Project 2: Team Pleutre

Kapps Augustin	282377	augustin.kapps@epfl.ch
Giordano Lucas	284219	lucas.giordano@epfl.ch
Gruaz Lucas	287512	lucas.gruaz@epfl.ch

Hosting Lab	**MLO**
Martin Jaggi	martin.jaggi@epfl.ch
Bojana Rankovic	bojana.rankovic@epfl.ch

***Abstract*—The main goal of this project is to build a machine learning pipeline capable of grading french handwritten essays as usual professors do. This pipeline could save them at lot of effort regarding this time-consuming task. To do so, we have access to a dataset of essays graded by competent professors.**

I. Introduction

We are given scans of the handwritten essays associated with students and grades. The latter consist of several subcategories: *comprehensibility, essay task, grammar, logic, etc.* Each of them is constrained to lie between 0 and 9. Nevertheless, one should note that the data was initially stored as pictures (scans) of the essays. Previous research was conducted to map these scans into strings. An optical character recognition (OCR) process was used. It constitutes the first step of the pipeline. The second is to find a suitable text embedding that can then be fed to a regression model. We will first give details about BERT[1], a standard procedure for this task, to the reader. Then we will study how to use it for our special task (layer significance, fine-tuning,etc.). Finally, a regression will be trained to predict grades. This complex pipeline is prone to a lot of errors and we will also spend time studying them. Note that we are working in collaboration with the *MLO* lab which as already implemented a usable OCR. We were required to study the transformers as well as to perform some background tasks (data cleaning, labeling).

II. Optical character recognition (OCR)

The OCR transforms images of text into proper strings. Although we did not contribute to the OCR, we will still briefly explain its mechanism. It is composed of a CNN followed by RNN model. The first network is responsible for extracting significant features while the next keeps track of the read characters. For instance, having already decoded the characters 'hell', this sequence is likely to be followed by 'o'. In related works, spellings mistakes are usually corrected using some post processing tools. However, it is not an option here since those very mistakes may simply be part of original text (non-french writters) making this task even more complicated. More details can be found in *'IEEE___ocr_paper'*.

III. The transformers

As stated above, we used transformers to obtain vector representations of the essays. We decided to mainly focus on CamemBERT[2] and MT5[3]. Both are based a similar transformer called BERT. In this section, we start by providing insights about BERT. Then specificities of CamemBERT and MT5 will be presented.

A. What is BERT ?

The Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained model NLP developed by Google that can be used to represent texts as vectors. It was trained to reconstruct *masked words* of some text. Using Google's massive resources, they were able to come up with a powerful model for this unsupervised learning task.

A tokenization process is applied to each word, including special tokens. Then, in the first layer of the network, every single token is mapped to a 768 vector. We have:

- the token [CLS] is placed at the beginning of every text input. It flows through the network (see figure 1) and has special meaning. The model has been trained to encode everything it needs for the classification step into that single 768-value embedding vector.
- multiple [MASK] tokens are used to hide original tokens, randomly chosen. They also flow through the network and BERT is trained to reconstruct the hidden tokens.
- the [SEP] token separates sentences in the input.
- some [PAD] tokens, the neural network composing BERT needs a fixed-length input but texts can differ in their length. BERT standardises the lengths using these tokens.

The purpose of this document is not to give details related to BERT, so we will stop here. The key aspect to remember is that BERT is trained to reconstruct missing tokens and that lots of information is encoded inside the [CLS] token.

B. From BERT to CamemBERT

English texts were used to train BERT. Therefore, we need an adapted model that is able to handle our french dataset. Luckily, we have the CamemBERT at our disposal. It is a derivation of the roBERT model exclusively trained on french texts. The main difference with BERT is the tokenization part since CamemBERT uses different names for the tokens but the API makes it easier to work with.

C. MT5

MT5 is a rather new (22 Oct 2020 on arXiv) massively multilingual pre-trained text-to-text transformer developed by Google. The latter got trained using 101 different languages, including French and as the same structure as BERT.

