Adjusting Machine Translation Transformer's parameters for different input length texts for the language pair English-Spanish

Santiago Campo
scampo011@ikasle.ehu.eus

Xabi Carro
xcarro001@ikasle.ehu.eus

Lucía Palacios
lpalacios011@ikasle.ehu.eus

Abstract

This study delves into the influence of input sequence lengths on vector representation quality in Machine Translation (MT) models, specifically English-Spanish MT. We postulate that longer sequences could enhance the model's predictive abilities due to a deeper contextual understanding and greater interdependence among input tokens. We assess this hypothesis using three sub-corpora extracted from the same English-Spanish parallel corpora, categorized by sequence lengths: short (1-3 words), medium (4-10 words), and long (more than 10 words). We also explore the effect of hyper-parameter tuning on the model's performance, aiming to strike a balance between accuracy and practical constraints like training time and computational resources. The findings provide insights into optimizing sequence lengths and parameters in MT models.

1 Introduction

In recent years, neural network (NN) models have made significant strides in machine translation (MT) tasks. Pioneering work by Sutskever et al. (2014) introduced the encoder-decoder network, while Bahdanau et al. (2016) developed the attention-based architecture, and Vaswani et al. (2017) proposed the transformer model with self-attentions, which has since become the state-of-the-art. However, these models still face challenges in explicitly modeling output length. Encoder-decoder NMT models, for example, have been known to struggle with translating long sequences (Cho et al. (2014); Goodfellow et al. (2014)). Additionally, NMT systems can exhibit length bias, as demonstrated by Koehn and Knowles (2017) who found lower translation quality for very long sentences but better performance for sentences up to 60 words in length.

Each output symbol in NMT is conditioned on the complete input sequence. Nonetheless, the risk of label bias exists because it is still possible to overestimate or underestimate the likelihood of the output length sequence.

If the input sequences are considerably brief, such as individual words or short sentences, a short sequence length may be enough to capture the important information. If the input sequences are longer, such as full sentences or paragraphs, a greater sequence length may be required to ensure that the network processes the entire sequence and captures all essential information.

Taking into account the parameters of our Seq2Seq NMT we tuned them and tried different configurations in order to generate the best performance for each length sequence. The parameters we tweaked are the size of the training corpora, batch sizes, sequence length, and embedding dimensions.

We kept in all experiments the attention layer for our model, taking into account Luong et al. (2015) conclusions; (local) attention-based NMT models are superior to non-attentional ones in many cases, for example in translating names and handling long sentences.

By applying explicit length normalization to input sequences, we examined how a simple NMT model would perform under various scenarios concerning the number of words in each sequence. Following Koehn and Knowles (2017), we divided datasets into buckets based on source sentence length (1-3 word tokens, 4-10 word tokens, and +10 word tokens) and calculated corpus-level BLEU and CHRF scores for each.

The results of these experiments will be discussed in Section 5, which will provide an overview of the findings. A the repository containing the model with the default parameters can be found at the provided link.

2 Data

In this study, we conduct our experiments using the English-Spanish language pair. Large training datasets are available for these languages, which makes them an ideal choice for our investigation. We utilize an English-to-Spanish translation dataset provided by the Project (2006), a collection of bilingual sentence pairs in a tab-delimited format. The structure of the dataset consists of an English sentence, followed by a tab, the corresponding translation in Spanish, another tab, and finally the attribution of the source.

As previously mentioned, we have divided the dataset into three distinct buckets based on sequence length.

- **1-3 words.** 12456 pairs. For example: *I hit Tom. Golpeé a Tom.*
- **4-10 words.** 98666 pairs. For example: *I just need a little help. Solo necesito un poco de ayuda.*
- **Plus 10 words.** 4781 pairs. For example: *By seven o'clock in the evening, the streets are deserted in this town. Aquí las calles están desiertas a partir de las siete.*

This approach allows us to analyze the impact of different sequence lengths on the performance of our translation model, helping us identify potential patterns and trends that may emerge as a result of varying input lengths.

