Block 1

Oscar Morris

10 Jun 2022

1 Experiment 1

1.1 Aims

The first set of training was run to establish answers to the following questions:

1. Can outliers be detected by comparing multiple models?
    - are the outlier answers all the same, or do they change depending on the model?
2. How does the distribution of machine-given marks compare with that of the human markers?
3. How do models' predictions compare with each other (similar to question 1)?
4. Do the results found in the EE hold for more repeats and training samples?

1.2 Planned Method

Three models were going to be trained on the Tin Iodide synthesis dataset (approx. 100 samples) to determine differences, these models were the highest performing in the EE:

1. BERT (at the bert-base-cased checkpoint)
2. ALBERT (at the albert-base-v2 checkpoint)
3. Longformer (at the longformer-base-4096 checkpoint)

Each model was trained for 30 epochs with a batch size of 8. Tests were done evaluating the model on unseen data, and on training data.

1.3 Results

During training, the BERT model achieved similar values as in the EE, with r^2 around 0 and MAE around 0.1.

As can be seen from Fig. 1, the BERT model is always predicting approximately the same value, this value is approximately the mean of the human given marks (0.69). This is a common issue when training regression models, since most of the time it is better to predict the mean of the training data rather than randomly (and consequently learn the task). On re-examination of the data collected in the EE, it is very likely this this was also occurring in

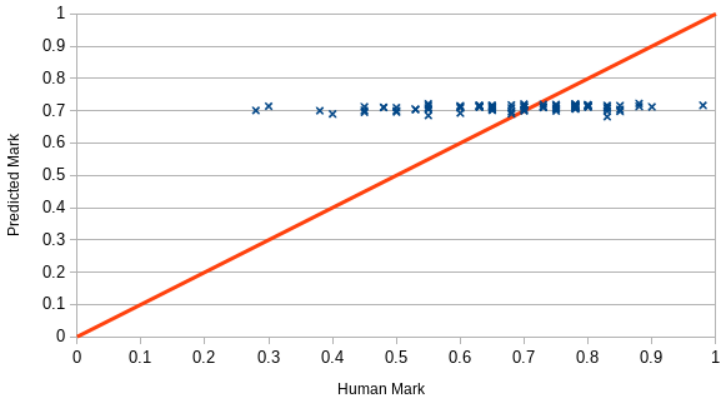

Figure 1: BERT predictions against Human markers on training data, Orange is the ideal solution

those experiments. This problem is known as "underfitting," either to the training data or to the evaluation data, in this case it appears that the model is underfitting to both splits as the distribution is very similar between seen and unseen samples. Because of this, only the BERT model was trained as the data collected in the EE showed that it was likely this was occurring for all models.

1.3.1 Cause hypotheses

On the viewing of the data collected in this experiment, a few hypotheses were created for the causes (and potential solutions) to this problem:

1. The transformer-based models were too large to effectively learn from the data
2. The distribution of human-given marks in the training data was too unbalanced, leading to underfitting.
3. The AES task cannot be effectively learned on this dataset.

1.3.2 Cause Solutions

Possible solutions to each hypothesis are:

1. Use a smaller pre-trained transformer based on BERT [1], or create and train a small LSTM or self-attention based model
2. Testing on a more balanced dataset, or adjusting the loss metric to account for unbalanced data
3. Testing on a different dataset that has been proven to be effective in previous work.

Each solution has been tested on the Kaggle AES dataset [2] in further experiments.

2 Experiment 2

2.1 Aims

The second experiment aimed to determine the effectiveness of creating, and training a self-attention model from scratch.

2.2 Methods

The basic model architecture is shown in Fig. 2, with the architecture of the encoder shown in Fig. 3.

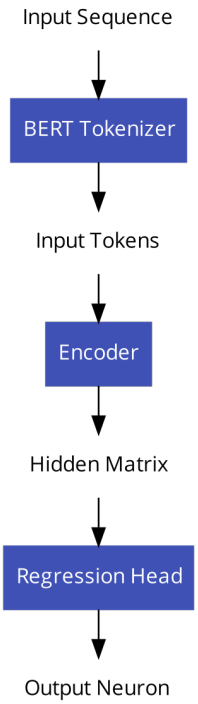

Figure 2: Basic regression transformer architecture

Each model was trained for 5 epochs, the model has a hidden LSTM size of 256 and an embedding length of 300.

The model was then pre-trained and evaluated on the Kaggle AES (Automated Essay Scoring) dataset [2] and the Kaggle SAS (Short Answer Scoring) dataset [3].

