DEEP LEARNING WITH PYTHON

Course project : Text Translation

Joanne ADAM
Data ScienceTech Institut
joanne.adam@edu.dsti.institute

Introduction

The goal of the "Deep learning with Python" course's project is to train a language model on a translation task between two languages. Because of my Alsatians origins, I chose to work on the translation from French to Alsatian, a French regional dialiect in Alsace (North-East of France). Alsatian dialect is a non-standard, spoken language with limited written ressources. After a thorough research, no pre-trained transformer exists, to my knowledge, for a French to Alsatian translation task. Therefore the dataset gathering task must be carefully executed to develop a robust model.

This report presents the steps undertaken to perform this study: data collection, database pre-processing, model fine-tuning and model evaluation.

Complementary materials (source files, models, datasets...) are availaible in github : github.com/JoanneAB/translator_fr-als.

Data collection

Words and sentences in French and their translation in Alsatian have been downloaded from the Internet from several sources. For each of the sources, a python script have been writen to extract and clean the data to generate a database in the python's `datasets.Dataset` format.

www.alsa-immer.eu

The `www.alsa-immer.eu` website [1] have been thoroughly downloaded. The words and sentences are already labelised using HTML tags *e.g.* `<als>...</als>` and `<fr>...</fr>` for Alsatian and French language respectively. The existing labelisation improves the efficiency of the data selection. The resulting dataset consists of 1791 sentences with a total of 13272 Alsatian words and 15217 French words.

www.alsatext.eu

Parts of `www.alsatext.eu` website [2] have been downloaded for which words and sentenced are labelised for language using HTML tags (*e.g.* `<ex_als>...</ex_als>` and `<ex_fr>...</ex_fr>` for Alsatian and French languages respectively). The resulting dataset consits of 7912 sentences with a total of 15839 Alsatian words and 17147 French words.

Alsatian - French lexicon

The lexicon provided by [3] is a compilation of four lexicons sources from the French Alsace region. It contains the French-Alsatian translation of 5036 words.

Global dataset

A total of 14739 sentences have been collected with a total of 34312 French words and 37400 Alsatian words. Figure 1 shows the distribution of number of words per sentence for the French and Alsatian dataset. We see that most of the sentences have five to ten words, which corresponds to relatively short sentences.

There are no labels defining the context, topic of the text... Systematic check with human eye is not applied.

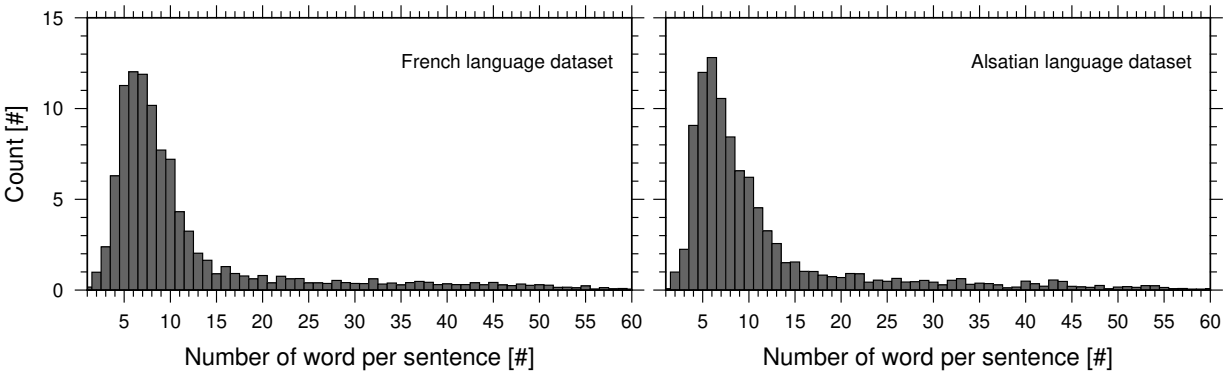

Figure 1: Distribution of the number of words per sentence for the French and Alsatian datasets.

