Data Challenge: Human Written Text vs. AI-Generated Text

Laychiva Chhout, Emmanuel Gnabeyeu

École Polytechnique

16 March 2023

Presentation Plan :

Project Overview

- Goal : Classify Human Written Text and AI-Generated Text.
- Dataset : 4000 tuples of the training set and 4000 tuples of the test set.
- Evaluation Metric :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

Project Overview

- Goal : Classify Human Written Text and AI-Generated Text.
- Dataset : 4000 tuples of the training set and 4000 tuples of the test set.
- Evaluation Metric :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.
- Our approach :

Feature engineering

Aim : Extract some relevant features to distinguish Human-written vs AI-generated texts.

- Vocabulary richness (diverse and varied vocabulary), the sentence length.
- Word frequency, grammar usage, and the style of phrases(tone, register, figurative language).
- The syntax, semantics, etc.
- The subject matter of the text(social and cultural context) : awareness vs sensitivity

NLP's techniques for Feature extraction

Aim : use basic and more advanced NLP techniques to extract more nuanced information Human-written vs AI-generated texts

- Word frequency, style and tone of the text :
  - Tokenization and counting or LDA and TD-IDF (capture the important words)
  - Sentiment polarity of the text (positive, negative, neutral) to capture the emotional tone of the text

NLP's techniques for Feature extraction

Aim : use basic and more advanced NLP techniques to extract more nuanced information Human-written vs AI-generated texts

- Word frequency, style and tone of the text :
  - Tokenization and counting or LDA and TD-IDF (capture the important words)
  - Sentiment polarity of the text (positive, negative, neutral) to capture the emotional tone of the text
- Capture information about the syntax and semantics :
  - Sequence models such as N-grams(contiguous sequences of N words in a text)
  - e.g. by varying lengths : unigrams, bigrams, trigrams

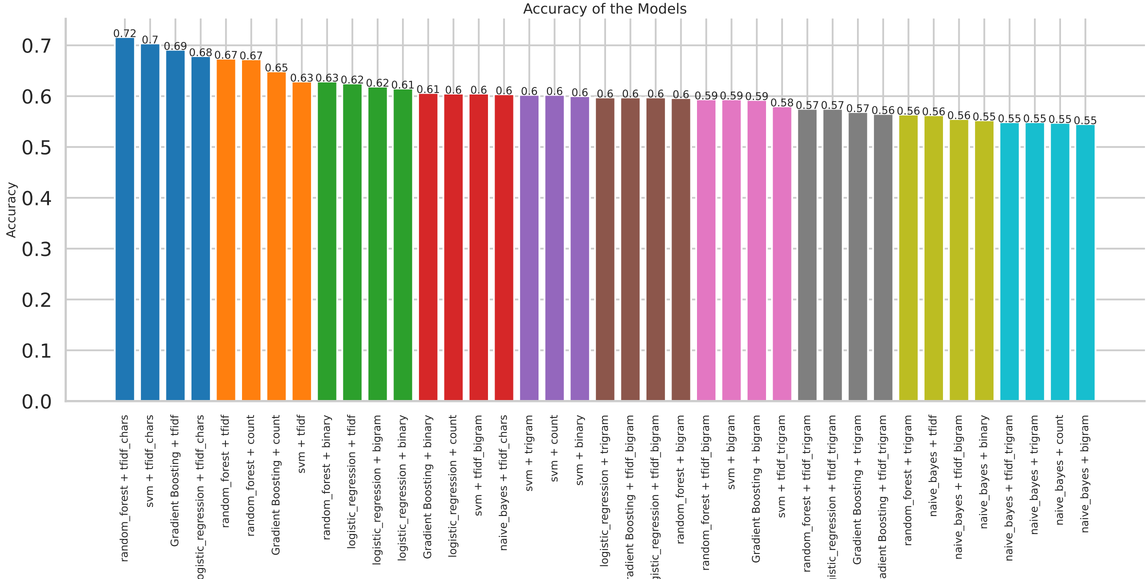
NLP's techniques for Feature extraction

Aim : use basic and more advanced NLP techniques to extract more nuanced information Human-written vs AI-generated texts

- Word frequency, style and tone of the text :
  - Tokenization and counting or LDA and TD-IDF (capture the important words)
  - Sentiment polarity of the text (positive, negative, neutral) to capture the emotional tone of the text
- Capture information about the syntax and semantics :
  - Sequence models such as N-grams(contiguous sequences of N words in a text)
  - e.g. by varying lengths : unigrams, bigrams, trigrams

Classification algorithms Binary classification problem (Logistic Regression, Random Forest, Support Vector Machines, Naive Bayes, and Gradient boosting classifiers).