By evaluating the performance of our translation model on these separate buckets, we aim to better understand the role of sequence length in translation quality and provide insights into the potential improvements that can be made to the model. This analysis will also contribute to the broader field of neural machine translation research, as it helps to identify the most effective strategies for handling several input sequence lengths and their impact on translation performance.

3 Implementation

The translations in this work are provided by a seq2seq MT model, trained on a dataset of parallel sentences in English and Spanish. The model is intended to map English sentences to their Spanish counterparts. It accomplishes this by converting the input sentence into a fixed-length vector, which is then decoded into a sequence of Spanish words.

It is critical to recognize that the translation quality is dependent on the quality of the training data and the model design, and may not always be perfect. To optimize our model's performance, we conducted experiments by adjusting various parameters with the goal of obtaining the best results for each bucket of input sentences, based on sequence length. By refining these parameters, we aim to enhance the overall translation quality and gain valuable insights into the most effective strategies for improving neural machine translation across different sequence lengths.

We have based our code implementation on the Keras notebook[1] of Chollet et al. (2015). We follow their step-by-step procedure to process and train or seq2seq model for English-to-Spanish translation:

3.1 Preprocess

Downloading the data: we work with and English-to-Spanish translation dataset provided by Anki Corpus (2017)[2].

- Parsing the data: Each line in the dataset contains an English sentence (source sequence) and its corresponding Spanish sentence (target sequence). We locate the token [start] and [end] to the Spanish sentence.
- Splitting the data: we shuffle the sentence pairs and split them into the training set, validation set, and test set.
- Vectorizing the text data: we use two instances of the TextVectorization layer to vectorize the text data (one for English and one for Spanish), converting the original strings into integer sequences where each integer represents the index of a word in a vocabulary. The English layer uses the default string standardization (split on whitespace), while the Spanish layer employs a custom standardization that includes the character ¿ in the set of punctuation characters to be stripped.
- Formatting the datasets: At each train step, the model will predict target words (n + 1 and beyond) using the source sentence and the target words (from 0 to n). The training dataset takes a tuple (inputs, targets), where input is a dictionary with the keys encoding the input,

[1]https://keras.io/examples/nlp/neural_machine_translation_with_transformer/
[2]https://www.manythings.org/anki/

and target is the target sentence offset by one step.

3.2 Model

In this code, we construct a sequence-to-sequence Transformer model for machine translation. The model comprises a TransformerEncoder and a TransformerDecoder connected sequentially, and a PositionalEmbedding layer to maintain word order information. The overall process can be described in the following steps:

- Building the model:
  - The source sequence is passed through the TransformerEncoder, which generates a new representation of the input. This new representation is then passed to the TransformerDecoder along with the target sequence so far (target words 0 to N).
  - The TransformerDecoder aims to predict the next words in the target sequence (n + 1 and beyond).
  - To prevent the model from using future information when predicting token n + 1, causal masking is applied in the TransformerDecoder.
- Defining the layers:
  - The TransformerEncoder layer is created with multi-head attention, layer normalization, and a dense projection. The PositionalEmbedding layer computes token embeddings and position embeddings, and combines them to retain positional information. The TransformerDecoder layer contains two multi-head attention layers, layer normalization, and a dense projection. Causal masking is implemented within this layer.
- Assembling the end-to-end model:
  - The input layers for encoder inputs, decoder inputs, and encoded sequence inputs are defined. Positional embeddings are applied to encoder and decoder inputs.
  - The final Transformer model is created by connecting the encoder and decoder layers.
- Defining the layers:
  - The TransformerEncoder layer is created with multi-head attention, layer normalization, and a dense projection.
  - The PositionalEmbedding layer computes token embeddings and position embeddings, and combines them to retain positional information.
  - The TransformerDecoder layer contains two multi-head attention layers, layer normalization, and a dense projection. Causal masking is implemented within this layer.
- Assembling the end-to-end model:
  - The input layers for encoder inputs, decoder inputs, and encoded sequence inputs are defined.
  - Positional embeddings are applied to encoder and decoder inputs.
  - The final Transformer model is created by connecting the encoder and decoder layers.
- Training the model:
  - The model is compiled using the RMSprop optimizer and the sparse categorical crossentropy loss function.
  - Accuracy is used as a metric for monitoring training progress on the validation data.
  - The model is trained for at least 30 epochs to achieve convergence.
- Decoding test sentences:
  - A decoding function is defined that takes an input sentence, tokenizes it, and iteratively generates the next token in the target sequence until the ”[end]” token is reached.
  - The decoding function is applied to a selection of test sentences to evaluate the model’s translation performance.