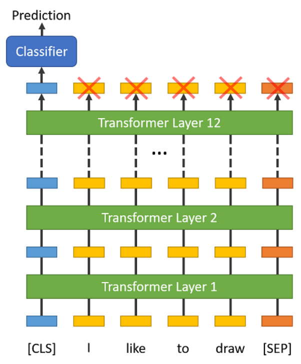

Fig. 1: BERT architecture

IV. Layers significance

The transformer encodes the information contained in a given text in the [CLS] token that flows through the layers, now the question is: At which layer (of a given transformer) the information contained in the token is the most relevant for our application. A transformer is composed of a set of layers (generally 13), each of which is a representation of the original text input. Using the concatenation of all of them as the input of the regression would yield good results but it is computationally expensive (each layer has a length of 768) thus we want to restrict ourselves to one of them. To answer this question, we set up the following experiment.

A. *The experiment*

A dataset containing all the [CLS] encoding (at all layers) of 100 manually labelled essays is generated. Therefore, no error can be attributed to the OCR. Grades associated with each essay are also included. Then, we considered all combinations of (layer, grading metric) since the most relevant layer may vary from one metric to another. For each pair, a regression of the layer on the grading metric is fitted. Its test loss (MSE) is saved in a dictionary (with keys: layer ID and metric name).
We run the experiment twice: the first model was a NN of 3 hidden layers of 100 nodes with Relu activation. Secondly, we used a simpler model consisting of single linear layer (OLS). Both of them are optimised using ADAM through a 5-fold cross validation (allows to compute 95% confidence intervals). Additionally, an Early stopping callback based on the test loss with 16 epochs patience is set. Note that the goal of this task is not to obtain the perfect fit on the dataset but to be able to compare all the layers and grading metric. Finally, we can repeat this process with different transformers to compare their relative performance.

B. *Results of the experiment*

Results are summarized in figure 3. If the ols combined with mt5 is excluded, we can deduce that the last layer (12) carries the most significant pieces of information for our regression task. Indeed, it yields the lowest test error. This can observed in deeper details in figure 4 where the test error obtained on each layer while predicting metric 'grammar' is plotted. The error drops at layer 12 but not significantly (overlapping CI). Moreover, we also notice in figure 3 that the comprehensibility score is the easiest to predict, while grammar is the most difficult one. We notice that there isn't much difference between the 12 layers with camemBERT and three hidden layers, while there is with the other setups. We also see that camemBERT seems to be the most powerful model. Using again 'grammar', a comparison between camemBERT and MT5 is built in figure 5 which supports our claim. Note also that we clipped loss values higher than 10 to avoid scaling issues in the plot and to ease the comparison.

C. *Discussion*

In the experiments, we used basic versions of the chosen transformers namely: *camembert-base* and *mt5-base* but there exists larger and,therefore, more powerful extensions. We could cite *camemBERT-large*, MT5-XXL, etc.. They were similarly trained but with larger datasets. Thus we might consider using these in the future. Note also that we experimented only using 100 data points, we were limited because we essentially used manually labeled data to avoid introducing error from the OCR. Re-running this experiment with more data should be considered since it will yield more accurate results and smaller confidence intervals.

D. *Bag of words*

As an extra step of the study, we wanted to have a deeper understanding of what was impacting the losses. To do so we used a bag of words naive approach because it is more interpretable than BERT models. We looped overall grading metrics, at each iteration we fitted a linear regression (also interpretable) using a bag of words representation of the essay as the features. Then we printed the words corresponding to the top 5 highest (absolute sense) coefficients to see which words impact the predictions. We did so on all the possible grading metrics. For example, for the 'spelling' metric we obtained that the word 'aéroport' was one of the most determining with a positive coefficient. Which makes sense since it is a word that is hard to spell. However, we also found some inconsistent result like the word 'accostera' linked to a negative coefficient despite the fact that it is well spelled. This might be due to the small size of the training set (80 samples).Moreover, in figure 2we see that the bag of word (with linear model) approach performs quite well, especially to predict the logic metric. We see that the loss is even better than using BERT models. However, this method works well on the 100 manually labeled data but it is not scalable to a bigger dataset.

