Data ScienceTech Institute

DEEP LEARNING PROJECT

S23 Cohort, Applied MSc in Data Engineering

DEEP LEARNING FOR NLP: MACHINE TRANSLATION

DEEP LEARNING

Performed by: Gali Aydaraliev

MSc in Data Engineering 2023-2024

Data ScienceTech Institute, SPOC

Date: 23.04.2024

TABLE OF CONTENTS

I. INTRODUCTION

Machine translation is the method of utilizing artificial intelligence, namely deep learning mechanisms, which allows you to translate text from one language to another. Using neural network architectures, particularly sequence-to-sequence models and transformers, deep learning systems have demonstrated remarkable results in translating text between languages. However, despite their capabilities, the main weakness of such systems is the quality assessment, since machines have difficulty capturing the aspects of different language connections and interactions. Nevertheless, it is currently possible to create a system capable of translating with relatively good accuracy. This goal of this project is to prove this by revealing all the basic deep learning processes related with machine translation and examining various quality indicators.

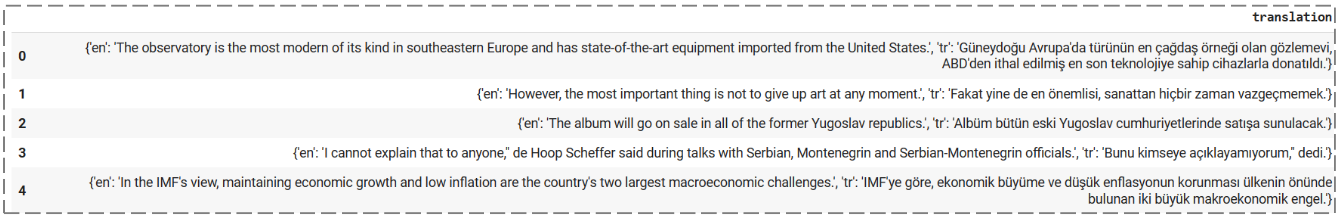
This report describes the methodology used, the learning process, and the benchmarks used to evaluate the effectiveness of the system. The report will examine the potential of deep learning models in translation challenges.

It should also be noticed that all model building and training operations were carried out using *Google Colab* platform.

II. EXPERIMENTS

1. DATASETS

Before building a neural translation model, it is necessary to determine the source and target languages and the corresponding dataset. Thus, in accordance with the chosen case, the language translation model should translate English sentences into their Turkish counterparts. One of the most popular datasets for this purpose is from *WMT (Workshop on Statistical Machine Translation)*. It is freely available on *HuggingFace datasets* and can be downloaded using `Datasets` library. Below are a few random pairs of sentences from there.



	translation
0	{'en': 'The observatory is the most modern of its kind in southeastern Europe and has state-of-the-art equipment imported from the United States.', 'tr': 'Güneydoğu Avrupa'da türünün en çağdaş örneği olan gözlemevi, ABD'den ithal edilmiş en son teknolojiye sahip cihazlarla donatıldı.'}
1	{'en': 'However, the most important thing is not to give up art at any moment.', 'tr': 'Fakat yine de en önemlisi, sanattan hiçbir zaman vazgeçmemek.'}
2	{'en': 'The album will go on sale in all of the former Yugoslav republics.', 'tr': 'Albüm bütün eski Yugoslav cumhuriyetlerinde satışa sunulacak.'}
3	{'en': 'I cannot explain that to anyone," de Hoop Scheffer said during talks with Serbian, Montenegrin and Serbian-Montenegrin officials.', 'tr': 'Bunu kimseye açıklayamıyorum," dedi.'}
4	{'en': 'In the IMF's view, maintaining economic growth and low inflation are the country's two largest macroeconomic challenges.', 'tr': 'IMF'ye göre, ekonomik büyüme ve düşük enflasyonun korunması ülkenin önünde bulunan iki büyük makroekonomik engel.'}

Figure 1 – The example of dataset sentence pairs

The "WMT 18 TR-EN" dataset consists of more than 210 thousand pairs of sentences in English and Turkish of which 6k are evenly distributed among the sets for testing and validation. It is definitely that processing such a large dataset will require a lot of time and resources but the model will be able to provide higher performance with such input data.

