NLP Project – Automatic Essay Scoring

Hippolyte Gisserot [*1] **Guillaume Kunsch** [*1]

Abstract

In this paper, we focus on using NLP to automatically grade essays. We build on a Kaggle challenge which provided data and grades from US-located schools. We frame the problem as a regression and we try various pre-trained embeddings - GloVe and Word2Vec - as well as models - Conv1D, Transformers trained from scratch and Transfer Learning from pre-trained Transfomers architecture - on the data. For the best model, we manage to get a MAE of 10% which is promising as a first approach but not enough to be used as in a real-world educational purpose.

1. Introduction

In France, some curriculum rely almost exclusively on multiple-choice standardized test that can be evaluated automatically. Not all studies field are prone to such a grading method though. One of the key roadblocks to advancing school-based curricula focused on critical thinking and analytical skills is the expense associated with scoring tests to measure those abilities. Because of those costs, standardized examinations have increasingly been limited to using tests that deny the opportunities to challenge students with more sophisticated measures of ability.

As well as the obvious social interrogation that computer-graded students would faced, this task is challenging because of the variety and specificity of words and meaning attached to them in each field. Sometimes it's not even about the statistical occurence of words - as in traditional NLP - but rather the profound meaning of manipulated concepts - think of mathematical abstractions and combinations of formula.

In this paper, we will only focus on long-form constructed response , i.e essays, of about 300 hundred words. The more challenging tasks of evaluation for short-form constructed response and symbolic mathematical/logic reasoning are left aside.

[*]Equal contribution [1]IASD 2022-2023, PSL University, Paris.

2. Data

2.1. Dataset presentation

The dataset of interest is composed of 12,978 essays written by students, divided into 8 subsets corresponding to subjects of various levels of difficulty. Each essay set is graded on a specific integer score range (Table 1).

Set	#Essays	#Grades	Grade range
1	1783	1	2-12
2*	1800	2	1-6, 1-4
3	1726	1	0-3
4	1772	1	0-3
5	1805	1	0-4
6	1800	1	0-4
7	1569	1	0-30
8	723	1	0-60

Table 1. Data summary

*Essay set 2 presents two distinct grades, respectively for writing applications and for language conventions.

2.2. Data preprocessing

The data preprocessing phase is essential for this task. It includes: putting the grades on the same scale, converting the essays into continuous and same size matrix representations, and finally divide the obtained dataset into training, validation and test subsets.

Grade scaling

As it is shown in the previous section, each essay set is graded on a specific score range, thus making it inconvenient for machine learning purposes. For harmonization, we simply scale all the grades between 0 and 1. For essay set #2, we sum the two grades before scaling the total score between 0 and 1. Having all the grades on the same score range then makes it possible to consider the task as a single-output regression problem.

Essay embedding

Maybe the most critical task in this data preprocessing phase is to find a proper way to convert the raw essays (string type) to a ML-interpretable format. For that matter, we rely on

two different pretrained word-embedding models:

- Word2Vec (Mikolov et al., 2013), which is built on a neural architecture that maximizes the probability of observing the training data given the embeddings. The model was pretrained on a Google News corpus and generates embeddings of size 300. For computational efficiency purposes, we reduced the embedding size to 50 using PCA.
- GloVe (Pennington et al., 2014), which is based on the co-occurence matrix words, that counts how often pairs of words co-occur in a large corpus (in this case a combination of Wikipedia and Gigaword news corpus). The model already generates embeddings of the desired size, 50.

More precisely, the translation from raw to embedded essays is handled in the following way:

- First, the essay is tokenized, i.e., converted into a sequence of words and punctuation.
- Then, each word of the sequence is translated into a vector and concatenated at the end of the embedding matrix. If the word cannot be found in the dictionary (e.g., wrong spelling), it is simply translated into a vector full of 0.
- The last step is repeated for the two embedding models.

Essay padding

Then, we need to pad the essay matrices so that they are all the same size and can serve as input to our models. To do so, we simply concatenate 0-vectors at the end of each essay until it reaches the dimension of the longer essay present in the dataset, i.e., 1266.