We perform two diffrente methods for evaluating the quality of the translation produced by the model. The BLEU (Bilingual Evaluation Understudy) score and the chrF score. In order to do so, we implemented a function:

- Calculate BLEU and chrF score:

- It takes two arguments: a reference sentence and a candidate sentence.
- The sentences are split into tokens and then the BLEU score is computed between the reference and candidate sentences. A smoothing function is used to handle cases where there are 0 counts.
- The test English and Spanish sentences are extracted from the test pairs.
- For each English sentence in the test set, we used a implemented function to translate the sentence.
- The "[start]" and "[end]" tokens are removed from the translation.
- The BLEU score is then calculated between the translated sentence and the actual target Spanish sentence. All the BLEU scores are stored in a list.
- The chrF score is then calculated between the translated sentence and the actual target Spanish sentence. All the chrF scores are stored in a list.
- Finally, the average BLEU score is computed and printed.
- and the average chrF score is also computed and printed.

3.3 Parameters

The architecture of the model was the original from the Chollet et al. (2015) Notebook[3] The default parameters for this model are the following:

- Corpora: Split of 0.7 for the training and 0.15 for both the test and the dev set.

- Batch sizes: batches of 64 pairs, and all sequences are 20 steps long.
  - Vocab_size. Defines the number of different tokens that can be represented by the inputs_ids passed when calling the model. It is composed of 15000 tokens.
  - Sequence_length. Maximum number of tokens in a sentence. In this case, the maximum is 20.
  - Batch_size. The number of samples sent to the model at a time. The default number is 64.

[3]https://keras.io/examples/nlp/neural_machine_translation_with_transformer/

- Dimensionality of the layers. The number of dimensions in the embedding space that represents words or sentences. In this model it is set to 256 embed_dim by default.

We have played with the values of most of these parameters to observe how it affects the predictions produced by the model based on the length of the sentences it processes and generates.

4 Evaluation

The metrics used to evaluate the translations generated by the Machine Translation model are the lexical metrics BLEU and chrF.

BLEU is the most used metric in Machine Translation, in fact, it is almost the only evaluation system used even though there are several systems that offer a better or a more accurate performance. The way it works is by measuring how similar the prediction or output generated by the model is to the references with which that same model has been provided. The more similar the output is to the reference, the higher the score and the closer to 1 (maximum score). The biggest issue with this metric is that the scores do not tell us whether the machine translation is grammatically correct or whether it is intelligible or not. In fact, perfectly grammatical sentences can score a 0 if they are different from the reference. This is the main reason we have decided not to exclusively use BLEU in our analysis and have recurred to another lexical metric system: chrF.

According to Marie (2023), chrF, along with RIBES, is the only evaluation system apart from BLEU that has been used in more than two studies in the recent years among around 700 papers on Machine Translation. This metric is better than BLEU in the sense that it has a better correlation to human evaluation. In addition, chrF is tokenization independent, that is, it does not require previous tokenization, which ensures the liability of its results.

It is worth noting that we attempted to use the neural metric BLEURT to evaluate the translations but were unsuccessful due to issues we cannot control. We attempted to utilize the BLEURT metric for evaluation, however we encountered significant obstacles that made it impossible. Our first approach was to clone the GitHub repository and import the BLEURT library to run its metrics. Unfortunately, this method was unsuccessful as our

Google Colab notebook was unable to recognize the library.

In response to this issue, we tried an alternative approach by downloading the BLEURT metric from Hugging Face, both from the dataset package and directly. Although the metric was initially loaded correctly, it crashed when we attempted to use it. This problem exacerbated by the fact it caused multiple ussues with our function designed to align the source text and target.

