoder is composed of a stack of identical decoding layers. These layers operate iteratively, processing both the final contextual representations produced by the encoder and the sequence of tokens generated by the decoder itself up to the current step.

The objective of each encoder layer is to generate representations with a context for each token. This is achieved by enabling each token's representation to integrate

information from all other tokens in the input sequence through the mechanism of self-attention.

Self-Attention

Another significant part of the transformer is the self attention mechanism, which was mentioned in a previous paragraph. Implemented in an encoder block, self attention allows the model to look at other positions in the input sequence for additional information. It creates a vector based on dependency of each word with the other by using query vector, key vector, and value vector.

Feed-Forward Network

Each position in the sequence is independently processed using a feed-forward network located in both encoder and decoder - a two-layer fully connected network applied position-wise. This procedure helps the transformer learn complex representations of input features.

Layer Normalization

This type of layers normalizes activations to stabilize training dynamics. It is especially critical for small datasets where parameter updates can be volatile.

Dropout

Randomly deactivates neurons (the rate was set to 10%) to prevent over-reliance on specific features, mitigating overfitting.

Parameters

Model Dimension (d_model): The dimensionality of the words vectors within the model's vector space.
Number of Heads (n_heads): Number of heads used in the multi-head attention mechanism. Each head learns to focus on different parts of the input sequence, capturing different relationships.
Number of Layers (n_layers): Number of stacked encoder or decoder layers in the transformer model.

Maximum Sequence Length (max_seq_len): the maximal number of tokens that can be processed. Longer texts are truncated to the maximum sequence length, while the shorter ones are extended to the number of this parameter.
Dropout Probability (dropout): Set to 0.1, balancing regularization without underfitting. Adjusted empirically based on validation performance.

Training Process

The training pipeline was primarily based on the paper incorporating some changes to address the challenges of training a Japanese-language model, balancing computational efficiency with stability. Below is a detailed explanation of the training cycle, including its components and their role in optimizing the model for Japanese linguistic features.

The training process begins with hardware and parameter configuration. The model is assigned to a CUDA-capable GPU if it is available.

Key hyperparameters include:
Epochs: Set to 100 cycles through the dataset, sufficient for training without overfitting on limited Japanese-English data.
Learning Rate: Initialized at 1e-4, one of the standard values for training the transformer or the transformer-based model.
Batch Size: Determined by the `train_loader`, typically 64-128 samples to fit GPU memory constraints. Gradient accumulation (not explicitly shown but implied by loss normalization) simulates larger batches and improves the stability during the process of gradient estimation.

The Adam optimizer was chosen as a basic optimizer from the paper. Its parameters include momentum terms (betas=(0.9,0.98)) and a small epsilon value (1e-9) to stabilize gradient updates.

Checkpointing and Training Resumption

A checkpointing system saves model states after each epoch to the `./checkpoints` directory. Each checkpoint includes:
- Model weights (`model_state_dict`)
- Optimizer state
- Current epoch and loss value

If interrupted, training can resume from the last saved checkpoint by calling a function for loading a checkpoint, preserving the optimizer's momentum and learning rate schedule. This is particularly valuable for long training runs on limited hardware.

Training Loop Execution

The training loop iterates over the dataset for the specified number of epochs. For each batch:

Data Preparation: The training process begins with putting inputs ("ja_ids", "en_ids") to the target device (GPU).

Forward Pass: The model processes the tokenized text pairs, generating logits and loss. The loss is normalized by gradient resetting.

Backward Pass: After the gradients are computed, the optimizer updates model weights.

Progress Tracking: A progress bar (`tqdm`) displays real-time updates, including the current epoch and loss.

After each epoch, the average loss is logged, and after every 10 epochs a checkpoint is saved. This ensures recoverability and provides snapshots for post-hoc analysis.

This structured training process ensures the model can adapt to Japanese linguistic nuances while maintaining computational efficiency.

Metrics and the translation quality

The trained Transformer was evaluated on 200 Japanese-English text pairs and achieved a training loss of 0.1459, validation loss of 2.8688, perplexity of 1.1571, and a BLEU score of 0.6270.

These results reflect a nuanced and instructive performance profile. The BLEU score of 0.6270—particularly high given the model's limited scale and dataset—confirms the Transformer's inherent capacity to capture syntactic reordering and lexical mapping between Japanese and English through its self-attention mechanism. The remarkably low perplexity of 1.1571 indicates strong predictive accuracy on the training distribution,

demonstrating mastery over the specific linguistic patterns present in the training corpus.

However, the significant gap between the low training loss (0.1459) and higher validation loss (2.8688) highlights a deliberate and expected limitation: the model has achieved deep pattern recognition at the expense of broad generalization, revealing clear signs of overfitting. This outcome aligns directly with the original 2017 characterization of the Transformer as a data-hungry architecture now applied to a constrained dataset.

Common Errors

After an evaluation,the translation outputs were checked. This helped find some common mistakes:

1. A complete distortion of the original sentences.
2. Using incorrect words or name entities
3. Dropping words, losing the specific meaning of the Japanese samples after translation.
4. Failing to use capital letters where they are necessary.
5. Not paying attention to emotional emphasis.