The dataset is divided into three independant sets for training, evaluation and testing the model. The ratio for each set will be further discussed in this report. Listing 1 presents the piece of code writen to generate the dataset in the appropriate format.

```python
from sklearn.model_selection import train_test_split
from datasets.dataset_dict import DatasetDict
from datasets import Dataset

from database_object import database # user-writtend class to read and store the dataset

db = database()

# Read and store the dataset from the several sources:
db.get_data_alsaimmer()
db.get_data_alsatext()
db.get_data_motsAlsacienMulhouse()
db.get_data_alignments()

# Split the dataset into training/validation/testing sets:
train, validtest = train_test_split(db.db    , test_size=0.6, random_state=0) # 60% training
valid, test      = train_test_split(validtest, test_size=0.5, random_state=0) # 20%-20% valid, test

# Create the dataset dict:
d = {'train'     : Dataset.from_dict({'translation': train}),
     'validation': Dataset.from_dict({'translation': valid}),
     'test'      : Dataset.from_dict({'translation': test})
     }
d = DatasetDict(d)
```

Listing 1: Part of Python script to generate the dataset

Pre-processing the dataset

Each entry of the dataset is converted into tokens to transform the dataset from raw text to a numerical format that is optimized for the modeling task. This process ensures consistent input formatting and enables the translation algorithm to learn more effectively meaningful patterns.

Various types of tokenizers can be used, each offering different strategies. In this project I considered the tokenizers developed by Google (`google-t5`) or Meta (`Facebook/mbart-large-50-many-to-many-mmt`). (see code presented in Listing 2). These models will be described in more details in the following section.

```python
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, DataCollatorForSeq2Seq, Seq2SeqTrainingArguments, Seq2SeqTrainer, pipeline
from transformers import NllbTokenizer
from transformers import AutoConfig

```

```python
# Google Tokenizer:
checkpoint = "google-t5/t5-base"
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
config = AutoConfig.from_pretrained(checkpoint)

# Facebook tokenizer
#checkpoint = "facebook/mbart-large-50-many-to-many-mmt"
#tokenizer = NllbTokenizer.from_pretrained(checkpoint)

# translate French to Alsatian:
tokenizer.src_lang = "fr"
tokenizer.tgt_lang = "als"

# Create the tokenized dataset:
tokenized_datasets = d.map(encode, batched=True, fn_kwargs={"tokenizer":tokenizer})
```

Listing 2: Part of Python script to tokenize the dataset

Fine tuning

A pre-trained model is tuned and the model's parameters are adjusted to improve the translation from French to Alsatian. The adjusted parameters are : ratios between training, evaluation and testing datasets, choice of pre-trained model, learning rate, batch size and weight decay. Table 1 presents a selection of tested models and their corresponding parameters. Listing 3 presents the code used to generate a model to translate text from French to Alsatian languages.

	Train set [%]	Valid set [%]	Test set [%]	Pre-trained model	Learning rate	Batch size	Weight decay	Color
model 18	60	20	20	Google-T5/T5-small	2e-4	16	0.01	
model 16	60	20	20	Google-T5/T5-base	2e-4	16	0.01	
model 17	60	20	20	Facebook/mbart-large-50	2e-4	16	0.01	
model 05	60	20	20	Google-T5/T5-base	2e-5	8	0	
model 08	80	10	10	Google-T5/T5-base	2e-5	8	0	
model 19	60	20	20	Google-T5/T5-base	2e-4	8	0	
model 06	60	20	20	Google-T5/T5-base	2e-4	8	0.01	
model 15	60	20	20	Google-T5/T5-base	2e-4	8	0.05	
model 14	60	20	20	Google-T5/T5-base	1e-6	8	0	
model 20	60	20	20	Google-T5/T5-base	2e-4	32	0.01	

Table 1: Selection of models that are presented in this report with their corresponding model parameters. The color represents the color used in the following figures for each model.

```python
# Options for the training function:
args = Seq2SeqTrainingArguments(
    "my_model",
    eval_strategy = "epoch",
    save_strategy = "epoch",
    learning_rate = 1e-6, # varying parameters (table 1)
    per_device_train_batch_size = 8, # varying parameters (table 1)
    per_device_eval_batch_size  = 8, # varying parameters (table 1)
    save_steps       = 512, # save every 512 steps.
    weight_decay     = 0, # varying parameters (table 1)
    save_total_limit = 1,
    num_train_epochs = 15,
    predict_with_generate = True,
    fp16                  = True, # True to speed up training on GPU
    metric_for_best_model = 'eval_loss',
    greater_is_better     = False,
    load_best_model_at_end= True,
```

```
    seed=1,
  )

data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)

trainer = Seq2SeqTrainer(model, args,
    train_dataset    = tokenized_datasets["train"],
    eval_dataset     = tokenized_datasets["validation"],
    data_collator    = data_collator,
    processing_class = tokenizer,
    compute_metrics  = compute_metrics, # function for model evaluation
)

# model training:
trainer.train()
```

Listing 3: Part of Python script to generate a translation model.

