INF582 Project

Human Written Text vs. AI Generated Text

Hugo Bouigeon - Thomas Li - Jules Potel

March 2023

Table of contents

Introduction

AI generative text is the process of creating natural language texts using artificial intelligence models. These models are trained on large amounts of text data and learn to generate coherent and fluent texts based on some input, such as a prompt, a keyword, or a context. AI generative text has many applications in various domains, such as content creation, summarization, dialogue systems, and natural language understanding.

Lately, with the unprecedented advance of Deep Learning, some models such as ChatGPT and GPT-4 are being taught to understand queries and to generate text accordingly that is almost indistinguishable from a human-written one. However, this also raises ethical and social concerns about the potential misuse and abuse of such technologies. For example, AI-generated texts could be used to spread misinformation, manipulate opinions, impersonate others or create fake content. Therefore, it is important to develop and implement responsible and trustworthy practices for using and AI generative text systems. A great way to do that would be to have a tool to help us determine if a given text was written by a human or generated by an AI.

This is precisely the object of this Kaggle challenge. To do so, we have a labeled training dataset of 4000 corpus each containing between 30 and 500 words. And for the final evaluation we have an unlabeled test dataset of 4000 corpus.

Chapter 1

Transformer classifier model

1.1 BERT architecture

In class, we saw how good transformer models are at modelling language. This is why after trying out several different models (such as Logistic Regression, Decision Tree, Random Forest and XGBoost) on the TF-IDF features and getting a best baseline of 69%, we wanted to try out a transformer based model. In the lab 7 on transformer models, we saw how to train a transformer decoder architecture with the upper triangular masking and how good it was at language modeling ie. generating new text. However, even though we can change the classifier head for a classification task, the predictions were only based on the left context. This is why, inspired by the Pre-training of Deep Bidirectional Transformers for Language Understanding paper, we decided to go for a bidirectional transformer encoder architecture which is BERT available on HuggingFace.

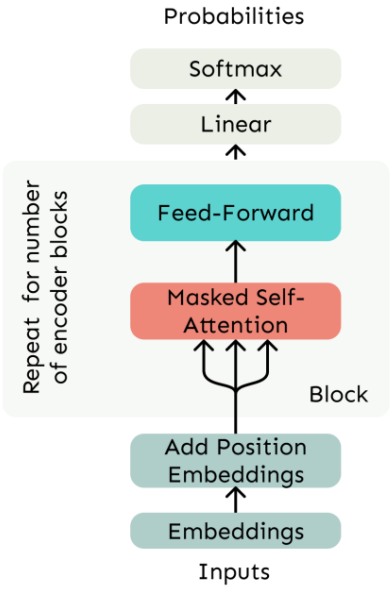

Figure 1.1: BERT architecture (there is no mask in the multi-head self attention layer)

1.2 BERT training

Obviously, training a big model like BERT from scratch (12 layers of transformer blocks, 12 self-attention heads as 12, hidden size of 768, 110 million parameters) is impossible. So we downloaded a pretrained version of BERT. During pre-training, the model was trained on large

amounts of unlabelled text data for one main task : Masked Language Modeling. Based on the context of the surrounding words, the model had to predict a certain number of words of the input (15%) where some were replaced with a special [MASK] token (80%), some were replaced by a random word (10%) and some were unchanged (10%). To achieve this in practice, we would create masked versions of the dataset before training, and during training and we would only add the losses corresponding to the words that the model had to predict.

