Exploring the Impact of Tokenization Strategies on Machine Translation using Transformer Seq2Seq Architecture - Experiment

Filippo Merlo[1]
[1] CIMEC, University of Trento
filippo.merlo@studenti.unitn.it

April 30, 2025

1 Intoduction

Transformers have significantly improved Neural Machine Translation (NMT) by removing the constraints of sequential processing, allowing for more efficient training and scalability to larger datasets [1] [5]. However, the performance of these models is highly sensitive to the tokenization strategy used, which breaks down text into tokens that the model can process. Tokenization directly impacts the model's input representation, influencing its ability to capture linguistic nuances and, consequently, its translation quality.

This project investigates three tokenization methods: character-level, word-level, and subword-level (using WordPiece tokenization). Character-level tokenization treats each character as a separate token, which can capture detailed information and handle words not in the vocabulary but often results in longer sequences, complicating model training. Word-level tokenization treats each word as an individual token, simplifying implementation but struggling with out-of-vocabulary (OOV) words and potentially leading to an excessively large and inefficient vocabulary.

WordPiece tokenization [6], a subword-level approach used in models like BERT [2], addresses these issues by breaking down words into smaller, more manageable subwords or characters. This method reduces the vocabulary size while effectively handling OOV words, enhancing the model's ability to process real-world text data. The process starts with a small vocabulary and iteratively merges subwords based on a calculated score that prioritizes less frequent combinations, optimizing the model's performance.

The project hypothesizes that different tokenization strategies will significantly impact the translation quality and overall performance of Transformer-based Seq2Seq models, measured using the Bilingual evaluation understudy (BLEU) score [3]. with subword-level tokenization expected to offer a balanced approach, optimizing vocabulary size and translation results.

2 Data and model

To test our hypothesis, I utilized the CCMatrix dataset [4], which comprises 32.7 billion unique sentences across 38 languages, mined using margin-based bitext mining with the LASER toolkit.

Specifically, I selected 203,000 English-Italian sentence pairs, dividing them into 200,000 pairs for training and 3,000 pairs for testing translation quality.

The model I used is a standard seq2seq transformer architecture. This setup comprises two neural networks: an encoder transformer for processing the tokens of the initial English sentence, and a decoder transformer for generating tokens of the Italian translated sentence. Both the encoder and decoder have 1 layer each, 8 attention heads per layer, an input size of $s = 200$ (the size of the embedding vector for each token in the sequence), and a hidden size of $d = 512$. ReLU activation functions are used in the linear layers. Sinusoidal positional encodings are added to the input embeddings.

3 Experimental setup

The experiment is organized into three main sections. First, I will describe the three different methods used for preprocessing the sentences. Next, I will provide details about the training process. Finally, I will explain the function used to evaluate the quality of the model's translations.

3.1 Preprocessing

For each of the three tokenization methods, both the representation of sentences before being fed into the model and the composition of the model's vocabulary are altered. Before tokenization, sentences are filtered based on their length; any sentence longer than 200 characters is excluded from both the training and test sets. This is because the model's input length is capped at 200, and during character-level tokenization, it's crucial to ensure sentences do not exceed this limit.

The first tokenization strategy I implemented is character-level tokenization. In this approach, the vocabulary is predefined and consists of all the letters of the alphabet, numbers, punctuation marks, and three special tokens: the padding token '<PADDING>', the start-of-sequence token '<START>', and the end-of-sequence token '<END>'. The entire vocabulary contains 87 tokens, as detailed in the appendix. Only with this approach, sentences undergo an additional filtering step to remove any that contain OOV characters. With this strategy, each sentence is converted character by character into the corresponding index based on the vocabulary. After tokenization, the start, padding, and end-of-sequence tokens are added to the sequence.

The second tokenization strategy I employed is word-level tokenization. In this method, each word in a sentence is treated as an individual token. Initially, a vocabulary is constructed by compiling all unique words and punctuation marks from the dataset plus the same special characters of the previous approach. This process results in the creation of two extensive vocabularies, one for each language. There are respectively 59885 tokens for the English one and 81825 tokens for the Italian one.