Choice of pre-trained model

A pre-trained model is applied and trained with the new French-Alsatian dataset to allow translation from French to Alsatian. The choice of pre-model plays a critical role in determining the performance of the translation model. Several pre-trained models have been tested to identify the most suitable for this project. Factors such as dataset size and computational resources were taken into account to determine the most effective model.

Thanks to its high number of included languages (200) and parameters (54.5 billions), the `Facebook/NLLB-200` model (No Language Left Behind model [5]) seems to be a good candidate for this project. However, due to its large size and RAM restriction from Google Colab environment, this pre-trained model could not be used.

A second pre-trained model candidate is `Facebook/mBART` (Many-to-Many Multilingual Machine Translation [6]). This model is pre-trained and fined-tuned on 50 languages and more than 100 million parameters.

The third and last pre-trained model used in this project if the `Google-T5/T5-base` model [4]. This language model is pre-trained and fine-tuned on four languages and 220 million parameters and could be a good compromise for efficiency and performance. To test performance and computational cost, `Google-T5/T5-small` [4] is also considered.

Figure 2 presents the training and evaluation loss for models the pre-trained models: `Google-T5/T5-small` (table 1, model 18), `Google-T5/T5-base` (1, model 16) and `Facebook/mbart-large-50-many-to-many-mmt` (table 1, model 17).

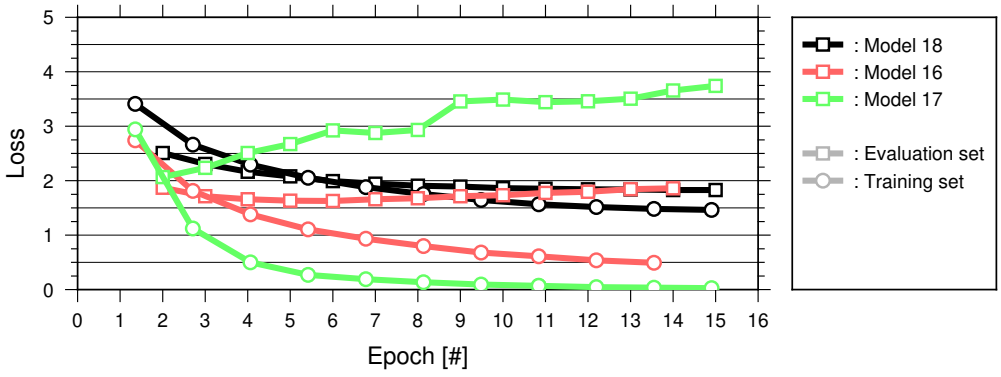

Figure 2: Loss curves for training (solid lines) and evaluation (dashed lines) sets of model 18 (`Google-T5/T5-small`, black), model 16 (`Google-T5/T5-base`, red) and model 17 (`Facebook/mbart-large-50-many-to-many-mmt`, green). See table 1 for more information on the model parameters.

Figure 2 shows that whatever the pre-trained model, the evaluation loss curves are above the training loss curves after few epochs. The models perform better on known data. Each and every model would fail in predicting a robust translation for data on which they have not been trained on and that are unseen. Moreover, models 17 and 16 that used the `Facebook/mbart-large-50` and `Google-T5/T5-base` pre-trained model respectively, show a widening of the gap between the training and evaluation loss curves. This indicates overfitting. These models memorize the patterns in the training dataset and perform poorly at generalizing to unseen data during the evaluation step.

Pre-trained models `Facebook/mbart-large-50` (model 17) and `Google-T5/T5-base` (model 16) will no longer be considered as a good pre-trained model for this study. Indeed, pre-trained model `Google-T5/T5-base` show very large evaluation and training loss values. Although the training loss is very small for the `Facebook/mbart-large-50` pre-trained model, its inefficiency to provide a robust translation on unknown words (large values of evaluation loss) excludes him as a robust candidate.

Choice of train/valid/test dataset ratio

Figure 3 presents the effect of changing the ratio between the training and evaluation sets. The training set of model 5 contains 60% of the total dataset while the training set of model 8 contains 80% of the total dataset. The remaining dataset is divided into two equal parts for evaluation and testing.