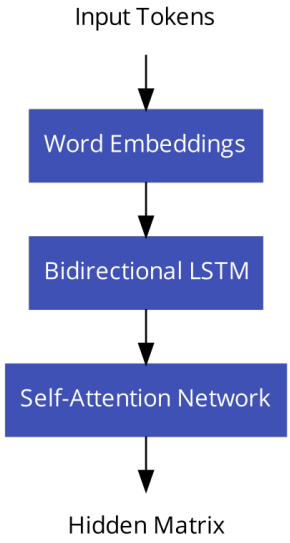

Figure 3: Encoder architecture

2.3 Results

The results for this model are very similar to the results for Experiment 1. This shows that the problem likely does not like with the model. However, due to the similar resuts to BERT, this model was used for all further Experiments due to the increase in control and data that a custom model gives.

3 Experiments 3, 4

No further progress was made in Experiment 3.

In Experiment 4 the model was adjusted slightly, replacing the regression head with a classification head, this was to test the feasibility of a classification system rather than a regression system. Previous work has found it to be ineffective when compared to regression [4]. However, because of the difficulty in creating a system that allows a regression model to learn effectively it was still attempted. The model achieved a classification accuracy of approximately 0.6, almost double what would be expected if the model was guessing randomly, although the model is still not performing well. However, with further tuning and model improvements it is possible that this score could be significantly improved.

4 Experiment 5

4.1 Aims

Determine the effectiveness of a custom loss metric combining the difference between the standard deviation of the model's predictions and the human marks, and an error metric such

as Mean Squared Error (MSE) or Root Mean Squared Error (RMSE). In later versions of the metric r^2 was added.

4.2 Methods

Metrics are combined to form a single loss function by means of a weighted sum. The importance of a metric in this weighted sum can be defined by some coefficient s:

$$L_T = s \cdot L_1 + (1 - s) \cdot L_2$$

It was found that the model was unable to optimise both metrics when they were combined with a constant coefficient. Therefore, the coefficient s was decayed throughout the training process. This significantly improved the performance of the model. The coefficient was then defined using an exponential decay function as follows:

$$s = e^{-a\frac{t}{T}}$$

where a is a coefficient determining the rate of decay, t is the current epoch and T is the total number of epochs.

This method was found to be much more effective, achieving an r^2 value of approximately 0.1, significantly higher than any previous attempt. This model was only used on the Kaggle AES dataset referenced above.

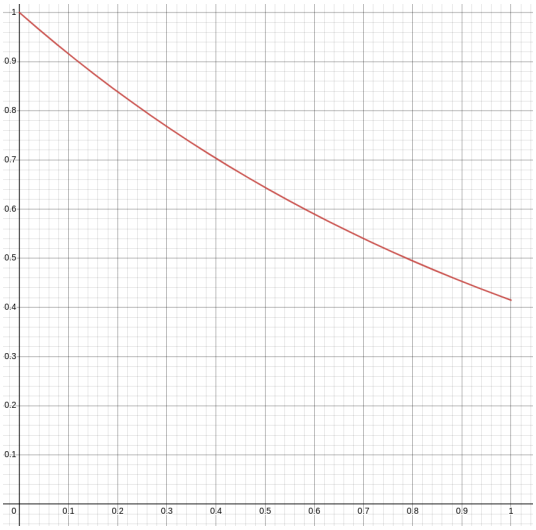

Figure 4: Graph of s against $\frac{t}{T}$, $a = 0.88$

During the training of this model, the difference between the predicted and human standard deviations remained relatively high, much higher than expected. Therefore, it was decided to redefine the decay curve to allow the model more time to focus on the achieving the correct standard devaiation. Therefore, s was defined as a piecewise function as follows:

$$s = \begin{cases} c & 0 \leq \frac{t}{T} \leq b \\ c \cdot e^{-a(\frac{t}{T} - b)} & b \leq \frac{t}{T} \leq 1 \end{cases}$$

where c is the starting value of s (usually set to 1) and b is the point at which the function begins to decay