Classification algorithms



Avanced NLP's techniques for Feature extraction

Aim : use basic and more advanced NLP techniques to extract more nuanced information Human-written vs AI-generated texts

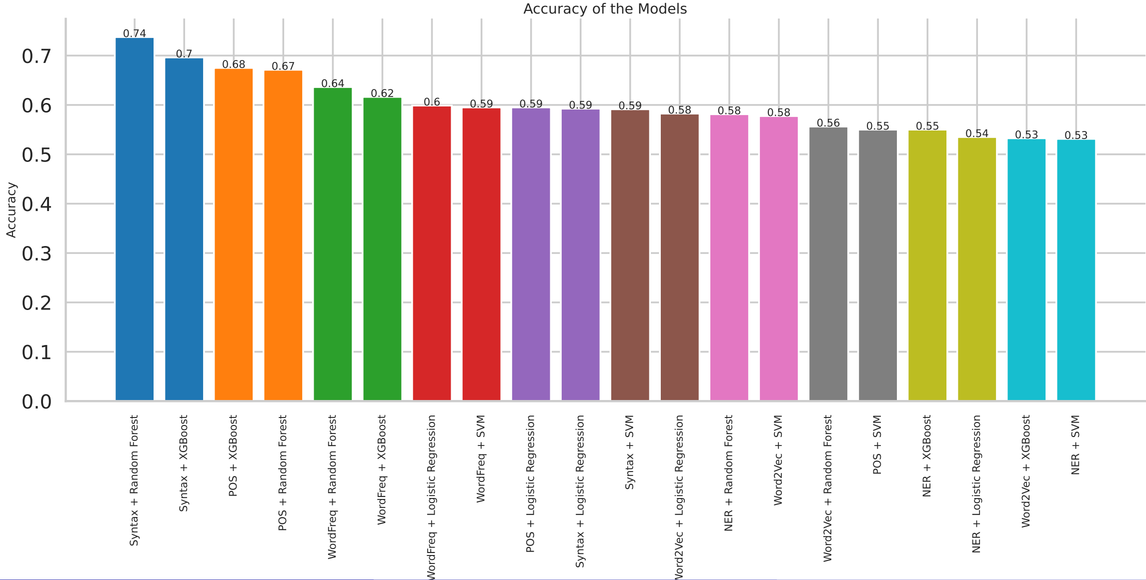
- Grammar usage and syntax :
  - Part-of-speech (POS) tagging (label each word with its corresponding part of speech)
  - Syntactic parsing (analyze the grammatical structure, capture the relationship)

Avanced NLP's techniques for Feature extraction

Aim : use basic and more advanced NLP techniques to extract more nuanced information Human-written vs AI-generated texts

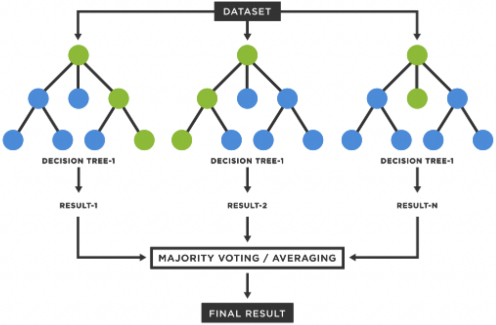
- Grammar usage and syntax :
  - Part-of-speech (POS) tagging (label each word with its corresponding part of speech)
  - Syntactic parsing (analyze the grammatical structure, capture the relationship)
- Capture information about the semantics(meaning of words) of a text :
  - Named entity recognition(NER) : identifying and classifying named entities in a sentence (e.g., person, organization, location, etc.)
  - Word embeddings (transfer learning) : dense vector representations of words that capture their meaning and context

A Comparative Study



A Comparative Study

- Best predictors :
  - Syntactic parsing
  - Part-of-speech (POS)
  - Word frequency
- Best Algorithms : Random Forest
  Reasons :
  - Categorical Data
  - Aggregates the result of many decision trees and then outputs the most unbiased result
  - A random forest produces good predictions that can be understood easily.
- Hyperparameter tuning (Grid Search using Scikit Learn)