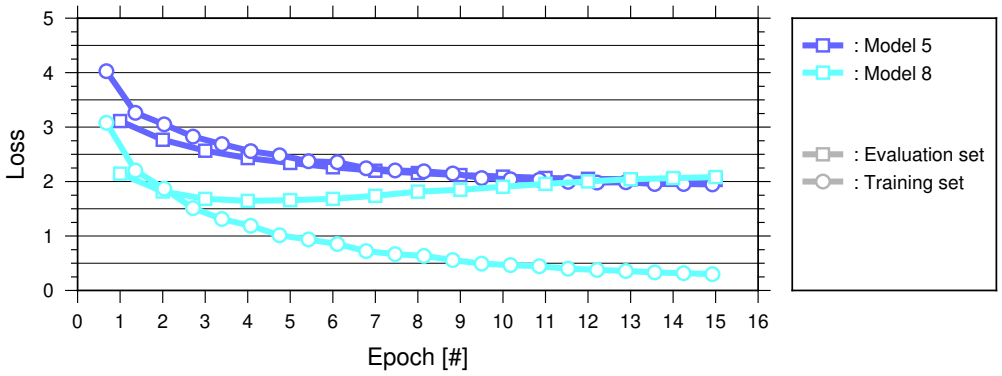

Figure 3: Loss curves for training (solid lines) and evaluation (dashed lines) sets of model 5 (training set is 60% of the dataset, dark blue) and model 8 (training set is 80% of the dataset, light blue). See table 1 for more information on the model parameters.

Increasing the amount of training data does not improve the quality of the model. Model 8 is overfitting and can not deliver a robust and accurate translation.

Choice of model training parameters

Batch size, weight decay and learning rate model parameters have been adjusted in order to find the model that produces the more accurate and robust translation. Figures 4, 5 and 6 present models with varying batch size, weight decay and learning rate, respectively.

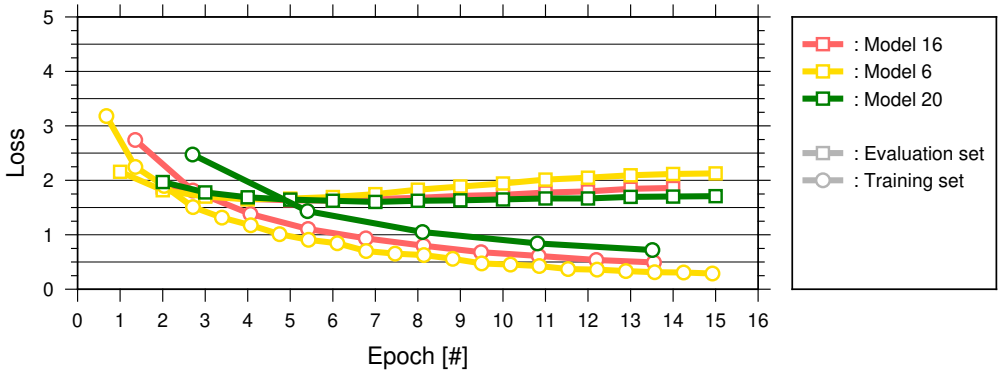

Figure 4: Loss curves for training (solid lines) and evaluation (dashed lines) sets of model 16 (batch size of 16, red), model 6 (batch size of 8, yellow) and model 20 (batch size of 32, dark green). See table 1 for more information on the model parameters.

The batch size is the number of training examples processed by the algorithm per step. Adjusting the batch size could help finding the correct balance between speed of execution, memory requirements and stability of the model. Nonetheless, one may note that the maximum batch size has been constrained to 32 due to RAM limitations in Google Colab (platform where all the computation have been performed). Figure 4 show that changing the value of the batch size does not improve significantly the quality of the model.