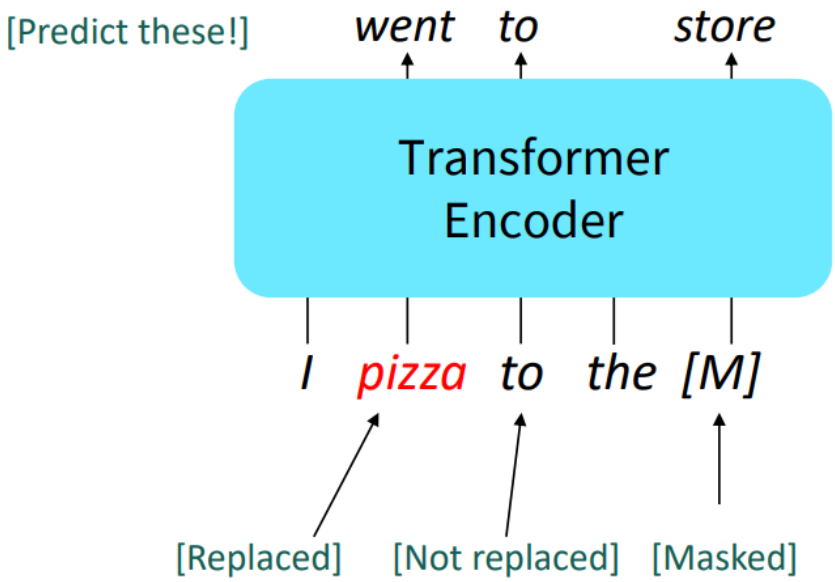

Figure 1.2: BERT masked training

For the fine-tuning, we swapped the final language modelling head for a binary classification head predicting two logits and used the Cross Entropy loss. We split the training data into training and validation datasets (80-20%) in a stratified way in order to measure the validation accuracy and the training and validation loss over epochs. At the end of the training loop we would always choose the model with the highest validation accuracy to prevent the model from overfitting. Splitting the training data into validation data also allowed us to fine-tune the different hyperparameters such as the learning rate and the hidden dimension or the number of layers for the classification head without overfitting on the test set.

1.3 DistilBERT and RoBERTa

For this challenge, we tried two different models derived from BERT. We first used the DistilBERT model which is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than BERT, runs 60% faster while preserving over 95% of BERT's performances as shown in the DistilBERT paper.

We also tried the RoBERTa model which is built on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates. Another difference is that in BERT, the masking is performed only once at data preparation time. Therefore, the variations of each sentence are limited. Whereas in RoBERTa, the masking is done dynamically during training which allows more variations for the masking.

Chapter 2

Hybrid Model

While the RoBERTa model gives us pretty accurate results, we wanted to go further by creating a hybrid model, which uses outputs from our RoBERTa model as well as some other features.

2.1 Relevant Features

The additional features we used for this model were:

1. len : The number of tokens ,i.e. number of words in the input text.
2. n_sentences : The number of dots counted in the input text, which corresponds roughly to the number of sentences
3. n_upper : The number of capital letters contained in the input text.
4. sentence_variance : a measure of the variance of sentence length, which to be precise is the length of the longest sentence divided by the length of the shortest one.

These Features are mostly relevant because they are hard or even impossible to take into account by the RoBERTa model. For instance, the tokenizer does not take into account upper and lower case. Thus giving our model this supplementary information makes sense, and on top of that, it is likely that AI-generated sentences also have trouble with uppercase usage for the exact same reason.

Length of the input is also hardly measured by the RoBERTa model, even though their is a 5% difference in the mean length off human written and AI generated text (AIs write longer text for some reason).

As simple classifier models using only the len, n_sentences and n_upper features could get over 75% accuracy results, we hoped for this hybrid model to improve our first model which it did.

Plotting our data points shows quite clearly how this allows us to distinguish between more data points:

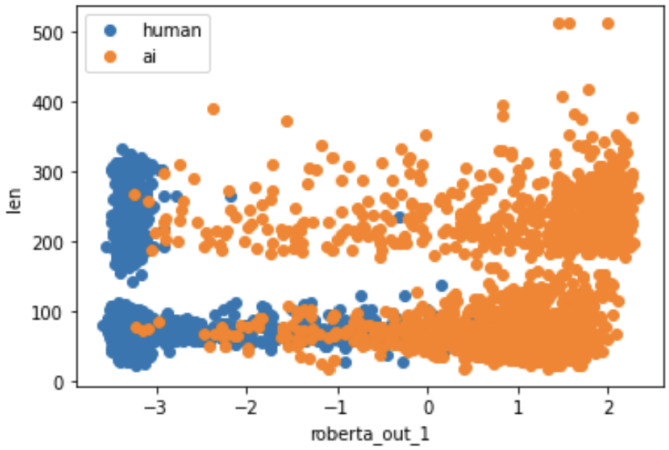

Figure 2.1: Scatter plot of the results of the RoBERTa model crossed with the length of input

As shown above, there is a good amount of points the RoBERTa model cannot predict (those who are centered on the x-axis). However the length feature allows to clearly distinguish a fair amount of the longer texts which were AI-generated (middle of the plot). It also becomes quite evident that the dataset is composed of at least 2 different clusters of text, On with text with around 0 to 150 words and the other with 150 to 350 words.