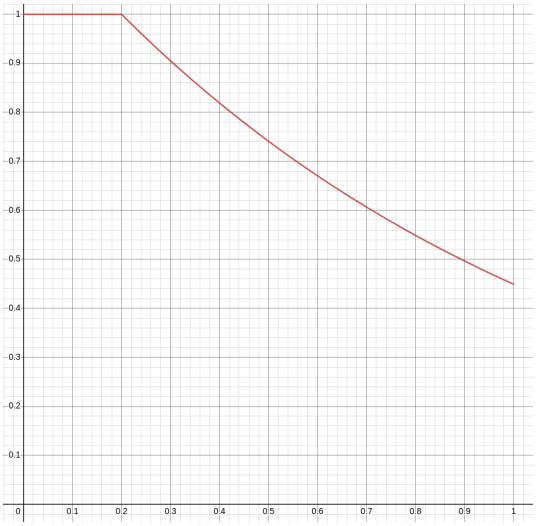

Figure 5: Graph of s against $\frac{t}{T}$, $a = 1$, $b = 0.2$, $c = 1$

The individual loss functions (after r^2 was added) are defined as follows:

$$L_1 = (\sigma_{pred} - \sigma_{true})$$

$$L_2 = \text{RMSE} - d \cdot r^2$$

where σ_{pred} is the standard deviation of the predicted values, σ_{true} is the standard deviation of the human given values, d is a constant coefficient and r^2 is the coefficient of determination.

The best performing models were trained for 200 epochs using a batch size of 64, however, no hyperparameter tuning has been executed. The model used has a hidden LSTM size of 512 and an embedding length of 128 giving approximately 70 million trainable parameters.

4.3 Pre-training Results

The model final model evaluated in this experiment beats all other attempts achieving an r^2 score approaching (and occasionally exceeding) 0.6 on both Kaggle datasets.

The model trained on the AES dataset was then further trained on the SAS dataset. It was found that the model pre-trained on the AES dataset performed worse than when the model was trained completely on the SAS dataset. However, when the model was pre-trained it could consistently achieve performance slightly lower than the better performing pure SAS models, average performance for both model has not yet been calculated.

The model trained only on the SAS dataset performed very well achieving an r^2 of approximately 0.63.

During the training it was found that this metric performs significantly better when training using a high batch size. This is likely because it makes the standard deviation and coefficient of determination more analogous to what the model would predict in a real world setting and can, therefore, better optimise the model for this task.

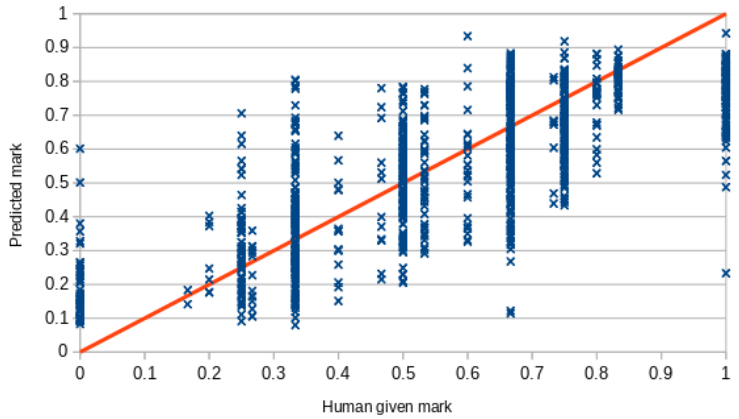

Figure 6: Predicted score against Human given score on a run achieving $r^2 \approx 0.58$

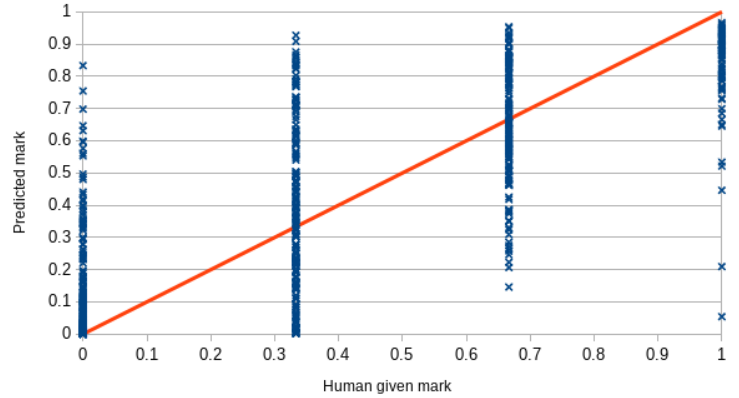

Figure 7: Predicted score against Human given score on a run trained on SAS

		human				
machine		0	1	2	3	total
	0	318	46	1	1	366
	1	47	79	31	2	159
	2	10	46	81	26	163
	3	0	9	31	40	80
	total	375	180	144	69	

Figure 8: Confusion matrix for SAS data

4.4 Fine Tuning Results

The data that was used for fine tuning after being pre-trained on either the AES or SAS datasets, there was no particular difference found between pre-training on the two datasets. The Tin Iodide, lab report dataset that was used for fine tuning contained only 100 samples, this means that the model was over-fitting to the training data significantly. Therefore, no accurate judgement can be made on the performance of the model. When evaluated on the training split, the model achieves an r^2 value approaching 0.3, whereas when evaluated on the testing split, the model achieves r^2 values of between -1 and 0, this significant difference shows severe overfitting.