Data splitting

Finally, we split the processed dataset into training, validation and test subsets (respectively 80%, 10% and 10% of the total number of samples).

3. Model experimentation

In this section, we present the results obtained on the essay scoring task using different neural architectures.

Let us denote $\mathcal{D} = \{(x_i, x'_i, y_i)\}_{i=1}^N$ the dataset, where $\{x_i\}_{i=1}^N$ are the embedded essays (size 50×1266), $\{x'_i\}_{i=1}^N$ the essay sets, $\{y_i\}_{i=1}^N$ the grades to be predicted, and $\mathcal{P}$ the distribution from which $\mathcal{D}$ is drawn. Denoting f_θ the neural network (which ends with a sigmoid activation function) and using the MSE as a loss function, the optimization problem can be summarized in the following way:

$$\begin{aligned} &\arg\min_\theta \mathbb{E}_{X,X',Y\sim\mathcal{P}}\left[(f_\theta(X,X')-Y)^2\right] \\ &\approx \arg\min_\theta \frac{1}{N}\sum_{i=1}^N (f_\theta(x_i,x'_i)-y_i)^2 \end{aligned} \tag{1}$$

Gradient descent is then conducted using the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate equal to 10^{-3} (if no indication otherwise), 100 epochs and a batch size of 64. An early stopping feature with a patience of 10 iterations is also added to prevent overfitting, using MAE (mean absolute error) on the validation set.

3.1. Convolutional model

1D convolutional networks have been traditionally used in computer vision tasks such as image recognition. However, they have also been used in NLP as well (Kim, 2014). One of the key advantages of using Conv1D networks in NLP is that they can effectively model the sequential nature of natural language data. In particular, they can learn local patterns and relationships between adjacent words, phrases, or sentences, which can capture important linguistic features and help improve task accuracy. Moreover, the use of such networks can reduce the number of parameters needed, leading to faster training and lower computational costs.

Our first approach relies on a 1D convolutional model, where the essay length is the dimension covered by the kernels and the embedding size is treated as the number of input channels.

The implemented network architecture can be described as follows:

- Input: $x \in \mathbb{R}^{50\times 1266}$, $x' \in \mathbb{R}$ (resp. embedded essay and essay set).
- Essay set encoding: x' is encoded to a one-hot vector and then passed through a linear layer that maps it to a vector of the same length as x. Next, the resulting embedding is added to each channel of x.

$$\begin{aligned} x_1 = \text{ReLU}(&\text{Linear}_{8\to 1266}(\text{OneHot}_8(x')) + x \\ &(\in \mathbb{R}^{50\times 1266}) \end{aligned} \tag{2}$$

- Sequential convolutions: a series of d sequential 1D convolutions is then applied to x_1 (with number of channels $= 50$, kernel size $= 5$, stride $= 1$, same padding, skip connections, batch normalization and ReLU activation).

$$\begin{aligned} &\forall 1 \le k \le d, \\ x_{k+1} = \text{ReLU}(&\text{BNorm}(\text{Conv1D}_{50}(x_k))) + x_k \\ &(\in \mathbb{R}^{50\times 1266}) \end{aligned} \tag{3}$$

- Last convolution: x_{d+1} is then passed through a last convolutional layer, but this time with only 1 channel.

$$x_{d+2} = \text{ReLU}(\text{Conv1D}_1(x_{d+1})) \quad (\in \mathbb{R}^{1266}) \tag{4}$$

- Full connection: finally, x_{d+2} is passed through a linear layer and a sigmoid activation function for grade prediction.

$$\hat{y} = \text{Sigmoid}(\text{Linear}_{1266\to 1}(x_{d+2})) \quad (\in \mathbb{R}) \tag{5}$$

We test the performance of the model for 3 different depth parameters: $d = 0, 5, 10$ (Figures 2, 3). Unsurprisingly, training performance improves as d increases (let us note however that $d = 10$ shows worse training performance than $d = 5$, which can be due to too much model complexity and therefore inefficient training in the former case). However, when it comes to validation performance, we notice that the best MAE is reached for $d = 0$ (0.1058) and that the model significantly overfits when $d = 5, 10$. This can be due to several parameters such as a not large enough number of samples or a too small embedding dimension. In short, for large values of d, model complexity surpasses the complexity of the dataset.