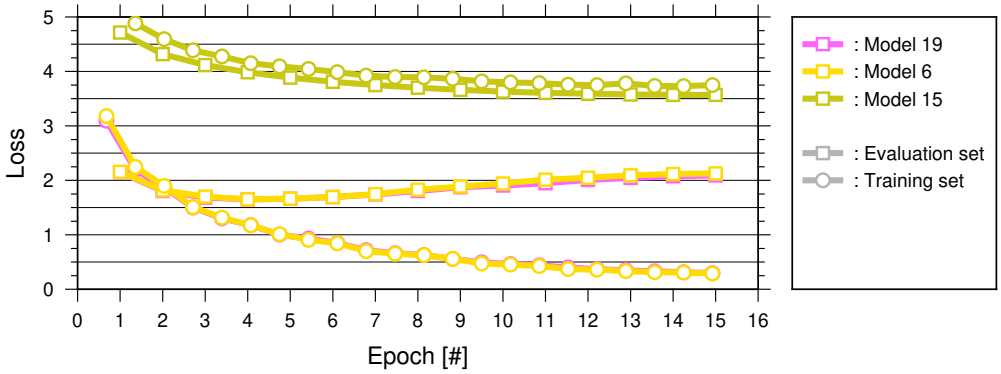

Figure 5: Loss curves for training (solid lines) and evaluation (dashed lines) sets of model 19 (weight decay of 0, pink), model 6 (weight decay of 0.01, yellow) and model 15 (weight decay of 0.05, green). See table 1 for more information on the model parameters.

The weight decay parameter is a regularization parameter to constrain the weight of the parameters and help prevent overfitting. Figure 5 presents models with similar parameters but the weight decay. Results show that models with small or no weight decay coefficient (0.01 for model 6 and 0 for model 19) show better results with small training and evaluation loss values.

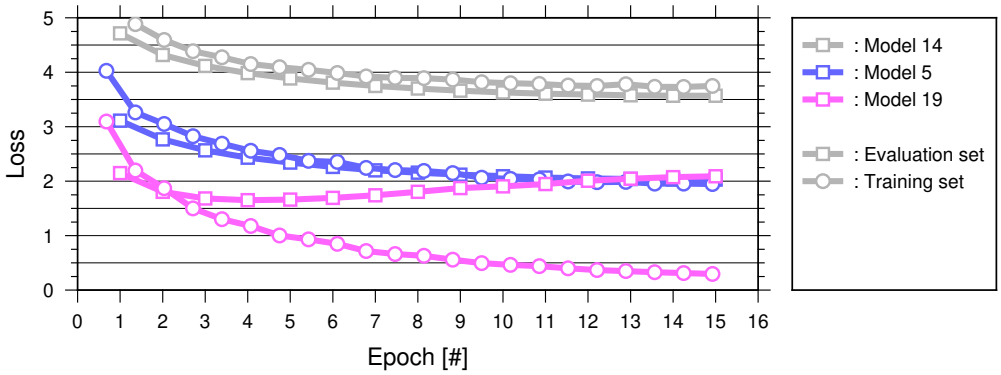

Figure 6: Loss curves for training (solid lines) and evaluation (dashed lines) sets of model 14 (learning rate of $1e^{-6}$, gray), model 5 (learning rate of $2e^{-5}$, dark blue) and model 19 (learning rate of $2e^{-4}$, pink). See table 1 for more information on the model parameters.

The learning rate controls the speed of learning. A too small learning rate could lead to slow and failing to converge while a too large learning rate could lead to missing the soluting and failing to converge. Indeed, figure 6 shows that with a very small learning rate (1e-6), the model 14 fails to converge to an accurate model. The model is learning a solution (the training and evaluation loss curves are decreasing) but the learning is slow and the model still show high values of loss and is not accurage. A large learning rate for the model 19 (2e-4) shows that the convergence to an accurate solution is fast (training loss is very small) but the solution is not robust with training loss larger than evaluation losses. The model is over-fitting and would fail to translate unknown data.

Evaluation

The goal of the evaluation process is to quantify the quality of the model for it to be as good as a human-quality translation. The evaluation process includes an automatic step with the compuation of metrics (*e.g.* BLEU) and a human evaluation step on a subset of translations to check for accuracy and fluency.