2. METHODOLOGY

It is worth noting that there will be no description of the design of the neural architecture. This research project used a pre-trained from *HuggingFace transformer model* (namely `Helsinki-NLP/opus-mt-tc-big-en-tr`). In addition, almost all basic elements for the translation task were derived from this example: *(https://github.com/huggingface/notebooks/blob/main/examples/translation.ipynb).*

2.1 TOKENIZATION

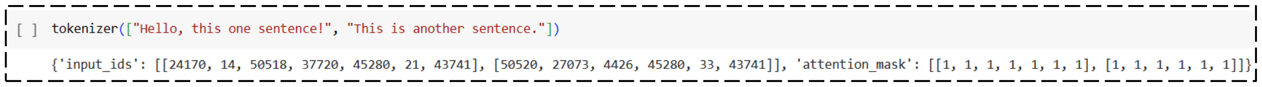
It is impossible for a neural network to work with text directly so it must be converted into data that the machine can understand. This can be done by converting each word in the dataset to the corresponding integer using `Tokenizer` function. The tokenizer must be set for both the source and target languages since their vocabularies are different.

It should be noted that the length of the sequence in the source and target languages is different. This is due to the fact that texts with the same meaning may contain a different number of words in two languages. This needs to be fixed so that all text sequences have the same length. For this purpose, the data must be split and processed in batches where an input data (the length of which exceeds what the selected model can handle) must be reduced to the maximum length accepted by the model. In addition, since sentences have a dynamic length, the padding must be added to the sequences so that they are of the same length.

To do all this, the tokenizer was instantiated with `AutoTokenizer.from_pretrained` method, which ensures that:

- the tokenizer will match to the desired model architecture
- the vocabulary that was used for selected pretraining model will be downloaded

The example below shows the tokenized samples without padding, so there are no zeros in the attention mask. However, the padding was implemented later using `data_collator` function.



```
[ ]  tokenizer(["Hello, this one sentence!", "This is another sentence."])

     {'input_ids': [[24170, 14, 50518, 37720, 45280, 21, 43741], [50520, 27073, 4426, 45280, 33, 43741]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1]]}
```

Figure 2 – The example of sentence tokenization

2.2 MODEL EVALUATION

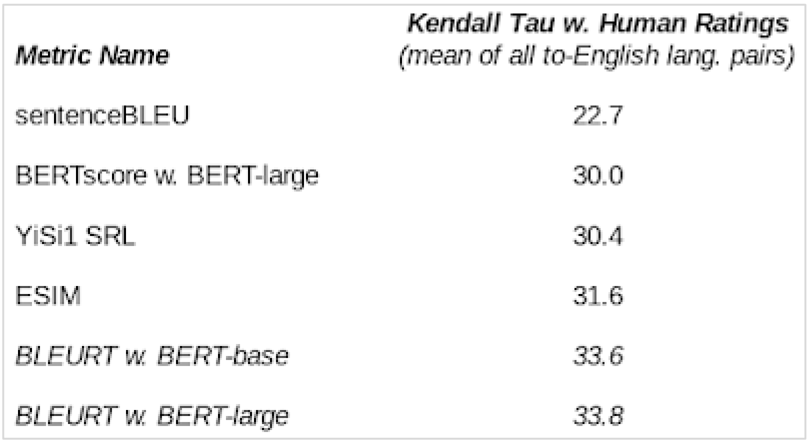
Estimating the quality of machine translation is essential for the project. According to the assignment, it was necessary to find indicators, which could be the most suitable As a result, the evaluation of the MT system was processed according to the following quality metrics:

1. **BLEU (bilingual evaluation understudy**, based on precision in 4-grams, measures how closely machine-produced translations match human references (result ranges from 0 to 1, higher score indicates better translation quality)

2. **Google BLEU** is a variant of the *BLEU* metric. *Google BLEU* scores are typically reported alongside other evaluation metrics to provide a comprehensive assessment of the quality of machine translation produced by Google Translate (result ranges from 0 to 1, higher score indicates better translation quality)

3. **METEOR (metric for evaluation of translation with explicit ordering)** is similar to *BLEU* but includes additional steps, like considering synonyms and comparing the stems of words. *METEOR* correlates better with human judgment than *BLEU* (result ranges from 0 to 1, higher score indicates better translation quality)

4. **BLEURT (Bilingual Evaluation Understudy with Representations from Transformers)** is an extension of the *BLEU* metric where pre-trained transformer-based language representations are used to compute a more robust similarity score between generated text and reference text. Higher *BLEURT* scores indicate better quality, while lower scores suggest poorer quality. According to the figure below, this metric has better correlation with human assessment than the previous ones.