Pre-trained LLMs from Huggingface

BERT, RoBERTa, XLNet

Architecture of RoBERTa

- Embedding layer : provide meaning, position, and sentence separation of words in the input sequence.
- Transformer layers : The self- attention mechanism calculates attention scores between different words in the input sequence and uses them to obtain a context vector for each word.
- Output layer : Map the output of the last transformer layer to a result.

```
==== Embedding Layer ====

roberta.embeddings.word_embeddings.weight                    (50265, 768)
roberta.embeddings.position_embeddings.weight                  (514, 768)
roberta.embeddings.token_type_embeddings.weight                  (1, 768)
roberta.embeddings.LayerNorm.weight                                (768,)
roberta.embeddings.LayerNorm.bias                                  (768,)

==== First Transformer ====

roberta.encoder.layer.0.attention.self.query.weight            (768, 768)
roberta.encoder.layer.0.attention.self.query.bias                  (768,)
roberta.encoder.layer.0.attention.self.key.weight              (768, 768)
roberta.encoder.layer.0.attention.self.key.bias                    (768,)
roberta.encoder.layer.0.attention.self.value.weight            (768, 768)
roberta.encoder.layer.0.attention.self.value.bias                  (768,)
roberta.encoder.layer.0.attention.output.dense.weight          (768, 768)
roberta.encoder.layer.0.attention.output.dense.bias                (768,)
roberta.encoder.layer.0.attention.output.LayerNorm.weight            (768,)
roberta.encoder.layer.0.attention.output.LayerNorm.bias            (768,)
roberta.encoder.layer.0.intermediate.dense.weight             (3072, 768)
roberta.encoder.layer.0.intermediate.dense.bias                   (3072,)
roberta.encoder.layer.0.output.dense.weight                   (768, 3072)
roberta.encoder.layer.0.output.dense.bias                          (768,)
roberta.encoder.layer.0.output.LayerNorm.weight                    (768,)
roberta.encoder.layer.0.output.LayerNorm.bias                      (768,)

==== Output Layer ====

classifier.dense.weight                                        (768, 768)
classifier.dense.bias                                              (768,)
classifier.out_proj.weight                                       (2, 768)
classifier.out_proj.bias                                             (2,)
```

Figure – RoBERTa Architecture

BERT, RoBERTa, XLNet comparisons

- Data : we used 90% of the training set as training data and 10% of the training set as validation data with **batch_size** = 16
- Optimizer : we used **AdamW**[1] with a learning rate of 5e-5 and an epsilon value of 1e-8.
- Trained on GPU NVIDIA A100-SXM4-40GB (Colab Pro Premium).

1. https ://huggingface.co/docs/transformers/main_classes/optimizer_schedules

BERT, RoBERTa, XLNet comparisons

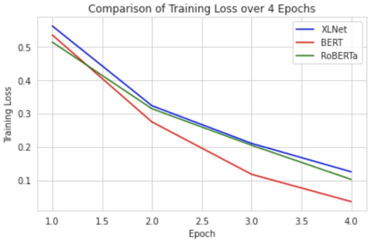


Figure – Training Loss on 4 epochs

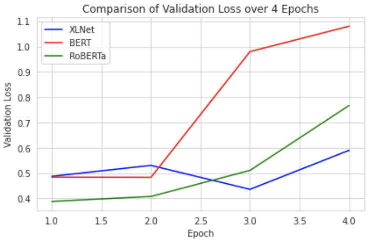


Figure – Valid. Loss on 4 epochs

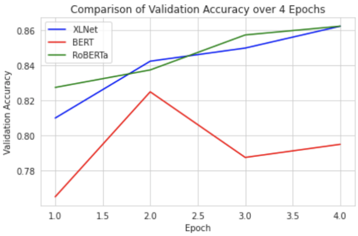


Figure – Valid. Acc on 4 epochs

BERT, RoBERTa, XLNet comparisons

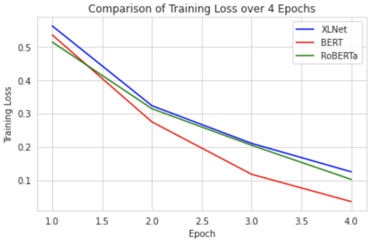


Figure – Training Loss on 4 epochs

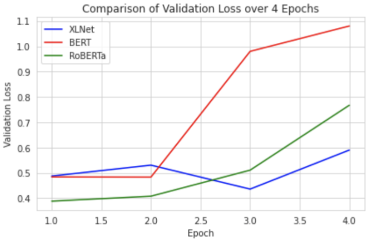


Figure – Valid. Loss on 4 epochs

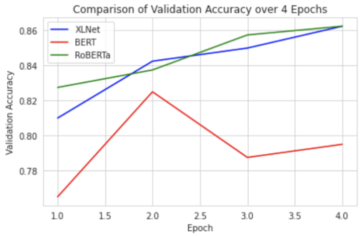


Figure – Valid. Acc on 4 epochs

Our comments :

- Validation loss saturates pretty quickly while the training loss continues to lower.
- The models are powerful and start to overfit if trained for longer.
- RoBERTa and XLNet perform best among these 3 pre-trained LLMs, but RoBERTa is **BETTER**.