3.2. Transformer-based model

More recently, a new neural architecture has emerged as the new state-of-the-art approach when it comes to NLP tasks: Transformers (Vaswani et al., 2017). Transformers allow for efficient processing of sequences of variable length, such as sentences and paragraphs. They are based ont wo main techniques: self-attention and positional encoding. Self-attention mechanisms allow Transformers to selectively focus on specific parts of the input sequence while also capturing long-range dependencies, which is difficult to achieve with traditional approaches like recurrent or even convolutional networks. Positional encoding enables the model to encode the relative positions of the words in the sequence, allowing the model to distinguish between the order of the words. Additionally, Transformers can process input sequences in parallel, which makes them faster and more scalable than some other traditional approaches.

Our strategy here is to use the traditional Transformer model (Figure 1) and adapt it to our specific task. The implemented network architecture can be described as follows:

- Input: $x \in \mathbb{R}^{50\times 1266}$, $x' \in \mathbb{R}$.
- Positional encoding (same as in the original paper):

$$x_1 = \text{PositionalEncoding}(x) + x \quad (\in \mathbb{R}^{50\times 1266}) \tag{6}$$

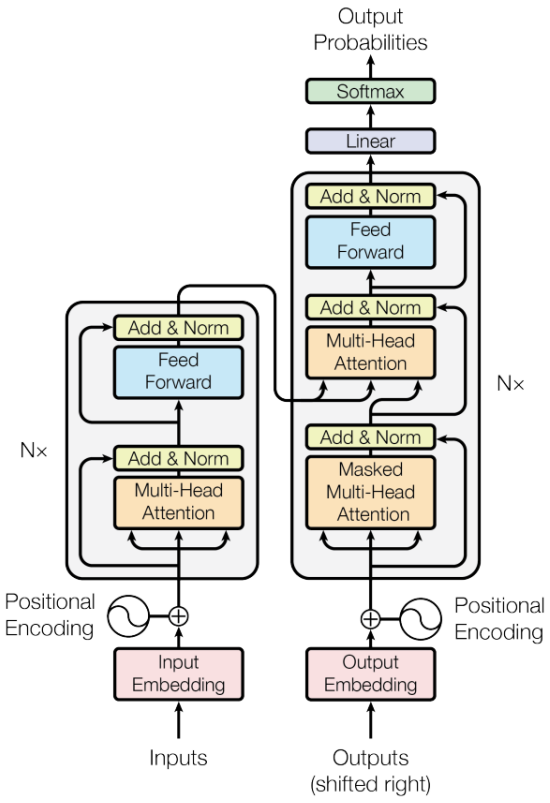

Figure 1. Transformer architecture (Vaswani et al., 2017)

- Essay set encoding (same as in the previous section):

$$x_2 = \text{EssaySetEncoding}(x') + x_1 \quad (\in \mathbb{R}^{50\times 1266}) \tag{7}$$

- Transformer encoder layers (same as in the original paper): x_2 passes through a series of n_{layers} sequential transformer encoder layers (with number of heads $= 5$, feedforward dimension $= 2048$ and dropout $= 0.1$).

$$\forall 1 \le k \le n_{layers} \quad x_{k+2} = \text{TransformerEncoderLayer}(x_{k+1}) \quad (\in \mathbb{R}^{50\times 1266}) \tag{8}$$

- Full connection: $x_{n_{layers}+2}$ is finally flattened and passed through a linear layer with sigmoid activation:

$$\hat{y} = \text{Sigmoid}(\text{Linear}_{50\times 1266\to 1}(\text{Flatten}(x_{n_{layers}+2}))) \quad (\in \mathbb{R}) \tag{9}$$

As expected, we had a lot of trouble training this model for $n_{layers} > 1$ due to the important number of parameters

to train, and this even using Google Colab GPUs. Unsurprisingly again, training this model with only $n_{layers} = 1$ is not sufficient to take full advantage of the Transformer architecture and therefore leads to disappointing training and validation performance (Figures 2, 3).