After much consideration, and in the face of these persistent issues, we decided to give up the use of the BLEURT metric in our study. Despite the potential that BLEURT could have provided, the practical difficulties in implementing this metric led us to focus on other, more accessible evaluation methods.

5 Results

In this section we show the numbers retrieved after testing the Machine Translation model. We include the values for loss, accuracy, val_loss and val_acc, as well as the results obtained with the MT focused metrics BLEU and chrF.

Table 1 shows the results obtained from testing the model with the default settings, which are the ones mentioned earlier: a 70-15-15 partition of the corpus, a batch size of 64 with 20 steps long sequences, and 256 embedding dimensions.

N of words	**1 to 3**	**4 to 10**	**+10**
loss	0,5236	1,574	0,4329
accuracy	0,911	0,7696	0,9152
val_loss	2,6021	2,4626	4,8916
val_acc	0,6766	0,6637	0,3912
BLEU	0,02091	0.08728	0,06389
chrF	0,16735	0.25616	0.18676

Table 1: Results with the default parameters.

On our first trial we tested the model with the predefined parameters but with a different partition of the corpus: this time we made a split of 80 for the training set and 10 for the dev and test sets respectively. The results for this test are displayed in Table 2.

For our next test we maintained every setting except for the batch size, which we changed to 128. Table 3 shows the results of doubling the samples sent to the model at a time.

In Table 4 are displayed the results of maintaining a batch size of 126 samples while exaggeratedly increasing the sequence length to 80 tokens.

N of words	**1 to 3**	**4 to 10**	**+10**
loss	0,6196	1,5772	**0,4098**
accuracy	0,8978	0,7687	**0,9182**
val_loss	2,467	2,3676	**5,0017**
val_acc	0,6904	0,6788	**0,4033**
BLEU	0,02118	0.08488	0,07459
chrF	0,16866	0.25024	0.18676

Table 2: Results with 80% training set, 10% both test and dev sets.

N of words	**1 to 3**	**4 to 10**	**+10**
loss	**0,3297**	1,1065	1,1452
accuracy	**0,9243**	0,8117	0,7515
val_loss	**2,431**	2,2265	4,447
val_acc	**0,6786**	0,6838	0,3816
BLEU	0,02028	0.09317	0,05686
chrF	0,16651	0.27806	0,15674

Table 3: Results with 128 batch size.

N of words	**1 to 3**	**4 to 10**	**+10**
loss	0,329	1,1007	1,1775
accuracy	0,9248	0,8133	0,7443
val_loss	2,449	2,2094	4,5196
val_acc	0,6752	0,6878	0,374
BLEU	0,02114	0.0928	0.05959
chrF	0,16636	0.2781	0.17053

Table 4: Results with 128 batch size and 80 sequence_length.

We then proceeded to reset the sequence length to 20 while maintaining a batch size of 128 samples, but also doubling the dimensions of the embedding space to 512. The results for this test are shown in Table 5.

Lastly, Table 6 displays the results for our last test. This time we maintained the changes in batch size and dimensions of embeddings but decreased the number of tokens in sequence_length to 10.

We devote the following section to interpret and discuss the results we obtained from testing the model by changing the parameters each time.

6 Discussion

From the results, several hypothesis can be drawn regarding the impact of different parameters on

N of words	1 to 3	4 to 10	+10
loss	0,662	**0,6732**	0,5457
accuracy	0,8802	**0,8575**	0,8764
val_loss	2,5473	**2,1693**	4,7107
val_acc	0,6787	**0,7024**	0,3967
BLEU	0.01992	error	0.07005
chrF	0.16714	error	0.19227

Table 5: Results with 128 batch size, 20 sequence_length and 512 embed_dim.

N of words	1 to 3	4 to 10	+10
loss	0,3792	0,7193	3,7858
accuracy	0,9166	0,85	0,4801
val_loss	2,5633	2,304	5,0966
val_acc	0,6761	0,6918	0,337
BLEU	0.02206	error	error
chrF	0.16908	error	error

Table 6: Results with 128 batch size, 10 sequence_length and 512 embed_dim.

the performance of the Machine Translation (MT) model.