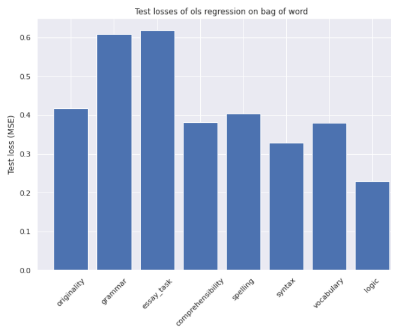

Fig. 2: Bag of words results

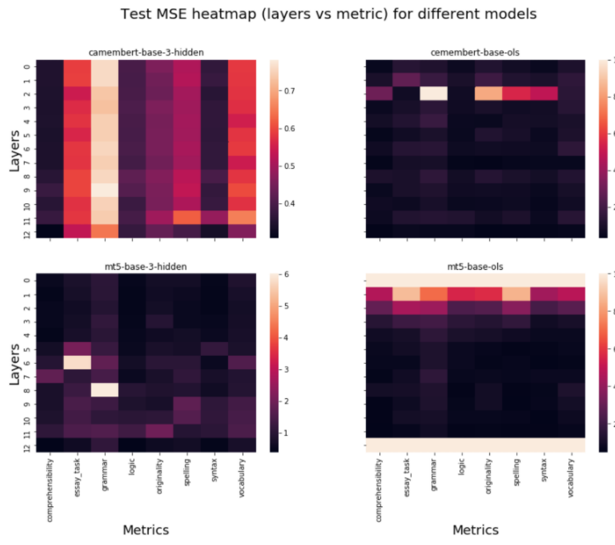

Fig. 3: Layer significance heat-map

V. Fine tuning

As explained earlier, the considered transformers are available as pre-trained architectures. However, it is still possible to update their

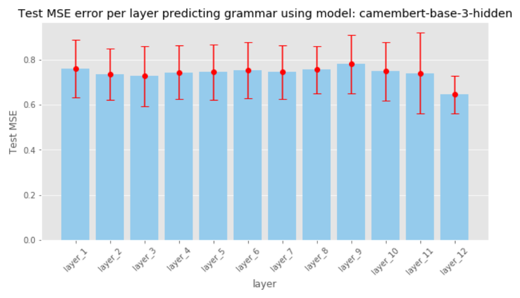

Fig. 4: Test loss camemBERT

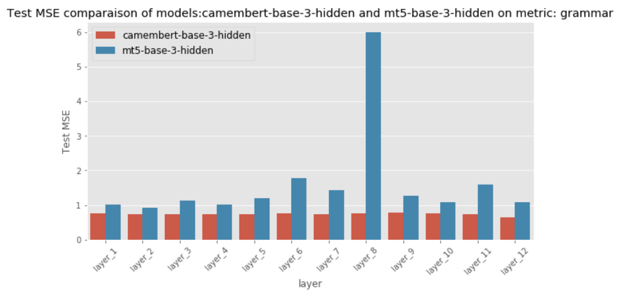

Fig. 5: Transformer loss comparison

weights and train a bit more the model. This procedure is called fine-tuning. It allows us to adapt the transformer to our dataset. To perform this task, an extra linear layer is added at the end of the pre-trained model. Then both of them are trained for the regression. Since the goal is simply to adjust the already existing weights, we only train during a few epochs (3-5) and use a small learning rate. The goal of this section is to study the effect of fine-tuning on the test loss.

A. *The experiment*

To study this impact, we will build two identical models consisting of the camemBERT transformer followed by an activation function (ReLU) and an extra linear layer. But we will train them differently, we train one of them using fine-tuning (updating camemBERT weights) for 5 epochs and then keep normal training for 120 epochs. We simply train the other one for 120 epochs without fine-tuning. We train using Adam optimizer with a learning rate of 8e-5, an MSE loss function, and a training set consisting of 90% of the dataset. At each epoch, we compute and save the test loss. Again, a dataset of 100 manually labeled data points is used.

B. *The results*

The results are shown in figure 6. We see that the fine-tuned model loss drops faster than the non fined tuned one. It takes 120 epochs for the non fine-tuned mode to get to the same loss level as the fine-tuned one. Moreover, we measured that on a google colab environment one fine-tuning epoch takes $\approx 3.6s$ compare to a standard epoch that takes $\approx 1.5s$. A fine-tuning epoch is only twice (roughly) as long as a normal one but it improves the loss significantly. Thus Fine-tuning might be interesting for us to save precious time but we do not observe a improvement in model expressiveness. Note that using a more complex architecture might change the results.

VI. Understanding the errors

Since the complete pipeline is quite complex ($OCR \rightarrow Transformer \rightarrow Regressor$), it might be interesting to understand

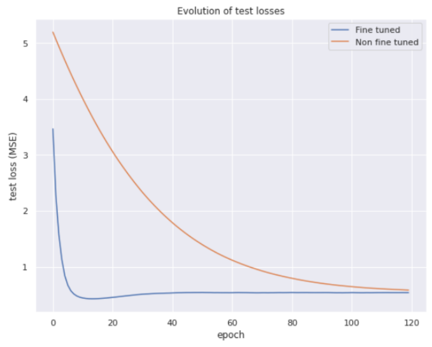

Fig. 6: Fine tuning

where errors could occur. We chose to visualize the data-points that caused large errors (MSE) in the test prediction.