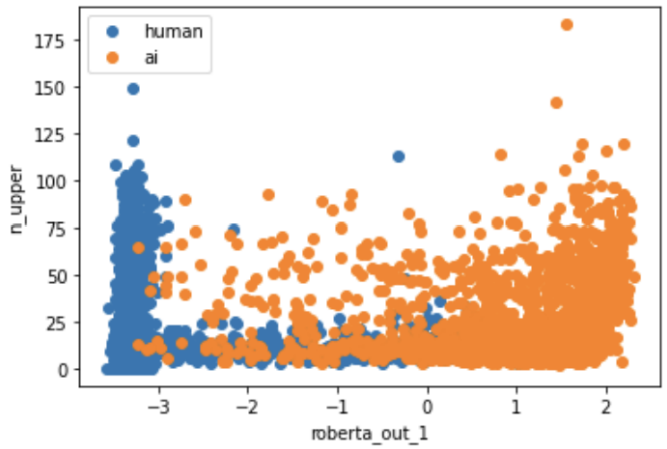

Figure 2.2: Scatter plot of the results of the RoBERTa model crossed with the amount of uppercase letters in the input

Once again, the n_upper feature allows to further distinguish between generated written text.

We also made some importance measures with random forests which will be further discussed in Section 3.3

2.1.1 Perplexity

The idea of computing perplexity stemmed from the following observation : In general, ordinary writing often select unpredictable words that make more sense with the theme of the text contrary to AI generated texts. Hence, with a language modelling model we should observe a lower perplexity on AI generated text compared to human written texts. Therefore we computed the perplexity of input sentences with the GPT2 model. The difference in perplexity for human and AI-generated text wasn't as high as expected (of around 20% when looking at mean values). Surprisingly, the most discriminating feature was the perplexity of the last token.

Combining previously discussed features with perplexity allowed for a rather decent model with around 82% accuracy. However we chose not to incorporate this to our final model for two main reasons. Computing perplexity is very time consuming. And It did not significantly improve the performance of our final model. We believe that the information obtained through perplexity was redundant with the one obtained through the RoBERTa model.

2.2 Model

The classifier model we used for our predictions was first a Gradient Boosted Trees Classifier from the sklearn.ensemble library. We later moved to the Catboost Classifier from the catboost library, known for its performances, which slightly improved our results.

A grid search for choosing best parameters yielded that using 150 estimators worked best, while standard parameters for the rest were optimal. We coupled the model to a Standard-Scaler to preprocess the data.

We also tried using a random forest, Knn and SVM classifiers, which yielded worse results compared to the Gradient Boosted Trees classifier.

Chapter 3

Results

3.1 Transformer based models

3.1.1 DistilBERT

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	0.477010	0.783750
2	No log	0.445592	0.795000
3	0.467300	0.478244	0.803750
4	0.467300	0.485554	0.813750
5	0.231200	0.483130	0.823750

Figure 3.1: DistilBERT results after hyperparameter fine-tuning(81.0% on test set)

The results obtained with DistilBERT coded with the huggingface pipeline can be observed above. This was a significant improvement compared to the baseline model based on the TF-IDF features showing the power of the transformer attention model. Training was very fast as well since this is a small model (2min 10 on google colab pro with GPU).

3.1.2 RoBERTa

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	0.387013	0.828750
2	No log	0.321231	0.872500
3	0.358300	0.390982	0.860000
4	0.358300	0.666108	0.823750
5	0.149500	0.702420	0.821250

Figure 3.2: RoBERTa results after hyperparameter fine-tuning (86.2% on test set)

Here are the results obtained with RoBERTa. There is still a significant improvement compared to the smaller DistilBERT model. Training was however slower (7min 20 on google colab pro

with GPU). We did also recode the whole model, training and dataset class in an attempt to have more control (over the hidden size, number of layers etc...) to further fine-tune the model but we did manage to get significantly better results that what we had already achieved with the huggingface pipeline. We did try however to make the model more explicable using SHAP values (see appendix). SHAP values showed that long "predictable" sequences made it believe that is was generated by a AI whereas irregular punctuation, abbreviations such as "mph" and uncommon sequences of words made it believe it was written by a human.