Metric Name	*Kendall Tau w. Human Ratings (mean of all to-English lang. pairs)*
sentenceBLEU	22.7
BERTscore w. BERT-large	30.0
YiSi1 SRL	30.4
ESIM	31.6
BLEURT w. BERT-base	*33.6*
BLEURT w. BERT-large	*33.8*

Figure 3 - Automated metrics correlation with human assessment

After all, each of the presented metrics has its own set of strengths and weakness, which indicates that they should all be considered only in aggregate. It is important to combine simple but understandable metrics (*like BLEU*, *METEOR*) with complex neural metrics such as *BLEURT* to see the full performance of the MT model. In the project, all of these metrics were implemented using the `evaluate.combine` method.

2.3 MODEL FINE-TUNING

When the data is ready and estimated metrics are defined, the next phase will be to start fine-tuning the model. The fine-tuning procedure consists of the following steps:

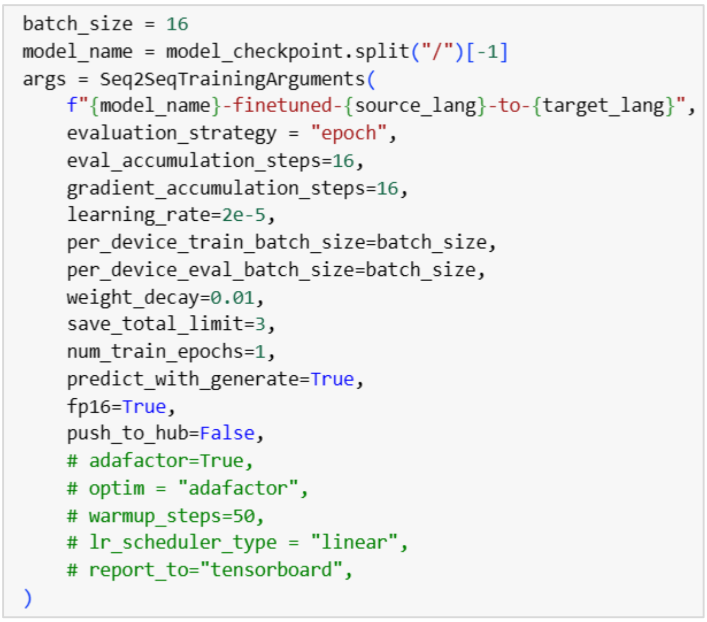
- Load a pre-trained model suitable for text translation
- Define training arguments, including the evaluation strategy, batch sizes, and the number of training epochs
- Create a `Trainer` instance with model, training arguments, and customized evaluation metrics
- Fine-tune the model on the training dataset

All these steps are important to comprehend model fine-tuning using *Hugging Trainer API*. That is why the operations performed should be described in a little more detail.

To begin with, since the task is of the sequence-to-sequence kind, `AutoModelForSeq2SeqLM` class was used. As in the case of the tokenizer, the `from_pretrained` method will allow to load a pretrained model with its weights and parameters.

After that, it was necessary to instantiate `Seq2SeqTrainingArguments`, which is a class that contains all the attributes for processing training. It worth noting that the default batch size for training and evaluation has been updated to 16; the gradient and evaluation accumulation steps have been added, which have been set to 16; the number of training epochs has been left equal to 1 (*will be explained in results section*).