Avoid Overfitting : Regularization

The usage of weight decay in **AdamW**

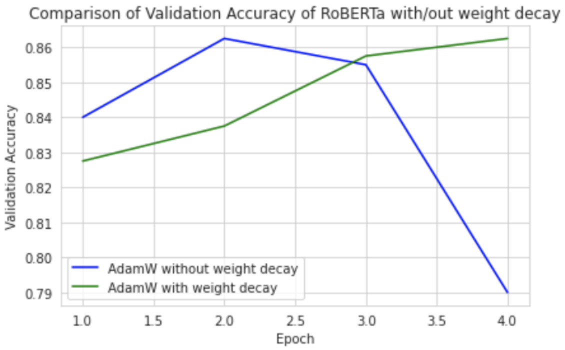


Figure – Comparing model with and without weight decay in AdamW

Avoid Overfitting : Regularization

The usage of weight decay in **AdamW**

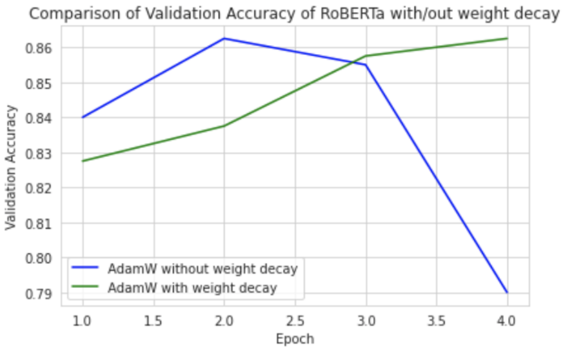


Figure – Comparing model with and without weight decay in AdamW

Our comment : It has been demonstrated that our model is overfitted when regularization is not implemented.

Our final result

epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	0.46	0.30	0.88	0:00:56	0:00:02
2	0.21	0.32	0.88	0:00:56	0:00:02

Table – RoBERTa results on 2 epochs

Conclusion

In conclusion, this data challenge allowed us to :

- expand our understanding of NLP and fundamental techniques in text preprocessing like feature selection/extraction, and data cleaning.
- provides insights into the capabilities of current pre-trained models.

Further step

We will try to improve our performance in Kaggle by trying another pre-trained LLMs :

Model	Core differentiator	Pre-training objective	Para-meters	Access	Information Extraction	Text Classification	Conversa-tional AI	Summari-zation	Machine Translation	Content generation
BERT	First transformer-based LLM	AE	370M	Source code	Highly appropriate	Highly appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
RoBERTa	More robust training procedure	AE	354M	Source code	Highly appropriate	Highly appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
GPT-3	Parameter size	AR	175B	API	Somewhat appropriate	Somewhat appropriate	Highly appropriate	Appropriate	Appropriate	Highly appropriate
BART	Novel combination of pre-training objectives	AR and AE	147M	Source code	Appropriate	Somewhat appropriate	Appropriate	Highly appropriate	Appropriate	Highly appropriate
GPT-2	Parameter size	AR	1.5B	Source code	Somewhat appropriate	Somewhat appropriate	Highly appropriate	Appropriate	Appropriate	Appropriate
T5	Multi-task transfer learning	AR	11B	Source code	Appropriate	Appropriate	Somewhat appropriate	Highly appropriate	Appropriate	Highly appropriate
LaMDA	Dialogue; safety and factual grounding	AR	137B	No access	Appropriate	Somewhat appropriate	Highly appropriate	Somewhat appropriate	Somewhat appropriate	Appropriate
XLNet	Joint AE and AR	AE and AR	110M	Source code	Appropriate	Highly appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate	Somewhat appropriate
DistilBERT	Reduced model size via knowledge distillation	AE	82M	Source code	Highly appropriate	Appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
ELECTRA	Computational efficiency	AE	335M	Source code	Highly appropriate	Appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
PaLM	Training infrastructure	AR	540B	No access	Somewhat appropriate	Appropriate	Highly appropriate	Highly appropriate	Highly appropriate	Highly appropriate
MT-NLG	Training infrastructure	AR and AE	530B	API	Somewhat appropriate	Highly appropriate	Highly appropriate	Appropriate	Somewhat appropriate	Highly appropriate
UniLM	Optimised both for NLU and NLG	Seq2seq, AE and AR	340M	Source code	Appropriate	Appropriate	Highly appropriate	Highly appropriate	Somewhat appropriate	Highly appropriate
BLOOM	Multilingual (46 languages)	AR	176B	Source code	Appropriate	Appropriate	Appropriate	Appropriate	Highly appropriate	Highly appropriate

AR = Autoregression
AE = Autoencoding
Seq2seq = Sequence-to-sequence

Highly appropriate
Appropriate
Somewhat appropriate

Figure – Table 1 : Summary of the features of the most popular Large Language Models