INF582 - Introduction to Text Mining and NLP

Human Written Text vs. AI Generated Text

Maximilien Chau[1] and Haileleul Zeyede Haile[2]

[1]M2 Data AI, maximilien.chau@ip-paris.fr
[2]M2 Data AI, haileleul.haile@ip-paris.fr

March 16, 2023

1 Introduction

This papers describes a semester data challenge conducted at Ecole Polytechnique/Institut Polytechnique de Paris for the INF582 Introduction to Text Mining and NLP course. The goal of this project is to study and apply machine learning/artificial intelligence techniques to predict whether a paragraph is written by a human or generated by an AI. We are tasked with studying existing standards and methods for analyzing a text sequence and design a model capable of classifying a given text sequence as whether it was generated by an AI system of by a human with good accuracy. We are given a train dataset which is labeled to train and validate our model and a test dataset which is used to compete in a data challenge on Kaggle.

2 Background

Text generation in AI refers to the process of creating natural language text using artificial intelligence models. These models can be trained on large datasets of human language, allowing them to generate new, coherent text based on patterns and structures they learn from the data. Text generation has now become a trend with the introduction of ChatGPT. In mid-March of 2023 OpenAI, the company that released ChatGPT, released GPT 4. GPT 4 has promised to perform around 500 times better than its predecessor GPT 3.

The advancement of text generation models poses a challenge in determining whether a text is generated by AI or a human. This becomes extremely important specially in educational systems where teachers expect students to produce text. Nowadays these flagship text generation AI tools can produce text for any context in a matter of seconds. Its really hard or almost impossible for a human to identify whether a text is human or AI generated.

To tackle the above mentioned problem, an elegant approach is to train a deep learning model that can learn the features in the text sequence that can distinguish between the two sources by itself. However, the modern methods need to learn embedding of the language's vocabulary which takes a huge amount of data and training time to achieve a good accuracy. Luckily, transfer learning allows us to use pre-trained embedding and fine-tune them using the dataset from out task. After fine-tuning the word embedding, we can experiment by passing the output from that embedding to a classifier deciding between AI or human generated text.

3 Data Analysis

The dataset is in JSON format with each entry containing *id*, *text*, and *label* values. The *label* values are only present for the training set. The *id* uniquely identifies the text sequence in the dataset. The *text* values hold the actual text corpus for that entry. The *label* values for the training dataset classify the *text* as either 0 or 1 - 0 representing that it was written by a human and vice versa.

An important step in developing Machine Learning or Deep Learning models is evaluating the trained model on a dataset previously unseen by the model. Since the test dataset is not provided with labels and the model's performance on the test set is evaluated on the Kaggle data challenge, we partitioned the training dataset into train and validation sets. To do that we randomly selected 20% of the labeled training dataset as a validation set on which the performance of the model will be evaluated on. Using Stratified 3-Fold, we built more reliable results.

We conducted further analysis on the training data to try to see if there are any features that can be used to distinguish between the AI generated and human generated text. Figure 1 gives us a rough view of how much these features can distinguish a text as human or AI generated if there is significant difference in their distribution. For example, the *wordcount* plot in Figure 1, shows that longer texts from the corpus may be human generated. However, these and the other features considered don't seem to contain as much information to accurately distinguish between human and AI generated text. Hence it could be useful to use deep learning methods which can automatically learn features which can distinguish between the classes.

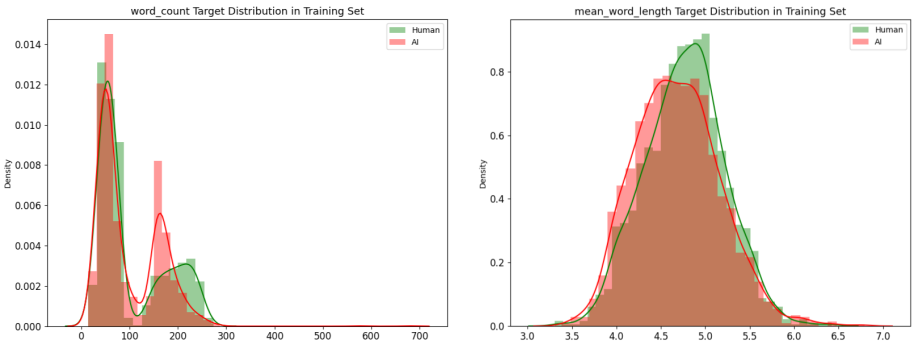

Figure 1: Comparison between AI generated and human generated text samples in the training dataset using features written on top of each plot.

4 Methods

4.1 Building our base model

To measure how different optimization techniques would perform in the classification setting of the challenge, we wanted to use a lightweight pretrained model which would provide both fast, decent and representative results. DistilBERT [3] did exactly fit our requirement. This model is very compact as it removes half of the original BERT layers but maintains good results using the distillation principle.