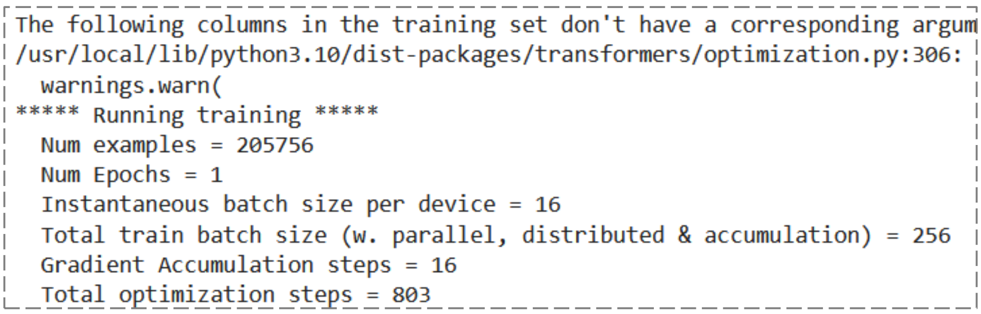
The figure below shows all hyperparameters that were utilized to train MT model, but not all of them were included as the final inputs. All of these training arguments were extracted from the *HuggingFace training arguments documentation.*



```
batch_size = 16
model_name = model_checkpoint.split("/")[-1]
args = Seq2SeqTrainingArguments(
    f"{model_name}-finetuned-{source_lang}-to-{target_lang}",
    evaluation_strategy = "epoch",
    eval_accumulation_steps=16,
    gradient_accumulation_steps=16,
    learning_rate=2e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    weight_decay=0.01,
    save_total_limit=3,
    num_train_epochs=1,
    predict_with_generate=True,
    fp16=True,
    push_to_hub=False,
    # adafactor=True,
    # optim = "adafactor",
    # warmup_steps=50,
    # lr_scheduler_type = "linear",
    # report_to="tensorboard",
)
```

Figure 4 – Hugging Face training arguments in the model

The final part of the process was to transfer the training arguments to the `Seq2SeqTrainer` along with model, train and evaluation datasets, data collator, tokenizer and computer metrics *(a function used to calculate metrics during evaluation).* In the end, the model was finally prepared for calling the `train` method, the result of which looks like this:

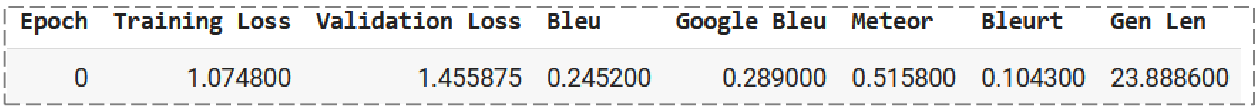


```
The following columns in the training set don't have a corresponding argum
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:306:
  warnings.warn(
***** Running training *****
  Num examples = 205756
  Num Epochs = 1
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 256
  Gradient Accumulation steps = 16
  Total optimization steps = 803
```

Figure 5 – The description of training parameters

III. RESULTS

It is time to check the results that were obtained upon training the machine translation model and define a high-quality solution for MT. For this purpose, various combinations of hyperparameters have been implemented in the train dataset to achieve the best quality scores in the validation sets. The following result was found to be the most appropriate.

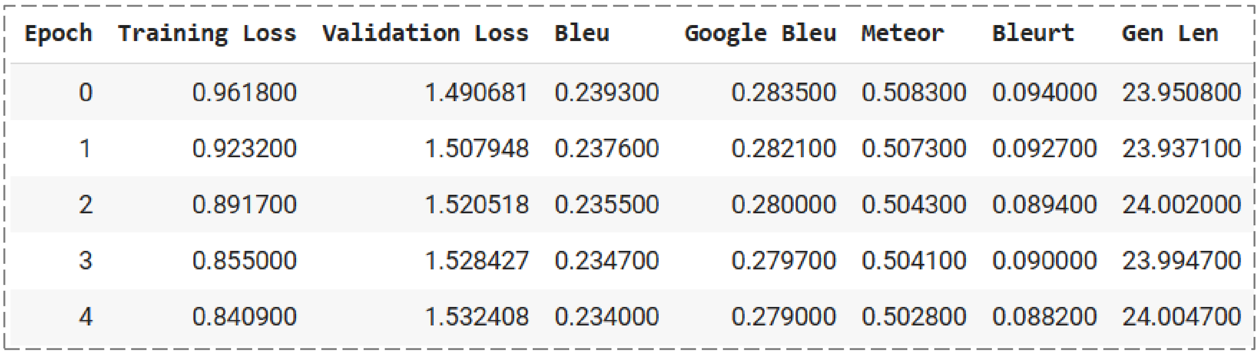


Epoch	Training Loss	Validation Loss	Bleu	Google Bleu	Meteor	Bleurt	Gen Len
0	1.074800	1.455875	0.245200	0.289000	0.515800	0.104300	23.888600

Figure 6 – The model performance output (with one epoch)

METEOR has been shown to give a higher score than other evaluation metrics, which means that the model does well with indicators such as binding and synonymy but the semantics of the text are probably not so good. A lower *BLEURT* score may confirm this assumption, since this specialized metric reflects the semantic similarity between texts since this indicator was trained using a supervised, learned function using human translation scores. As for *BLEU* and *Google BLEU*, they are not as representative but they can be used as benchmarks for future training experiments.