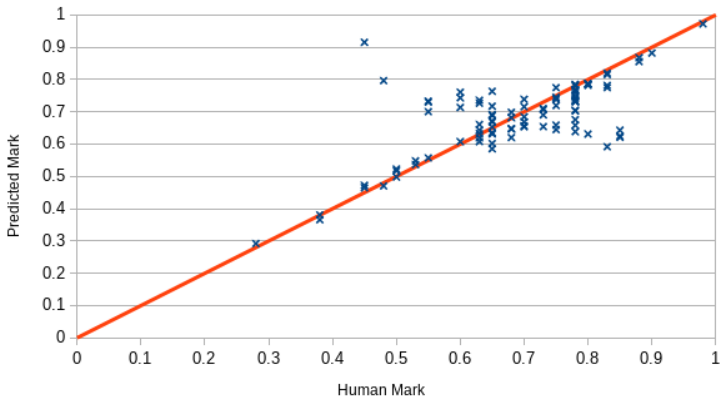

Figure 9: Prediced score against Human score on the training split of the Tin Iodide dataset, showing signs of severe overfitting

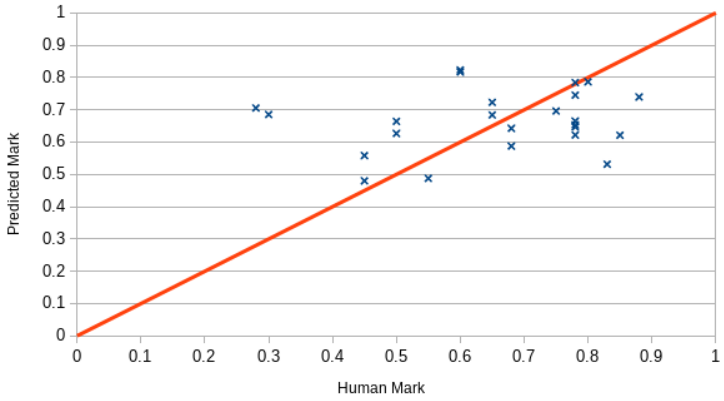

Figure 10: Predicted score against Human score on the testing split of the Tin Iodide dataset, achieving an r^2 of 0.03, showing signs of overfitting to the training split

The most likely cause of this overfitting is the lack of training data available, it is possible that another cause could be the lack of variation in the training data, as is expected with real-world data, this dataset is very unbalanced with a large number of samples achieving approximately a score of 0.7 when rated by human markers.

5 Conclusion

The models created in this block (length: 2 weeks) have shown significant improvement over previous attempts using transformer models for AES and show significant promise using the Kaggle datasets as a pre-training method to create a pre-training framework specific to AES, allowing for fewer samples to be used when fine-tuning a model for a real-world application.

The pre-training step is time-efficient as the model is able to achieve comparatively high performance in just over an hour of training (on **either** the Kaggle AES dataset **or** the Kaggle SAS dataset).

When the pre-trained models were fine-tuned to marking the Tin Iodide lab reports, the model showed signs of severe overfitting, likely due to a small number of training samples, or a lack of variety in the training samples (probably some combination of both). Another issue with this dataset is that the marks are given for the whole report, (out of which only one question is used) rather than just the question for which marks are being predicted. This does change the task slightly, from marking an answer, to predicting a score for a whole paper, based on the answer two one question. Because of this, it is very likely that the model will perform significantly better with better data.

These results show great promise when it is considered that the hyperparameters presented here have not been tuned but only roughly estimated. Therefore, it is reasonable to suggest that with hyperparameter tuning the models could perform significantly better than presented here.

References

[1] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-Read Students Learn Better: On the Importance of Pre-training Compact Models," Sept. 2019.

[2] "The Hewlett Foundation: Automated Essay Scoring." https://kaggle.com/competitions/asap-aes.

[3] "The Hewlett Foundation: Short Answer Scoring." https://kaggle.com/competitions/asap-sas.

[4] S. Johan Berggren, T. Rama, and L. Øvrelid, "Regression or Classification? Automated Essay Scoring for Norwegian," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, (Florence, Italy), pp. 92–102, Association for Computational Linguistics, 2019.