Using this setting, we have tried different techniques to optimize our classifier:

- Freezing the first layers of the transformer to focus the training on the classifier. This enables the transformer to keep coherent embedding of words and context, not too highly impacted by the classification task.
- Pooling techniques on the embeddings to find the best representation of the sequences.
- Slight architecture changes including the addition of meta-features, dropout layers or a preclassifier.

4.2 Embedding Pooling Techniques

Pooling techniques are commonly used in machine learning to reduce the dimension of some data while keeping the most relevant information. These techniques find their way in Natural Language Processing as we can apply convolution on sequences of tokens. Moreover, they can be applied on top of high dimensional embeddings. Indeed, the outputs of the transformer encoders store a very large representation of their input. To perform a task as text classification, the whole information captured by the embedding of every token of the sentence might not be necessary. To summarize the information captured by the model, we can use various pooling techniques.

CLS Pooling The classification head is built on top of the sequence of embeddings output by the model. The encoding of each sentence starts with the encoding of the <CLS> token, while some SEP tokens split the sentence in smaller sequences. The CLS Pooling uses only the embedding on the CLS token from all the embeddings output by the model.This token has the role of encoding the whole sequence, which will be learned during the training of the model to improve on the classification task.

Concatenation Pooling BERT among other transformers stacks encoder layers on top of each other. First layers contain no contextual information. Last layers contain information very specific to BERT's pre-training tasks, like Masked Language Modelling and Next Sentence Prediction. However, the challenge does not require the model to fit those tasks. To accurately predict the Human vs AI generated text, the embeddings have to compress enough contextual information to caption the meaning of the word, while remaining abstract enough. Concatenate Pooling takes advantage of the different hidden representations of language models [5] by stacking different layers encoding embeddings. The classification head can then pull the most relevant information.

Other Pooling techniques In our experiments, we tried various other pooling techniques like mean Pooling, max Pooling and mean-max Pooling.

4.3 Model Optimization

After experimenting different optimization ideas on a lighter model, it made sense for us to try larger and more efficient pretrained models. Looking at the State-of-the-art models, we chose to finetune a few of them on our AI generated classification task. As this task is only similar to the common sentiment analysis benchmarking, the results of the models were expected to vary from the original global benchmarks like GLUE or SuperGLUE.

RoBERTa[2] is a model based on BERT but improves it by doing several modifications. RoBERTa trains on a larger dataset which improves the performance of an "under-trained" BERT model. Additional design choices such as using variable mask sizes in training or more optimal encoding make it more efficient on a variety of downstream tasks. Another pretrained model we used is DeBERTa[1]. DeBERTa uses a slightly different approach to training, called dynamic masking. It handles long sequences of text better and performs well on tasks that require reasoning and attention to detail. In the training however, we could not replicate better results with the base model even when augmenting the maximum sequence length size. Both models share large versions with more parameters and a bigger hidden size of embeddings. To train those models, we had to reduce the batch size to fit on the GPU's RAM.

Optimizing the best model Once we found a new good baseline model for the classification task, we tried to optimize several hyperparameters[4] to reach a better accuracy. The most important ones were the number of epochs, the learning rate, and the maximum sequence length. Through experiments, we learned that for classification models based on pretrained language models, it was best not to tweak too much the classification head and a simple architecture is often the optimal to get the best results.

5 Results

Models	**Loss**	**Accuracy**	**Training time (s)**
DistilBERT-base-cased	0.336	0.848	257
DistilBERT-base-uncased	0.448	0.789	259
DistilBERT-base-cased with frozen layers	0.676	0.613	112
DistilBERT-base-cased with preclassifier	0.744	0.831	258
DistilBERT-base-cased with mean Pooling	0.382	0.836	262
RoBERTa-base	0.62	0.872	340
RoBERTa-large	0.488	0.885	1410
DeBERTa-base	0.689	0.825	500
DeBERTa-large	**0.284**	**0.903**	954

Table 1: Metrics computed over an evaluation set

Our experiments on the baseline DistilBERT model show that some techniques were inefficient on this task. All the models in the table feature CLS pooling, retrieving information from the embedding to pass it to the classifier.

Uncasing the text to make it simpler might have led to too much information loss, which resulted in a lower accuracy.

Although we tried to freeze the pretrained layers and only finetune the classifier with a low learning rate and number of epochs, it did not work. However, the training remained faster as we could expect since there are less layers to train.

Pooling techniques overall had a positive impact on the results of the classifier. The simple CLS Pooling was the most efficient. Other pooling techniques not present in the table did not improve our model.

Bigger pretrained model were the most important to get a high accuracy. Tuning them correctly was not easy as they were quite sensible to hyperparameters changes.