3.2 Statistical features model

3.2.1 Text vectorizers

A first approach was to naively try different types of vectorizers and classifiers to fit the data.

Vectorizers	Number of features	Classifiers
TF-IDF	10	Logistic Classifier
CountVectorizer	100	Decision Tree
/	1000	Random Forest
/	10000	Gradient Boosting
/	31865	/

By doing a grid search - cross validation on the hyper-parameters of the classifiers, we obtained a maximum score of 67%. We were also able to extract some interesting data like in 3.7.

3.2.2 Text features

Using 5 fold cross-validation on the training set with a Gradient Boosted Classifier, we obtained a 75% accuracy with the features len, n_sents (number of sentences) and n_upper (number of uppercase letters only).

In Figure 3.6, we display the importance of these features, measured through Mean decrease in Impurity on a random forest.

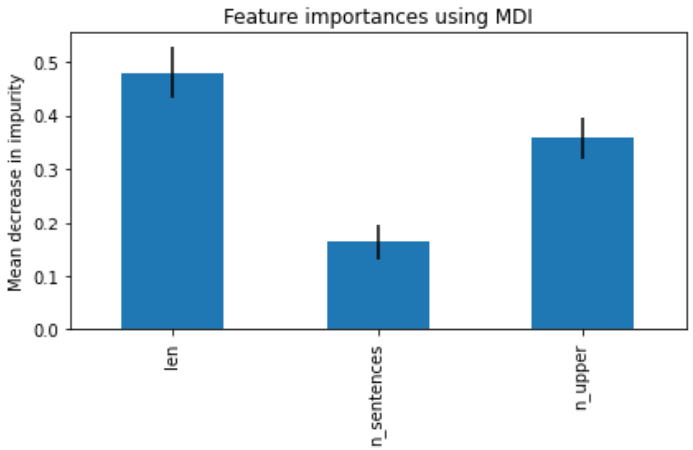

Figure 3.3: Feature importance extracted from a random forest for statistical features

3.2.3 Statistical features model and perplexity

Using perplexity measures combined with some other feature, we obtained at most 82% accuracy measures using cross-validation on the training set.

In Figure 3.4, we display the importance of different features we extracted from the outputs of the GPT2 model. For each input word we extract the probability that the GPT2 model will predict this input word given the left context. We can then extract quantiles, mean values, min and max values and the perplexity of the first and last token.

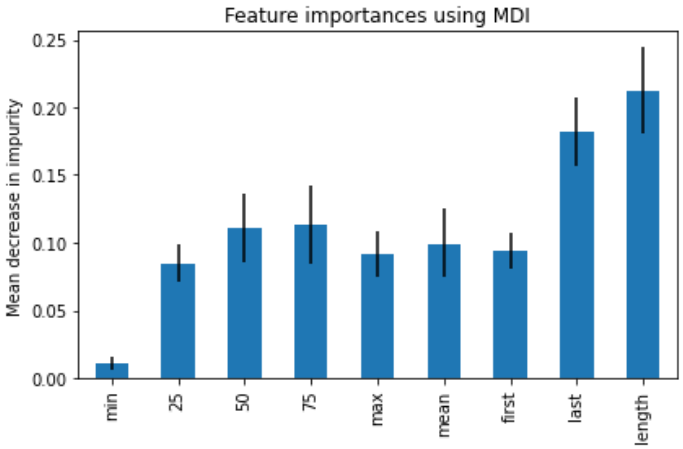

Figure 3.4: Feature importance extracted from a random forest for perplexity features

3.3 Hybrid model

The best score we obtained with our hybrid model is 91% on test set. Using cross-validation on the training set, we obtained 93.5% accuracy. We believe that this discrepancy is due to using RoBERTa outputs as an input, which are more likely to overfit on the training-set.

Figure 3.5 shows that the outputs from the RoBERTa model are what the classifier uses primarily to distinguish between written and generated text.