A. *The experiment*

To conduct this experiment, the camemBERT-base transformer, more specifically its last layer, is used (result of section *IV*). We also arbitrarily decided to study the *grammar* metric. We had to restrict ourselves to such a choice since it is not feasible to look at errors linked to all possible layers and possible grading metrics.
The neural network trained contains 3 hidden layers of 100 nodes each and uses a ReLU activation function (same optimizer as before). Using an early stopping callback and by manually checking the learning curves (test and train) on the *tensorboard* we managed to obtain a test error of 0.3205. The trained model is a regression hence it outputs continuous values, but the grades (target variable) are positive integers thus we are mostly interested in errors that are higher than 0.5 (absolute sense). We call this value the decision threshold, it is the one that would been used to discretize the outputs of the model. The model is then used to perform a prediction on the test set (20% of the data) and to compute the residuals. For this task, we want to capture errors that are coming from anywhere in the pipeline. Thus we trained and tested the model on a dataset produced by the OCR.

B. *The results*

As a first result, the distribution of the obtained errors is showed in figure 7. The values between the two dashed lines are not proper errors since they are closer to the actual grade than to any other integer. By inspecting the essays that were misclassified ($>$ threshold decision level), we noticed that a lot of errors could possibly come from the OCR. We read some texts that were probaly far from the original version, example: ”sum vemt 1 tonul Je a M jal- ott”. Since bert-like models are usually very robust to noise we wanted to make sure that the OCR was indeed the origin of some of these errors.

C. *Impact of OCR*

To quantify the impact of the OCR on the performance of the model, we will use a set of data for which we have both the essays predicted by the OCR and the manual labels (assumed to be the ground truth). We use this dataset to train and test the model once on the OCR data and once on the manually labeled data. We compute the losses on all the possible grading metrics and then average the results. We obtained the two following test losses: 0.463 for the OCR data and 0.466 for the manual data. We see that the model performs a bit better than the manually labeled data but the difference is not significant.

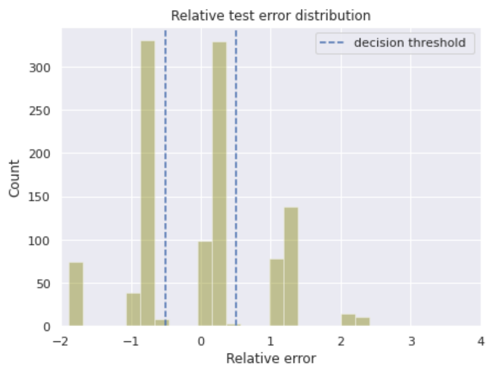

Fig. 7: Error distribution

D. Benefits of OCR

The OCR enables us to access a lot more text essays than before (recall that we only used manually labeled essays earlier). Quantifying the improvement brought by the OCR in terms of test loss is a natural task to carry out. We start with a training set consisting of the 80 manually labeled data and it is iteratively expanded using more and more of the 5660 OCR predictions. At each step, we test it using the same test set consisting of the remaining manually labeled data (avoid introducing bias due to OCR errors). Again the regression model is 3 hidden layers of 100 nodes neural network trained using Adam optimizer and an early stopping callback (patience=32) based on the test loss We save all the obtained losses.

We should pay attention to the fact this computation produces really noisy plots. Indeed, we tried to repeat this experiment with different random states. If the trend was generally the same, we observe some shift differences on the x-axis (drop starts earlier or later). In order to reduce the variance, we did several versions of this experiment and took an average to smother the curve. We did $k = 4$ of them and it yields figure 8. To reduce further the noise, we could have increased k but one should note that it already takes several hours to obtain one single such result.

We see that adding data produce a significant drop at the beginning. We believe that it simply appends due to the fact that adding more data usually yields better results in a deep learning setting. However, when adding too much data from the OCR dataset, an increase in test loss goes can be observed. One possible explanation could be that some bias is introduced by the errors of the OCR which can only be impact-full after a certain threshold. Nevertheless, the end of the plot lets us think that with a bit more data, the network might be able to overcome the induced bias.