These are some samples of the translation outputs from the validation dataset which failed to represent the original samples well enough:

A)「今夜電話してもいい？」「いいよ」
Reference: May I call you tonight? "Sure."
Model: may i call you tonight? "no, please.".
Interpretation: A drastic change of a communicative situation. The model translated the agreement「いいよ」("Sure.") as a rejection, which completely changes a situation in the dialogue.

B) 夜通し勉強した。
Reference: I was up all night studying.
Model: i worked all night.
Interpretation: Unable to translate a verb accurately. As a result the sentence lost its specific. The original 勉強した (“was studying”) was translated as “worked”, which is 働いた in Japanese.

C) 簡単に手に入れたものはすぐに失いやすい。
Reference: That which is easily acquired is easily lost.

Model: easy come soon.
Interpretation: The model completely failed to translate an original sentence. It only detected the core theme of the sample - something easy, yet did not manage to depict a sense of the sentence.

D) フランス語は便利よ。
Reference: French is useful
Model: i've heard french.
Interpretation: Use of another entity (a verb instead of an adjective) with an inappropriate meaning. Hearing French doesn't make sense if we look at the reference, which tells how useful the French language is. Moreover, the emotional aspect of a よ letter was lost.

E) 痛い目に遭わせるぞ
Reference: Don't make me hurt you.
Model: my eyes are sushi.
Interpretation: a phrase meaning "I'll make you suffer" or literally "I'll make you experience pain." The translation "my eyes are sushi" is complete nonsense.

Despite the major failures, the model still managed to translate some sentence correctly:

A) 宿題は自分でやったの？
Reference: Did you do your homework by yourself?
Model: did you do your homework by yourself?
Interpretation: A message of the sentence is saved. The Japanese rhetorical question "宿題は自分でやったの？" (literally: "Homework by yourself did?") is rendered well as the natural English "Did you do your homework by yourself?

B) 10ヶ月ぶりで彼は帰国した。
Reference: After an absence of ten months, he returned home.
Model: he returned back home after being away for ten months.
Interpretation: The Japanese "10ヶ月ぶりで" meaning "after ten months of absence" is accurately translated as "after being away for ten months." The model's answer sounds natural and does not misinterpret an idea behind the sentence.

In practice, this performance profile translates to a tool that excels on familiar, structurally straightforward sentences—accurately handling basic declarative statements and simple questions—while struggling with idiomatic expressions, nuanced dialogue, and cultural references. The model thus serves not as a production-grade

system, but as a highly effective specialized instrument: it offers a clear, empirical benchmark for studying Transformer learning mechanics and precisely quantifies the relationship between architectural capacity, data volume, and generalization in neural translation from Japanese to English.

Conclusion and Future Work

The performance of the final model—achieving a BLEU score of 0.6270 and a validation loss of 2.8688—demonstrates the architecture's fundamental capacity to learn meaningful translation mappings even from a constrained dataset. These results confirm the effectiveness of the self-attention mechanism in processing Japanese syntactic reordering while also exposing the expected limitation of overfitting, as evidenced by the marked divergence between training and validation loss. This outcome fulfills the project's primary objective: providing a practical, in-depth investigation into the Transformer's learning mechanics for the Japanese-English translation task.

Future work will systematically expand both model capacity and dataset size to mitigate overfitting and enhance generalization. A comprehensive error analysis will also be undertaken to categorize recurring translation failures—such as those involving idiomatic expressions, contextual particles, and compound nouns—to inform targeted architectural refinements.

The Literature Studied:

Ahmed, M., Ouda, A., Abusharkh, M., Kohli, S., & Rai, K. (2023). An optimized approach to
translate technical patents from english to japanese using machine translation models. Applied
Sciences, 13(12), 7126.

Cruz, J. C. B., & Cheng, C. (2019). Evaluating language model finetuning techniques for low resource languages. arXiv preprint arXiv:1907.00409.

Kentaro Kurihara, Daisuke Kawahara, and Tomohide Shibata. 2022. JGLUE: Japanese General Language Understanding Evaluation. In Proceedings of the Thirteenth Language Resources and Evaluation Conference, pages 2957–2966, Marseille, France. European Language Resources Association.

Lee, S., Lee, J., Moon, H., Park, C., Seo, J., Eo, S., ... & Lim, H. (2023). A survey on evaluation
metrics for machine translation. Mathematics, 11(4), 1006.

McGiff, J., & Nikolov, N.S. (2025). Overcoming Data Scarcity in Generative Language Modelling for Low-Resource Languages: A Systematic Review. ArXiv, abs/2505.04531.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). BLEU: a method for automatic
evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association
for Computational Linguistics (pp. 311-318).

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Vaswani, Ashish et al. “Attention is All you Need.” Neural Information Processing Systems (2017).

Dataset:

Original Tatoeba dataset: https://www.manythings.org/anki/

Github Link: https://github.com/applepi17618/ja_en_transformer