Surprisingly, an increase of epochs has a negative effect on the performance of the model. This can be seen from the figure below. A general decrease in the accuracy of predictions is observed during the five training epochs. Moreover, the validation losses gradually increase as the training losses decrease. This definitely indicates that the model is overfitted and this can not be an optimal solution for implementation. That is why the result with one epoch was considered acceptable under the given conditions.



Epoch	Training Loss	Validation Loss	Bleu	Google Bleu	Meteor	Bleurt	Gen Len
0	0.961800	1.490681	0.239300	0.283500	0.508300	0.094000	23.950800
1	0.923200	1.507948	0.237600	0.282100	0.507300	0.092700	23.937100
2	0.891700	1.520518	0.235500	0.280000	0.504300	0.089400	24.002000
3	0.855000	1.528427	0.234700	0.279700	0.504100	0.090000	23.994700
4	0.840900	1.532408	0.234000	0.279000	0.502800	0.088200	24.004700

Figure 7 – The model performance output (with five epoch)

However, despite the moderate quality metrics, it was decided to make sure that the trained model at least worked properly, so it was necessary to test the machine by translating some random sentence. It is worth noting that Yandex Translator was used as an analog of translation to compare the results (*Figure 8*). After implementing a translation function, the model was able to successfully translate a complex English sentence into Turkish. The translation is not completely identical to the translation from Yandex, but the actual meaning of both of them is similar. This proves that the created MT system is able to generate texts for translation almost accurately and fluently. Moreover, it can be assumed that the model will be able to improve its prediction capabilities if it receives more data than it currently has (*now there are only 210 thousand pairs*).

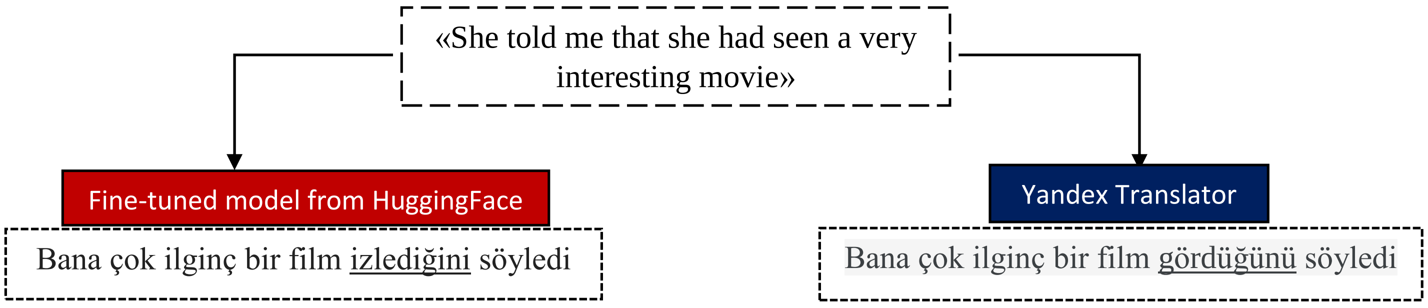

«She told me that she had seen a very interesting movie»
Fine-tuned model from HuggingFace
Bana çok ilginç bir film izlediğini söyledi
Yandex Translator
Bana çok ilginç bir film gördüğünü söyledi

Figure 9 – The translation comparison between models

IV. CONCLUSION

Within the framework of the project, the main actions necessary to build a deep learning systems for natural language processing were presented. Using *HuggingFace* tools, in particular sequence-to-sequence models and transformers, it has become possible to develop a reliable and efficient machine translation model from scratch.

It should be noted that the machine translation system has demonstrated good efficiency in creating high-quality translations by selected language pairs. Although the results obtained in terms of model quality are not so good, but it should be recognized that the field of MT quality measurement is constantly evolving. Undoubtedly, the time will come when machine will be able to accurately distinguish all the specific aspects of language mechanisms (like semantics, grammar, idioms, etc.), so it is necessary to continue developing advanced MT systems, which will certainly lead to more accessible and inclusive communication between languages and cultures.