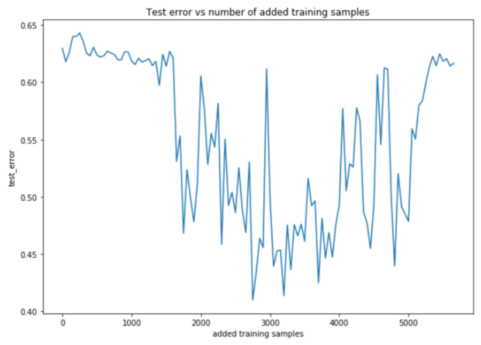

Fig. 8: Benefit of OCR

E. Progression of the OCR

While we were working on the transformer models, our colleagues at MLO were constantly improving the OCR predictions. We thought it would be interesting to compare new predictions to old ones. We could then decide if the OCR is worth improving further. The set-up is similar to previous experiments only the data differs. We have 2 sets of OCR prediction for 4000 essays and the same test set for both. WE obtained a loss (MSE) of 0.652 for the old predictions and 0.653 for the new predictions. We see that the difference is not significant. This might be due to the fact that BERT-based transformers robust to noise.

F. Errors of OCR

We have gone through the errors made by the OCR pipeline trying to get an idea of what kind of characters causes problems. It turns out that most of the essays are fully correctly transcribed, but some have a lot of mistake, probably because of an unexpected handwriting. Most of them were made between similar letters, for example an o is transcribed to an a, or v to u. These issues are understandable. Even for human it is sometimes difficult if not impossible to distinguish between o and a. Also the OCR often doesn't recognize points at the end of a sentence. For this and the other errors that are less natural (e.g once *"à la bordeaux. Pourrais vous"* was transcribed to *"ville. Je vais aup. Jssi so"*) it may be beneficial to train the model on more essays.

VII. Conclusion

In this study, we have seen how powerful BERT transformers are for the regression task. We quantified the importance of their layers in different contexts. We also looked at the relations between the OCR and the regression task, what were the errors possibly induced by the OCR. Moreover, it was seen that fine-tuning the transformer might be worth doing. As a background task, we doubled the size of the labeled dataset, which will allow later studies to be more accurate and decrease the character rate error of future OCR.

A. Personal appreciation

We have loved to work on this interesting project. Studying BERT transformers and attention learning techniques was particularly exciting for us. Indeed, that's the first real NLP application in which we have been working on. In this project, the major difficulty resided in studying how BERT models work as well as learning *PyTorch-lightning/PyTorch*.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. Camembert: a tasty french language model. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[3] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer, 2020.

VIII. Feedback(from assistants)

A. Feedback 1

- + The pipeline is described nicely.
- + A large set of studies analyzing different aspects of the pipeline are done.
- + Good use of figures to illustrate the findings.
- - Some Citations like the ones to the papers containing MT5 and RoBERTa are missing.
- - Font size is not proper for many of the figures.Minor
- - "BERT[1], a standard procedure for this task" should be "BERT[1], a standard model used for this task"
- - "More details can be found in 'IEEE_ocr_paper'."- I am unable to understand what this text says.
- - It is a derivation of the roBERT model exclusively trained on french texts". I believe it is RoBERTa instead of roBERT.

Score : 90%

B. Feedback 2

The report is well written. The introduction includes the explanation of the task at hand alongside the explanations to external papers and models used to get started tacking the given assignment.

The reasoning for using different models was explained and the differences in results interpreted thoroughly.

The students conducted an interpretability study using different layers of given models to see the outcome of the training on the results for different grading criteria. Moreover, in order to understand what impacts the loss further, they also tried bag of words representation of the essays and achieved interesting results in which they observed that certain words affect certain grading criteria more than other (hard spelling words for spelling criteria).

The impact of the quality of OCR was also explored, comparing the results with different versions of OCR transcriptions. There is an OCR error analysis about the characters that get mistakenly transcribed, which can be helpful to improve the OCR model.

Overall, the students displayed a structured approach to solving the given task, managing the results and providing interpretable insights.

The report could, however, be improved by eliminating small syntax mistakes and structuring the experiments/result section differently so that there is a better overview of the models and the differences in results they bring, maybe by including a table comparing the best results for all the models tried, together with the effects of the different OCR transcriptions used.

Score : 90%

Overall : a very good project for the scope of this course.