Our results suggest that the model's performance varies significantly based on the length of input sequences. For short sequences (1-3 words), the model consistently achieved the lowest loss and highest accuracy across all tests. However, it's interesting to note that as the sequence length increased (more than 10 words), the model's performance, in terms of validation loss and validation accuracy, deteriorated. This suggests that the model struggles to maintain quality translations with longer sequences, which is an area for further investigation and improvement.

Doubling the batch size from 64 to 128 resulted in a noticeable improvement in model loss and accuracy for short sequences. It's also noteworthy that the model showed improvement in performance for medium-length sequences (4-10 words). The validation accuracy, however, did not show a substantial change. This indicates that increasing the batch size might improve the model's ability to learn from the training data but does not necessarily enhance its performance on unseen data.

When we doubled the embedding dimensions from 256 to 512, the model showed an improvement in accuracy for medium-length sequences, but it resulted in higher loss for the same. For longer sequences, the BLEU and chrF scores could not be calculated due to errors, which is due to the increased computational complexity. This suggests a trade-off between the model's ability to capture more nuanced relationships in the data (with larger embedding dimensions) and the computational efficiency of the model.

Altering the partition of the corpus to 80-10-10 (training-validation-test) from 70-15-15 did not lead to significant improvements or deteriorations in model performance. This indicates that the model is not highly sensitive to the partitioning of the corpus, at least within the ranges tested.

It's interesting to note that the BLEU and chrF scores vary significantly depending on the length of the sentences. For sentences with 1 to 3 words, the scores do not change significantly across the different configurations. For sentences with 4 to 10 words, the highest BLEU and chrF scores were achieved in Table 3, when the batch size was increased to 128. For sentences with more than 10 words, the scores varied quite a bit, with the highest BLEU score achieved in Table 2 (80% training data), and the highest chrF score in Table 4 (sequence length 80). In the last two configurations (Tables 5 and 6), the model failed to produce BLEU and chrF scores for sentences with 4 to 10 words and more than 10 words, because of the computational limitation in the model's performance for longer sentences with these settings.

These results suggest that increasing the batch size and embedding dimensions can improve the model's performance in terms of loss, accuracy, and translation quality (as measured by BLEU and chrF scores) for sentences of up to 10 words. However, the model struggles with longer sentences, and further adjustments or different approaches may be needed to improve its performance on sentences with more than 10 words.

As we already mentioned, the translation's quality is dependent on the model architecture. Employing this pre-trained Transformer from Keras we are not able to tune all the parameters that are necessary for a highly accurate translation. As a consequence of this, we obtained some translated results that did not correspond properly to their English references.

7 Conclusion

Our investigation underscores the crucial role of input sequence lengths and hyperparameter optimization in modeling the performance of Machine Translation (MT) models. We found that shorter sequences (1-3 words) tell us superior results concerning model loss and accuracy. In contrast, the model encounters difficulties when facing longer sequences (over 10 words), indicating an opportunity for future research.

The translation quality metrics, BLEU and chrF scores, demonstrated considerable variance dependent on sentence length. Sentences built by 4-10 words tell us the most impressive scores when the batch size was amplified to 128. Despite this, sentences exceeding 10 words presented computational obstacles for the model, preventing the calculation of BLEU and chrF scores under certain configurations.

In essence, the study highlights the promising advantages of enhancing the efficiency of MT models through the optimization of sequence lengths and hyperparameters. Yet, it also accentuates the need for continued exploration, especially concerning longer sequences. The issues encountered in managing longer sentences indicate that supplementary modifications or innovative methodologies may be necessary to improve the model's performance and translation quality. The knowledge recollected from this research stands to encourage the development of more proficient MT models, thereby enriching the quality of English-Spanish translations and potentially offering a model for improvements across additional language pairs.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

François Chollet et al. 2015. Keras. https://keras.io.

Anki Corpus. 2017. Anki: Tab-delimited bilingual sentence pairs. https://www.manythings.org/anki.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation.

Benjamin Marie. 2023. Traditional versus neural metrics for machine translation evaluation. 100+ new metrics since 2010.

Tatoeba Project. 2006. Tatoeba: Collection of sentences and translations. https://tatoeba.org/en.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.