Automatic metric

In order to measure the precision of the translation, the BLEU (Bilingual Evaluation Understudy) score is computed after each epoch (see listing 4 for python function used to compute the BLEU score). The BLEU score is well suited for evaluation of translation models, where precision (grammar, contextual sentences...) is important. BLEU score measures how many words in the generated translation appear in the reference text by used the tokenized version of the sentences. The score is scaled to be a percentage for better readability. An interpretation of the BLEU score is proposed in table 2. Figure 7 presents the BLEU score for each of the models presented in table 1.

```python
metric = evaluate.load("sacrebleu")

def compute_metrics(eval_pred):
    preds, labels = eval_pred
    # HF may return a tuple; take first element
    if isinstance(preds, (tuple, list)):
      preds = preds[0]

    # If logits slipped in (B, T, V), convert to token ids safely
    if preds.ndim == 3:
      preds = np.argmax(preds, axis=-1)

    # Map ignore index to a real token id for decoding
    pad_id = tokenizer.pad_token_id
    preds  = np.where(preds != -100, preds, pad_id)
    labels = np.where(labels != -100, labels, pad_id)
```

```
    pred_seq  = tokenizer.batch_decode(preds,  skip_special_tokens=True)
    label_seq = tokenizer.batch_decode(labels, skip_special_tokens=True)

    decoded_preds, decoded_labels = postprocess_text(pred_seq, label_seq)
    result = metric.compute(predictions=decoded_preds, references=decoded_labels)
    result = {"bleu": result["score"]}
    return result
```

Listing 4: Python's function used to compute the BLEU metrics at each iteration.

$< 10\%$	Very poor
$10 - 20\%$	Hard to understand
$20 - 40\%$	Understandable, with minor errors
$40 - 50\%$	High quality
$> 50\%$	Close to human quality, fluent translation

Table 2: Proposed interpretation of the BLEU score.

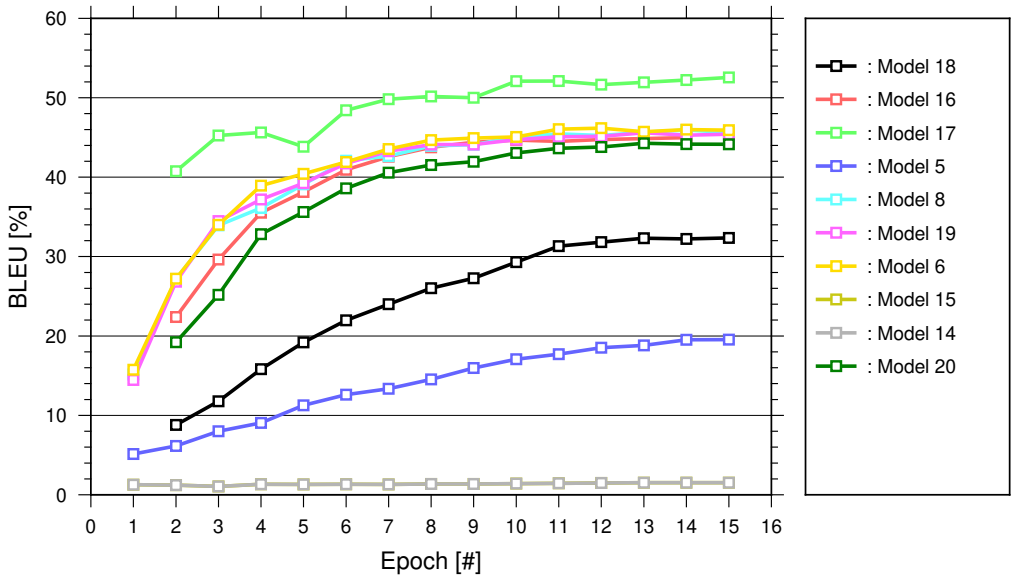

Figure 7

BLEU score of model 17 (figure 7, light green curve) reach up to 53% which is a very good score that shows that the translation could be fluent and close to a human translation. The model 17 uses the `Facebook/mbart-large-50-many-to-many-mmt` pre-trained model. Figure 2 howerver showed that this model would fail in robustly translating unknown words or sentences and was not considered as a good, comprehensive model.

Model 14 with a very small learning rate ($1e^{-6}$, see table 1 for more details) do not show any improvement of the BLEU metric with epoch (figure 7, gray curve). The metric remains lower than 10% which is considered as a very poor score. Figure 6 indeed show no significant improvement of the training and evaluation losses with training time for this model. This model is not learning well enough. Similarly, model 5 with a larger (but still very small : $2e^{-5}$) learning rate show better learning with iteration with an increasing of the BLEU score up to 20% and smaller training and evaluation losses. A BLEU score of 20% could be considered as a poor and hard-to-understant translation.

Many models show a BLEU score tending to 45% and could be classified, regarding this criteria, as a

high-quality translation model. Regarding the BLEU and training/evaluation loss metrics (figures 7 and 5), model 6 will be used for the second and last step of the evaluation process by humans.