3.3. Transfer learning-based model

The previous section illustrates how difficult it is to train a full Transformer model from scratch when not having access to appropriate computational power. In this section, our goal is to use transfer learning (Pan & Yang, 2010) relying on a pretrained Transformer model and only retraining the first and last layers in order to adapt it to our specific needs.

Why use transfer learning? Firstly, it allows for efficient use of limited training data. By pre-training a model on large amounts of data, it can learn general linguistic features that can be transferred to downstream NLP tasks with much smaller training sets, allowing for more efficient use of limited resources. Transfer learning has also made it possible to leverage the computational power of large-scale computing resources. Pre-training large language models can be computationally intensive and time-consuming, but once trained, these models can be fine-tuned on specific tasks with relatively little additional computational cost.

Why rely on Swin Transformers instead of the traditional version? Swin Transformers are a recent development in the field of NLP that offer several advantages over traditional Transformer models. One of the main advantages of Swin Transformers is their ability to process large inputs with fewer parameters, making them more efficient and scalable. This is achieved through a hierarchical architecture that divides the input sequence into smaller parts and processes them independently, allowing for more parallelization and reducing the computational cost. Additionally, Swin Transformers incorporate a window-based self-attention mechanism, which allows the model to selectively attend to nearby tokens, further improving efficiency without sacrificing accuracy.

Synthetically, the implemented network architecture can be described in the following way:

- Input: $x \in \mathbb{R}^{50\times 1266},\ x' \in \mathbb{R}$.
- Essay set encoding (as in the two previous sections):

$$x_1 = \text{EssaySetEncoding}(x') + x \quad (\in \mathbb{R}^{50\times 1266}) \tag{10}$$

- Retrained first layer: x_1 is passed through a 1D convolutional layers with number of channels $= 96$, kernel size $=$ stride $= 16$ and ReLU activation in order to match the input size of the first pretrained layer.

$$x_2 = \text{ReLU}(\text{Conv1D}(x_1)) \quad (\in \mathbb{R}^{96\times 79}) \tag{11}$$

- Truncated Swin Transformer model (without first and last layer, fixed weights) as described in the original paper (Liu et al., 2021):

$$x_3 = \text{TruncatedSwinTransformerModel}(x_2) \quad (\in \mathbb{R}^{768}) \tag{12}$$

- Last layer: x_3 is passed through a last linear layer with sigmoid activation.

$$\hat{y} = \text{Sigmoid}(\text{Linear}_{768\to 1}(x_3)) \quad (\in \mathbb{R}) \tag{13}$$

As evidenced by the training and validation performance graphs (Figures 2, 3), this model overfits quite significantly, probably for the same reasons as those presented in the convolutional model section (not enough data samples, too small embedding dimension). However, the training curve highlights how much transfer learning can be useful in NLP tasks. Indeed, the training loss keeps decreasing through the learning process while only a small portion of the weights is being retrained. The problem here seems to come more from the data than from the approach itself.

3.4. Final results

After all the experiments conducted, we realise that the best validation performance is reached with the ConvNet1D ($d = 0$), which is by far the simplest in terms of architecture and number of parameters. This counter-intuitive result at first sight can in reality be easily interpreted: the task is not complex enough and the dataset not precise enough to take full advantage of state-of-the-art architectures such as Transformers.

Finally, the selected ConvNet1D model achieves a test MAE of 0.1138, which is quite satisfactory considering that random predictions (before training) usually achieve a score between 0.2 and 0.25.

As an illustration of the automatic scoring process, we randomly select an essay from the dataset along with the subject it was graded on and display the predicted and actual grades that were given.

Subject: More and more people use computers, but not everyone agrees that this benefits society. Those who support advances in technology believe that computers have a positive effect on people. They teach hand-eye coordination, give people the ability to learn about faraway places and people, and even allow people to talk online with other people.