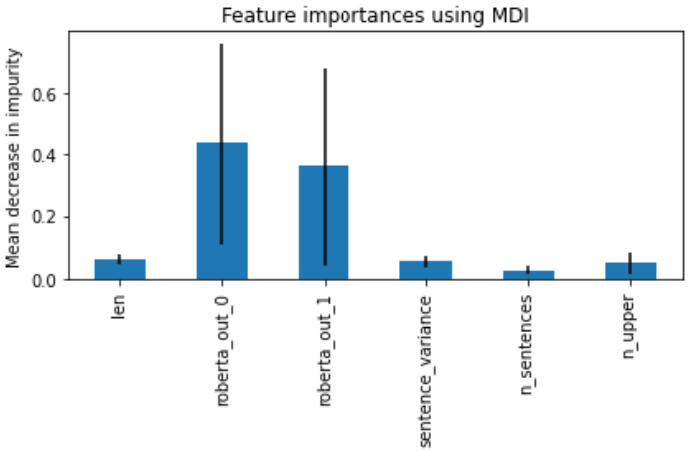

Figure 3.5: Feature importance extracted from a random forest for the hybrid model

Conclusion

While humans have a hard time distinguishing between generated and written text, AIs can help us by snitching on themselves. Our model could correctly classify over 90% of the test-set. The 10% of errors left is due to texts being too short to be able to distinguish between a machine written text and a human written text. Our study shows that even though transformer based models are already extremely powerful, coupled with an ordinary machine learning model and the right features, it can become even better.

Some concerns we have with our model are that it probably overfitted the small dataset we had. Text lengths were very clustered and looking at the texts, a good part of them seemed to be news article resumes. Moreover, the generated texts don't seem to be generated by ChatGPT but by a less powerful transformer model making the task much easier. Today with ChatGPT and GPT-4 it is much harder to determine if the text was generated or not and our trained model probably won't be able to make the distinction. Thankfully, OpenAI has already been working on this problem and created DetectGPT. However, it can only take inputs with at least 500 words and can only achieve 76% accuracy.

Appendix

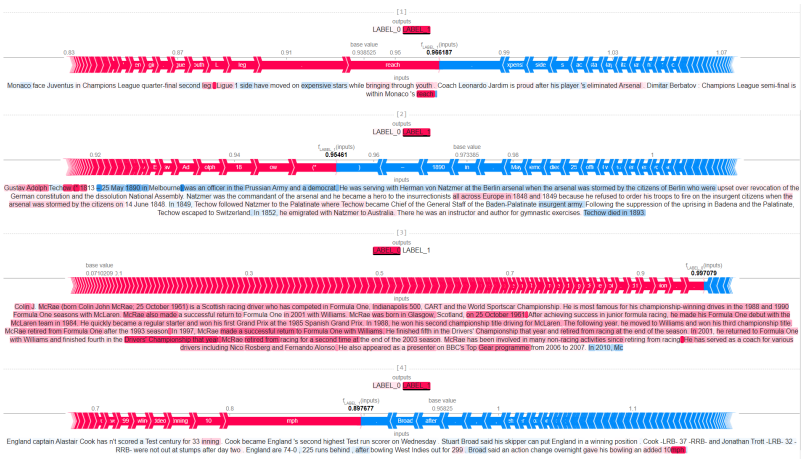

Figure 3.6: SHAP values obtained for the RoBERTa model

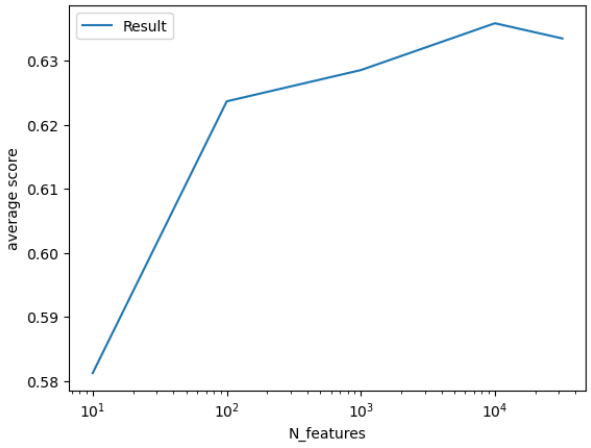

Figure 3.7: mean score against number of features for TF-IDF