Human evaluation of the quality of the translation

A selection of dozens of words and dozens of sentences (from three to ten words per sentence) in French and their corresponding translation in Alsatian have been compiled and distributed to some Alsatian-speaking persons. Some words may have two to three possible Alsatian translation. Exemples of words and sentences are presented in table 3. The full form can be viewed in this link. The participants where asked to rate the quality of the translation with a score as described in table 4.

parler →	Wrkung
manger →	Schlackenkisser
acheter →	kàuifa
Le chien →	S Hund
Un homme →	Ein Mànn
On travaille. →	's schàfft.
Est-ce que ça va ? →	Geht's ?
Veux-tu manger ? →	Wann dü assa ?
Je cuisine de la choucroute. →	Ech koch Sürkrütt.
Les Meyer ont une belle maison. →	D Meyer han a scheen Hüss.

Table 3: Sample exmaples of words and sentences distributed to test participants.

0	It is not a good translation.
1	It is not a good translation but some words are fine.
2	It is understandable.
3	It is a good translation.

Table 4: Proposed score to rate the quality of a translation by the test participants.

Figure 8 presents the results of the scoring from four persons. It shows that only 32% of the translations were considered as good translations while 54% of the set could be considered as understandable. One full quarter of the dataset are not understandable.

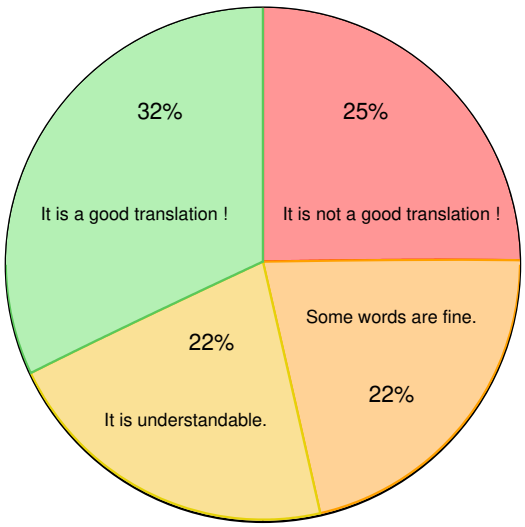

Figure 8: Summary of the scores of the rating of 125 French-to-Alsatian entries by four participants.

Participants said that it was difficult to read the Alsatian words and sentences because this language is a spoken language and is rarely seen written. However, people said that speaking the words out loud did help making the rating. In order to anticipate this issue, people were asked not to focus on grammatical and lexical misspelling but on the general meaning of the translation and whether it was understandable.

Conclusion

This project presents the work undertaken to generate a model for a translation task from French to Alsatian languages. Results show that the models are always over-fitting the data, showing that the compiled dataset is not sufficient to fine-tuned a model to get an accurate and robust translation for words and sentences that are not well represented in the dataset. Indeed, only 54% of the proposed translations to some test participants were considered as a good translation while more than 25% where not understandable. Improving the dataset by adding more data, inserting labels/contexts to each entry or manualy checking the entries could improve the fine-tuning that would lead to better accuracy of the translations.

Because Alsatian is a spoken and non-standard language with significant regional variation across Alsace, the development of a robust and accurate models was not expected. The objective of this project was to explore and apply the concepts learned during the course on a challenging application. I wanted to better understand the fine-tuning process and the importance of having a good understanding of both the data and the model parameters in order to build confidence in the resulting model.

Acknowledgments

This work benefited from the GPU computational resources of Google Colab (`https://colab.research.google.com`) and the experiment tracking and visualization tools provided by Weights & Biases (`https://wandb.ai`).

References

[1] Site web en Alsacien. http://www.alsa-immer.eu, . Accessed: 2025-11-03.

[2] Alsatext. http://www.alsatext.eu, . Accessed: 2025-11-03.

[3] D. Bernhard. Lexique multilingue alsacien - français - allemand relié aux synsets de babelnet. 2021. URL https://nakala.fr/10.34847/nkl.3f9b2i11.

[4] Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., and Liu P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[5] NLLB Team et al. No Language Left Behind: Scaling Human-Centered Machine Translation. *Arxiv*, 2022. URL https://huggingface.co/facebook/nllb-200-distilled-600M.

[6] Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. Multilingual translation with extensible multilingual pretraining and finetuning. 2020.