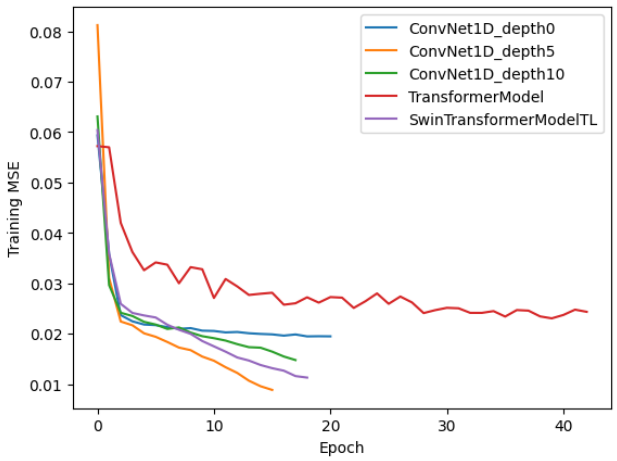

Figure 2. Training performance of the different models

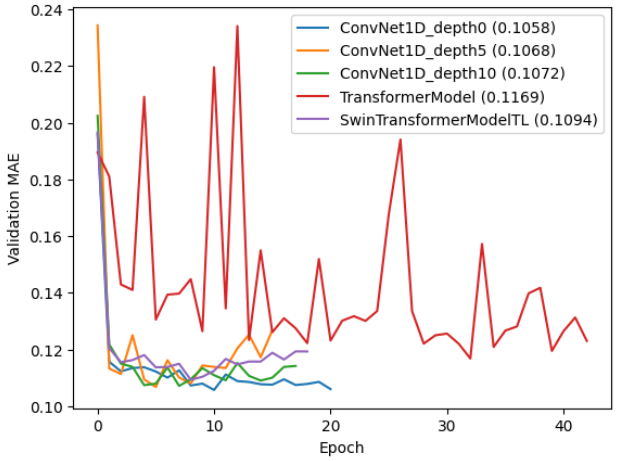

Figure 3. Validation performance of the different models (between brackets, best validation MAE)

Others have different ideas. Some experts are concerned that people are spending too much time on their computers and less time exercising, enjoying nature, and interacting with family and friends. Write a letter to your local newspaper in which you state your opinion on the effects computers have on people. Persuade the readers to agree with you.

Essay: *Dear local newspaper, I think effects computers have on people are great learning skills/affects because they give us time to chat with friends/new people, helps us learn about the globe(astronomy) and keeps us out of troble! Thing about! Dont you think so? How would you feel if your teenager is always on the phone with friends! Do you ever time to chat with your friends or buisness partner about things. Well now - there's a new way to chat the computer, theirs plenty of sites on the internet to do so: @ORGANIZATION1, @ORGANIZATION2, @CAPS1, facebook, myspace ect. Just think now while your setting up meeting with your boss on the computer, your teenager is having fun on the phone not rushing to get off cause you want to use it. How did you learn about other countrys/states outside of yours? Well I have by computer/internet, it's a new way to learn about what going on in our time! You might think your child spends a lot of time on the computer, but ask them so question about the economy, sea floor spreading or even about the @DATE1's you'll be surprise at how much he/she knows. Believe it or not the computer is much interesting then in class all day reading out of books. If your child is home on your computer or at a local library, it's better than being out with friends being fresh, or being perpressured to doing something they know isnt right. You might not know where your child is, @CAPS2 forbidde in a hospital bed because of a drive-by. Rather than your child on the computer learning, chatting or just playing games, safe and sound in your home or community place. Now I hope you have reached a point to understand and agree with me, because computers can have great effects on you or child because it gives us time to chat with friends/new people, helps us learn about the globe and believe or not keeps us out of troble. Thank you for listening.*

Actual grade: 60%

Predicted grade: 54.9%

4. Conclusion

In this paper, we researched how to use NLP for automated-grading. We showed promising results but far from usability in real-world applications. Though, the simplicity of the approach we undertook is a promising signal as for it's future potential. Some ameliorations of our work could focus on :

- Using different text pre-processing methods, as padding may for instance have a bad impact on performance for essays of significantly different lengths;
- Using other embedding types (at the sentence of character level) in order to handle grammatical/spelling correctness and out-of-vocabulary words more properly;
- Relying on specifically trained word embeddings in an end-to-end pipeline;
- Increasing the embedding dimension and number of samples to observe the superiority of more complex models.

References

Kim, Y. Convolutional neural networks for sentence classification, 2014.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359, 2010.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.