Another interesting thing to notice is how the embedding layer inside the model clusters words from a given sample text. This helps us visualize which words the model considers have similar semantic meanings or contexts in the training text corpus. To visualize the placing of words from a sample text on our embedding space we use the dimensionality reduction technique t-SNE (see Figure 2).

6 Conclusion

This challenge taught us a lot about several aspects of NLP. We applied data analysis on a practical case. This was useful to get some insight on the data we were dealing with. From there, we could think of ways of improving a baseline model, using meta-features. We learned new optimization techniques that can be used in a text classification task. We also have a better understanding of transfer learning in practice and the finetuning going in parallel. In the end, results showed that bigger pretrained models are the best for this text classification task and the pretrained model has the most important role for the accuracy.

References

[1] Pengcheng He et al. "Deberta: Decoding-enhanced bert with disentangled attention". In: *arXiv preprint arXiv:2006.03654* (2020).

[2] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[3] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[4] Chi Sun et al. "How to fine-tune bert for text classification?" In: *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*. Springer. 2019, pp. 194–206.

[5] Junjie Yang and Hai Zhao. "Deepening Hidden Representations from Pretrained Language Models". In: *arXiv preprint arXiv:1911.01940* (2019).

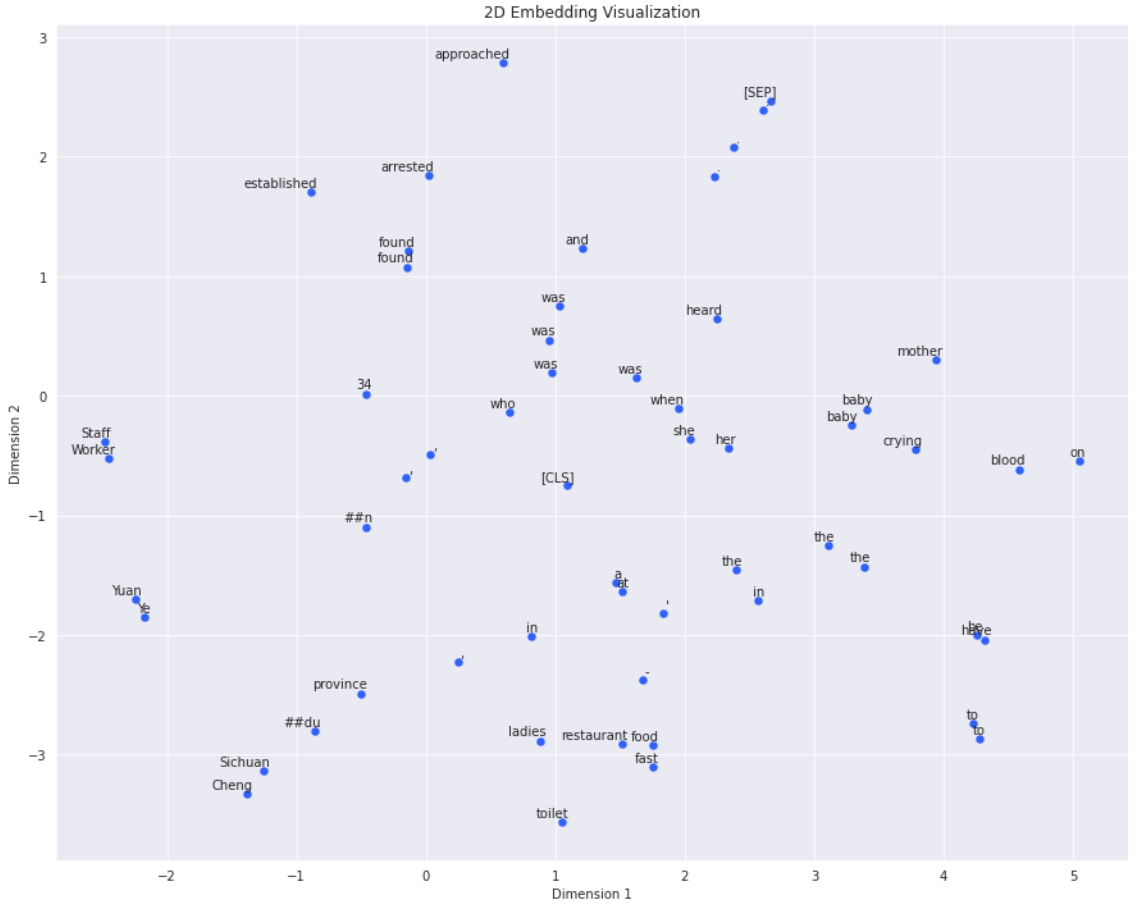

Figure 2: Words from a sample text on the t-SNE reduced embedding space. Words with similar context such as *mother* and *baby* closer on the embedding space.