The third tokenization strategy is WordPiece tokenization. This algorithm is designed to handle out-of-vocabulary (OOV) words by breaking them down into smaller, known subwords or characters, enabling the model to process words not encountered during training. By decomposing words into more fundamental and shared components, WordPiece reduces the overall vocabulary size while maintaining linguistic diversity.

The process begins with a small vocabulary, including special tokens and an initial set of characters. Each word is initially split into subwords using a prefix (e.g., "##"), so the word "word" becomes "w ##o ##r ##d." The model then learns to merge subwords based on a calculated score, which prioritizes less frequent combinations over more frequent ones. The score

for a subword pair is determined by dividing the frequency of the pair by the product of the frequencies of each of its parts:

$$\text{score} = \frac{\text{freq\_of\_pair}}{\text{freq\_of\_first\_element} \times \text{freq\_of\_second\_element}}$$

This calculation allows the algorithm to focus on merging pairs where the individual parts are less common in the vocabulary. The process continues iteratively until the vocabulary reaches the desired size. To build this tokenizer I followed the implementation provided by the hugging face tutorial "Building a tokenizer, block by block" available at `https://huggingface.co/learn/nlp-course/en/chapter6/8?fw=pt`. I set the vocabulary size to 10,000 tokens for both English and Italian.

3.2 Training

The training has been the same for all three conditions. The transformer is trained using the Adam optimizer to minimize the cross-entropy loss averaged over tokens, with a batch size of 30 data points (english-italian sentence pairs). The training lasts for 10 epochs, starting with a learning rate of 0.0001. A dropout rate of 0.1 is applied to the input embeddings and transformers. No hyperparameter tuning is applied since the aim of the study is to compare the change in performance among different tokenization strategies. All the parameters are kept constant in the three different trainings to allow a more fair comparison.

3.3 Evaluation

The quality of the translation is evaluated with a classical measure for machine translation, the Bilingual evaluation understudy (BLEU) score [3]. The BLEU score consists of a number between zero and one that measures the similarity of the machine-translated text to a set of reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality) while a value of 1 means there is perfect overlap with the reference translations (high quality). We computed the BLEU scores for the 3000 sentences of the test sed, dividing them into three groups according to their length. The first group is made from sentences with less than 50 characters. The second of sentences with a number of characters between 50 and 100, while the third one is made of sentences with more than 100 characters. In this way, I want to see if there is an effect on the length of the sentence in the final score.

4 Results

The results of the experiment are presented in Table 1. Overall, the scores are relatively low. Character-level tokenization yielded the best performance, while word-level tokenization performed the worst. WordPiece tokenization showed slightly better results than word-level, though the difference between the two is not significant. The best performance has been obtained for short sentences (less than 50 characters) by the character-level tokenization approach.

In the appendix, a sample of four pairs of translated sentences with their target sentences are shown.

Tokenization Level	S. length <= 50	50 < S. length <=100	S. length > 100	Tot
Character	0.16250	0.11960	0.11572	0.13260
Word	0.00000	0.00000	0.0001	0.00000
WordPiece	0.00021	0.00041	0.00008	0.00047

Table 1: Tokenization levels and their corresponding BLEU scores divided by sentence length.

5 Discussion

The results obtained did not align with our initial hypotheses, which predicted that WordPiece-level tokenization would outperform other approaches. Contrary to expectations, the character-level tokenization demonstrated the best performance. Analyzing the translated samples reveals that, in the other two tokenization conditions, the model failed to learn the translation task effectively. However, in the character-level tokenization, there is evidence that the model developed some understanding of the translation task.

This outcome could be attributed to the size of the vocabulary used. Given that the model employed was relatively shallow, with only a single layer in both the encoder and decoder, the character-level tokenization might have offered a more manageable and effective representation for the model to process, leading to better performance.

As a next step, an interesting question to explore would be at what model size each tokenization strategy begins to outperform the others. To investigate this, we plan to gradually scale up the model size by increasing the number of layers in both the encoder and decoder, enhancing the model's capacity to learn complex patterns and relationships in the data. Additionally, we will extend the training duration to give the model more time to converge on an optimal solution. By combining a larger model with longer training, we aim to improve performance across different tokenization methods, potentially allowing the WordPiece-level tokenization to realize its anticipated advantages.

6 Appendix

Due to the length of the code relative to this project and the fact it is distributed on multiple files, I decided to share it through a GitHub repository that is available at `https://github.com/filippo-merlo/transformer_translate_en_it.git`

The vocabulary used for the character-level tokenization:

```
[
    '<START>', ' ', '!', '"', '#', '$', '%', '&',
    "'", '(', ')', '*', '+', ',', '-', '.', '/',
    '-','-','0', '1', '2', '3', '4', '5', '6', '7',
    '8', '9', ':', '<', '=', '>', '?', '@', ';',
    '[', '\\', ']', '^', '_', '‘', '‘', '’', '“', '”',
    '...', '«', '»', 'a', 'b', 'c', 'd', 'e', 'f', 'g',
    'h', 'i', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's',
    't', 'u', 'v', 'z', 'j', 'k', 'w', 'x', 'y',
    'à', 'é', 'è', 'ì', 'ò', 'ù',
    '{', '|', '}', '~', '<PADDING>', '<END>'
]
```

Here some examples of target and translated sentences are shown.

Character level
Target sentence: leggi del tuo signore che è generosissimo.
Predicted sentence: leggi, per il tuo signore è il tuo generosi.
Target sentence: leggi, ché il tuo signore è il generosissimo,
Predicted sentence: leggi, e il tuo signore è il maggior parte della generosissimo,
Target sentence: e la parola di dio è certamente una di queste.
Predicted sentence: la preghiera del signore è sicuramente di questi.
Target sentence: ‘‘e noi (angeli) non discendiamo se non attraverso il comando del vostro signore.’’
Predicted sentence: e noi (angeli) non decenni) non aspettare la tua comanda del tuo signore (muhammad saw).

Word level
Target sentence: leggi del tuo signore che è generosissimo.
Predicted sentence: un’honda assomiglianza assomiglianza disponi assomiglianza d’istruzioni assomiglianza assomiglianza assomiglianza muori assomiglianza assomiglianza assomiglianza confiscati assomiglianza d’istruzioni assomiglianza

Target sentence: leggi, ché il tuo signore è il generosissimo,
Predicted sentence: riflettendoci assomiglianza assomiglianza mossa assomiglianza buddhisti assomiglianza assomiglianza assomiglianza muori assomiglianza assomiglianza assomiglianza confiscati assomiglianza 740 assomiglianza

Target sentence: e la parola di dio è certamente una di queste.
Predicted sentence: reali consentissero assomiglianza milena assomiglianza buddhisti assomiglianza assomiglianza assomiglianza outright assomiglianza assomiglianza assomiglianza confiscati assomiglianza confiscati assomiglianza

Target sentence: ‘‘e noi (angeli) non discendiamo se non attraverso il comando del vostro signore.’’
Predicted sentence: 2008/115 assomiglianza assomiglianza buddhisti assomiglianza buddhisti assomiglianza assomiglianza assomiglianza muori assomiglianza assomiglianza assomiglianza d’istruzioni assomiglianza d’istruzioni

WordPiece level
Target sentence: leggi del tuo signore che è generosissimo.
Predicted sentence: ##v - 2003 2003f / / frutto,,,
Target sentence: leggi, ché il tuo signore è il generosissimo,
Predicted sentence: ##v -f figlio aiuto erenza =
Target sentence: e la parola di dio è certamente una di queste.
Predicted sentence: cas00ttore smart una har etta,
Target sentence: ‘‘e noi (angeli) non discendiamo se non attraverso il comando del vostro signore.’’
Predicted sentence: i %mpo)) * ven bersateri politica ulti inoltre))
p p / * /

Acknowledgments

For the implementation of the Transformer model, I followed the guide offered by Ajay Halthor at `https://github.com/ajhalthor/Transformer-Neural-Network.git`.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.

[3] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02, page 311. Association for Computational Linguistics.

[4] Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. CCMatrix: Mining billions of high-quality parallel sentences on the WEB.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need.

[6] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation.