IE684 Web Mining Project Report
The AuTexTification: The Automated Text Identification

Poey Sie Chuah[*](1904644), Kok Teng Ng[†](1936360)

June 30, 2023

Abstract

This paper addresses the problems and challenges encountered in the dataset released by the Automated Text Identification competition's organizer, known as (AuTexTification, 2023). This study focuses on utilizing both pre-trained models known as BERT and XLM-RoBERTa tokenization function and model, to preprocess the obtained data and predict labels. The AuTexTification competition consists of two subtasks for participants to complete, each subtasks belong to its own training and testing dataset. The first subtask's training dataset contains 33,845 rows of sentences, each labeled as 'human' or 'generated' for binary classification problems. The second subtask's training dataset contains 22,416 rows of sentences, classified into six labels that represent those texts generated by six different text generation models. This paper discusses how the team was handling and thought on dealing with those 'dirty' words in the dataset and fine-tuning pre-trained models to achieve the goal of this study. In addition, this paper also discusses how to do hyperparameter optimization for letting the model become more accurate by using macro-F1 score to measure while dealing with these kinds of datasets.

Keywords: AuTexTification, Natural Language Processing, PyTorch, BERT, XLM-RoBERTa, Optuna, Trainer API, Hyperparameter Optimisation, Google Translate API

1 Introduction

This paper discusses the techniques and model that the team used while participating in the AuTexTification competition, which consists of two subtasks and two languages for each subtask as English and Spanish, but the team focused on English language processing. The competition provides separate training and testing datasets for each subtask, and those sentences in both datasets include foreign languages, emoji, emoticons, and non-UTF-8 encoded words.

The first subtask's training dataset consists of 33,845 rows of sentences and is labeled as either automatically generated by computer-generated or human-written. The second subtask's training dataset consists of 22,416 rows of sentences labeled as six different labels representing the text generated by six different text generation robots. Preprocessing techniques are important for cleaning the data to improve the model accuracy, those techniques include natural language processing methods. In addition, the data needs to be converted into PyTorch format that is able to fit into pre-trained models for fine-tuning purposes (Fine-tuning With Custom Datasets — Transformers 3.2.0 Documentation, n.d.).

For overcoming these kinds of challenges, the team utilized the Trainer API provided by HuggingFace, which is a kind of high-level API for training and fine-tuning transformer models. The Trainer API also makes Optuna the backend for executing hyperparameter optimization, this would help us to generate a more accurate model for classifying those sentences in both tasks (Trainer, n.d.). The team used the generated result from the fine-tuned models and fit them into the evaluation file specified by the competition's organizers by using the Macro-F1 score then made performance comparisons on both implemented models.

[*]University Mannheim, email: poey.chuah@students.uni-mannheim.de, Mannheim Master of Data Science

[†]University Mannheim, email: ng.kok.teng@students.uni-mannheim.de, Mannheim Master of Data Science

Overall, this paper discusses the team's approach to preprocessing the data, utilizing Trainer API and Optuna for searching hyperparameters and fine-tuning the model to let the model understand the downstream tasks that want to execute afterward, and evaluating the performance by using the provided evaluation file.

1.1 Problem Formulation

Based on the collected datasets from the AuTexTification competition that consists of two subtasks, those provided datasets contain 'dirty' words that need to be preprocessed by using natural language processing techniques. In addition, the team is required to execute hyperparameter optimization steps for the implemented pre-trained models based on the specific use case due in most cases using the default hyperparameter to fine-tune the model will lead to poor performance (Probst et al., 2019).

After the process of preprocessing and hyperparameter optimization, the team would fine-tune the pre-trained models to let the model know the downstream tasks as what it would need to execute afterward. To conclude the problems, the challenges in both subtasks involve preprocessing the data to handle 'dirty' words, optimizing the hyperparameters for the pre-trained models, and fine-tuning the models to handle the specific subtasks of sentence classification and text generation model identification.

1.2 Literatre Review

There are various natural language processing tasks such as translation, summarization, question answering, and reading comprehension, however, the team would like to focus on the text generation for this report. In recent years, the pre-trained language model has increased the ability of the systems to automatically generate content. For example, this can be done through the usage of language models such as Generative Pre-trained Transformer (GPT). According to (Radford et al., 2018) language models are unsupervised learner which can learn without any explicit supervision during the training of millions datasets. Additionally, these language models are trained using multitask training since it only requires several effective training pairs, rather than thousands of examples for generalization.

The availability of most of these models has enhanced the advancement of innovative applications through research and development. However, the language model can be exploited to disseminate false information, deceptive reviews, biased opinions, or even be toxic to the user. Hence, it is crucial to develop a system to detect generated text so that the language modeling is aligned with the user's intention (Ouyang et al., 2022). Although the Large Language Model (LLM) has shown major improvements in a variety of tasks, we have to consider the potential threats associated with it. The LLM includes BERT, DistilBERT, ALBERT, RoBERTa, GPT-3, etc. to mitigate them (Bender et al., 2021). Moreover, in certain legal and security contexts, simply detecting machine-generated text may not be satisfactory since the evolution of automatic text generation will lead to opinion spam.

According to the research findings by Massarelli, the generated text would show some abnormalities which cause variations in the distributions of automated text generation or human-generated text (Massarelli et al., 2020). Most importantly, the selection of a decoding strategy can greatly influence the overall quality of the text produced since the decoding strategy can produce less verifiable text. The author attempts to investigate the fluency and explore the verifiability of text produced by advanced pre-trained language models where a verifiable sentence is one that can be confirmed or refuted using information from Wikipedia. The decoding strategies include sampling-based which introduces stochastic decisions during the text generation and likelihood-based by defining heuristics to make the generation more practical.

In the other paper (Dugan et al., 2020), the author tried to come up with a system to tackle the challenges of detecting machine-generated text which is quite similar to our task in this project. The author separates the AI into different categories such as short stories, New York Times, and recipes. When interacting with the website, the user has to pick whether the sentences shown are human-written or machine-generated. In the case that machine-generated is chosen, it will be followed up by prompting the user's reasoning for believing that this sentence is generated by a computer. Finally, the system will let you know if you answer it correctly and state some common problems in generated text such as repetition, common-sense errors, factual errors, incoherence, and topic drift (Real or Fake Text, n.d.).

1.3 Objective

The objectives of this study are to preprocess the obtained datasets, specifically by removing 'dirty' words from the sentences, in order to improve the performance based on the specific study case in the datasets (Camacho-Collados & Pilehvar, 2018). The study focuses to demonstrate the utilization of the Trainer API and Optuna as the backend for hyperparameter optimization. The hyperparameters will be used to fine-tune the model and enable its execution in the downstream task as the model could accurately identify whether the sentences were generated by the text generation model or human written; for the second subtask, could also identify which of the six different text generation models generated the provided sentences.

2 Methodology

2.1 Data preprocessing

After reviewing the training and testing datasets and considering the goal of the competition, which is to train or develop a model that could accurately predict whether the given sentences were generated by a text generation model or human-written. Therefore, the team decided to try some preprocessing on those 'dirty' words in the given sentences, at the beginning, could use 'chardet' and 'polyglot.detect' libraries in Python to detect those sentences containing foreign languages based on the returned confidence score on English. Then the team could utilize the Google Translate API to do the translation those sentences into English. Additionally, the team also utilized the library of 'emoji', 'emot.emo_unicode', and natural language toolkit (NLTK) libraries to identify sentences containing emojis and emoticons (using symbols or special characters rather than actual emojis) and then convert them into text form, and also remove those stopwords from the given sentences because that will affect the word embedding weight for each token. Not only that, the team would also like to execute the process of lemmatization for converting those words back to their original form, which will determine on how the performance of converting them back to the original form. If the generated result becomes lower, the team will not execute these processes.

However, there might be some potential issues of converting those foreign languages, emojis, and emoticons that are in the sentences into English, as doing so may result in the loss of information and worse case lead to lower accuracy. In many cases, humans would tend to use their native language when encountering words they do not know in English and use emojis to express their emotions through words. Since the goal of the competition is to accurately classify the given sentences generated by text generation model or human-written, therefore, it does not necessary to convert those emojis and emoticons into text form because the purpose of converting to text form is only useful for sentiment analysis that is not the goal of this competition (Yuqiao et al., 2021). If those implemented pre-trained models were fine-tuned by those well-preprocessed data, it will also need to do preprocessing on those unseen data in the future prediction. That is not that intelligent and might consume more time on preprocessing as it always needed to preprocess well and then can only classify well. By keeping the original language, emoji, and emoticons, the model would be more accurate to determine if a sentence was written by a human. Additionally, since this paper would deploy pre-trained multilingual models such as BERT and XLM-RoBERTa for the classification tasks, these models were trained on multilingual data that contain more than 100 languages from sources like Wikipedia and Google and are able to comprehend multiple languages. Hence, translation of a foreign language into English is not required (Wu & Dredze, 2020).

The team implemented pre-trained models that utilize the Transformer architecture to design and develop their models. These pre-trained models have their own tokenizer, which is really important to preprocess the data to fit into the model. The Transformer architecture consists of two main groups of components, and for pre-training and fine-tuning the model, they only require to utilize of the encoder components. One of the key components in the encoder is self-attention, which will generate attention scores for each token in the inserted sentence. If the tokens receive a high attention score then it might get more attention from the model, additionally, the tokenizer will also generate word embedding weights for each token. Another key component is Feed Forward, which enables those attention scores to be parallelly passed to the next encoder. Those attention scores and word embedding weights are represented as 'attention_mask' and 'input_ids' and are stored in PyTorch format, they can be passed into a pre-trained model for fine-tuning, this process allows the model to understand the downstream task as it might need to perform in the upcoming task (Vaswani et al., 2017).

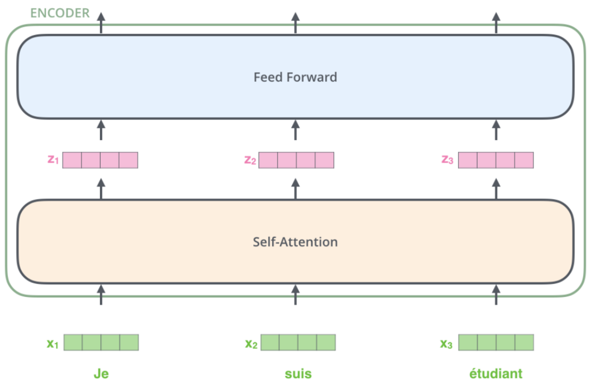


Figure 1: Transformer architecture (Alammar, 2018)

2.2 Models Selected

This paper focuses on implementing the pre-trained model by using the Transformer architecture to be trained by using a large amount of raw text data from Wikipedia and Google and executing the downstream tasks, such as BERT and XLM-RoBERTa.

2.2.1 Hyperparameter Optimization

In hyperparameter tuning, the team uses the Trainer class which provides an API from the Hugging Face library for feature-complete training. First of all, the data is converted into PyTorch datasets before training. The Trainer API is able to fine-tune a pre-trained transformer-based model like BERT and XLM-RoBERTa. The training arguments are defined to specify the parameters. Here is the list of training arguments that we used:

1. Learning rate: Determine the step size at each optimization iteration.
2. Weight decay: Also known as L2 regularization which adds a penalty term to the loss function during training so that the weights have a smaller value.
3. Number of train epoch: The number of times the whole training dataset is passed forward and backward through the model during the training process, have to trade-off between model performance and computational cost.
4. Per device train batch size: During each forward and backward iteration, the number of training examples processed in parallel on a single device.
5. Per device eval batch size: During the evaluation phase, the number of evaluation examples processed in parallel on a single device.
6. Warmup steps: The warm-up phase will use a smaller learning rate to allow the model to stabilize and avoid large gradient updates.
7. Logging steps: To record some information at regular intervals to keep track of the training progress and various metrics.
8. Evaluation strategy: To measure the performance of the model on generalization to unseen data.
9. Eval steps: Determine how often the model's performance is being measured on a separate evaluation dataset.

On the other hand, Trainer supports Optuna as a hyperparameter search backend which provides a flexible and easy-to-use API. Optuna can automatically search for optimal hyperparameters by defining a search space and it will explore different hyperparameter configurations. After exploration, Optuna will prune unpromising trials and optimize the objective function to find the best set of hyperparameters. Below shows the list of trials that we defined to opt for Optuna's suggestion:

Parameter	Range
Learning rates	low=1e-6, high=1e-4
Weight decay	low=1e-6, high=1e-4
Number of train epoch	[1, 2, 3, 5]
Warmup steps	low=10, high=1000
Eval steps	low=10, high=1000

Figure 2: Subtask 1 Hyperparameter Optimization Search Space

Parameter	Range
Learning rates	low=1e-6, high=1e-4
Weight decay	low=1e-6, high=1e-4
Number of train epoch	[15, 18, 20]
Warmup steps	low=10, high=1000
Eval steps	low=10, high=1000

Figure 3: Subtask 2 Hyperparameter Optimization Search Space

2.2.2 BERT

BERT is a NLP model introduced by Google in 2018 and it is based on the Transformer architecture which uses deep learning to process sequential data efficiently (Devlin et al., 2019). BERT utilizes a bidirectional approach that considers the context from both the left and right sides of the sentences during training so that it can capture a more comprehensive representation of language.

BERT is a pre-trained model, meaning it has been trained on huge amounts of text data to learn about general language patterns and semantics. BERT can be fine-tuned with just one additional output layer. There are 2 main steps in pre-training the model, namely Masked Language Modeling (MLM) and Next Sentence Prediction (NSP):

1. In MLM, a portion of the input sentences is randomly masked with a special 'MASK' token by replacing some words with it. Then, the BERT model will attempt to predict the original masked words based on the context of the surrounding words. This step helps BERT to learn the correlation between different words in a sentence.

2. In NSP, BERT joins pre-trains text-pair representations to predict whether 2 sentences in a pair are consecutive or not. This step enables BERT to understand the coherence between sentences.

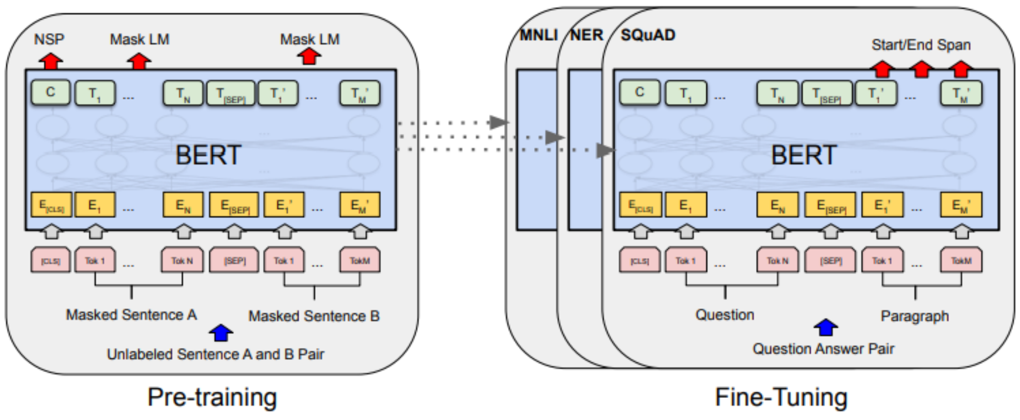


Figure 4: BERT model architecture

While using the tokenization of the BERT model, it will generate a couple of special tokens for each sentence such as classification token [CLS] is added at the beginning of every input, separate token [SEP] is used to separate multiple sentences from the input, and finally, padding token [PAD] is used for padding to ensure that they have the same length. If the sequence is shorter than the maximum length, [PAD] tokens are padded until they match the maximum length.

The pre-trained model parameters are used to initialize models and capture the general language representation. BERT trains the model by initializing tasks with the same pre-trained parameters and can be fine-tuned to a more specific downstream task like in our case, BERT is fine-tuned to better predict whether the text given is generated by a computer or written by a human. During fine-tuning, BertForSequenceClassification is used so that all the parameters are fine-tuned on task-specific labeled data. However, BERT's ability to detect computer-generated text heavily relies on the availability of properly labeled training data.

2.2.3 XLM-RoBERTa

XLM-RoBERTa was developed by Facebook and uses the same architectural design as the BERT pre-trained model, which is based on the Transformer architecture. Additionally, it was pre-trained by using more than 100 languages, which could address the limitation of traditional neural networks due to the fact that they are primarily focused on processes in English. Many non-native speakers combine English with other languages to complete the sentence, XLM-RoBERTa is able to understand and process such multilingual inputs without translating them. For instance, the given sentence 'The dog is active und er ist groß und glücklich' XLM-RoBERTa could accurately identify that 'The dog is active' is in English, while 'und er ist groß und glücklich' is in Deutsch, then could do the process on it for fine-tuning it.

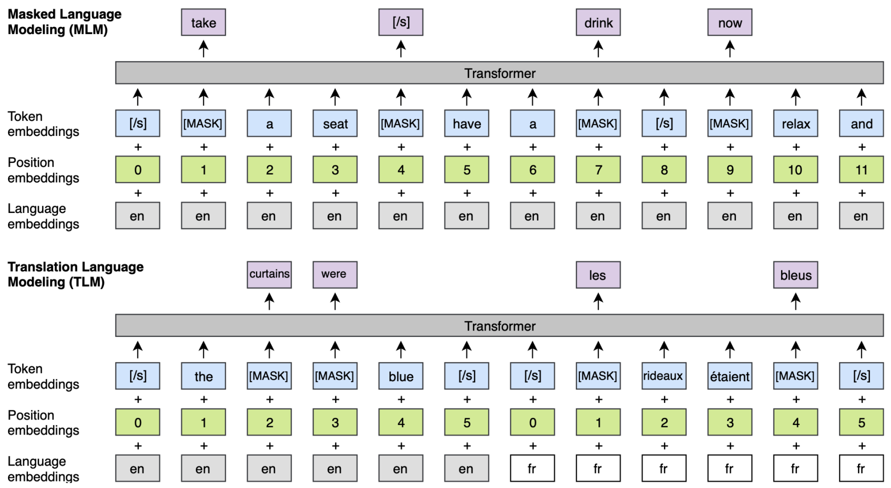

Masked Language Modeling (MLM)
take
[/s]
drink
now
Transformer
Token embeddings
[/s] [MASK] a seat [MASK] have a [MASK] [/s] [MASK] relax and
Position embeddings
0 1 2 3 4 5 6 7 8 9 10 11
Language embeddings
en en en en en en en en en en en en
Translation Language Modeling (TLM)
curtains
were
les
bleus
Transformer
Token embeddings
[/s] the [MASK] [MASK] blue [/s] [/s] [MASK] rideaux étaient [MASK] [/s]
Position embeddings
0 1 2 3 4 5 0 1 2 3 4 5
Language embeddings
en en en en en en fr fr fr fr fr fr

Figure 5: XLM-RoBERTa handle multi-lingual within the same sentence (Lample & Conneau, 2019)

To achieve Cross-lingual Language Models, the XLM-RoBERTa implements these models during the pre-training stage as

1. Causal Language Modeling (CLM): This model will make predictions on the probability of the next word in the sentence by considering the previous words, which is using the concept of Recurrent Neural Network (RNN) as P(wt—w1, w2, . . . , wt-1, θ). However, the XLM-RoBERTa is using the Transformer architecture to design this model even RNN is good (Lample & Conneau, 2019).

2. Masked Language Modeling (MLM): This model will randomly select fifteen percent (typically ratio) of the token from the given sentence or corpus and replace them with the special token as [MASK], then the model will make predictions the original words of those masked words by using the surrounding words in the given text. This step helps the model learn the attention to the tokens as shown in Figure 3 (Lample & Conneau, 2019).

3. The limitations of Causal Language Modeling and Masked Language Modeling are unsupervised and can only perform on those sentences that only contain a single language. To overcome the limitation, Translation Language Modeling (TLM) had been developed for improving the model

able to cross-lingual pre-training stage, which is an extension of MLM. It will also assign the special token as [MASK] to fifteen percent of the token within the sentence, but the model will also attend on those words surrounding English or French translation to make predictions on the masked token as shown in Figure 5 (Lample & Conneau, 2019).

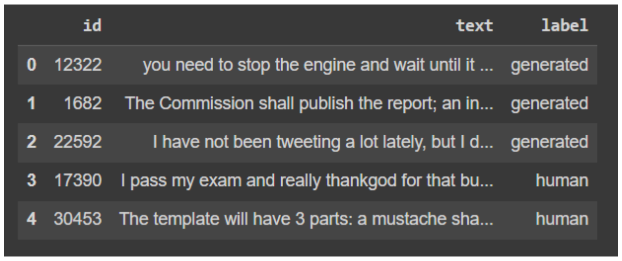
The preprocessing of those given sentences before fitting into the XML-RoBERTa model is the same as BERT, which will be needed to utilize the pre-trained model's tokenizer known as XLMRoberta-Tokenizer. This tokenizer will also compute the attention score and word embedding weight of each token and the special token but it will take into consideration the language of each token within the sentence (XLM-RoBERTa, n.d.).

After preprocessing on those given sentences, it will be ready to fit into the XLM-RoBERTa model for fine-tuning purposes as letting the model become more specific to the downstream tasks depends on the use case. In this study, the team will use the searched hyperparameter to fine-tune the XLM-RobertaForSequenceClassification model for classifying the label for both subtasks.

3 Experimental setting

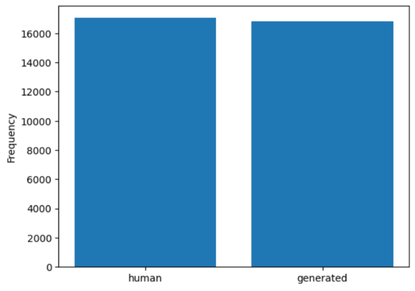
3.1 Structure and statistic of dataset

This study will focus on the English challenge for both subtasks and the English dataset consists of 2 parts:



	id	text	label
0	12322	you need to stop the engine and wait until it ...	generated
1	1682	The Commission shall publish the report; an in...	generated
2	22592	I have not been tweeting a lot lately, but I d...	generated
3	17390	I pass my exam and really thankgod for that bu...	human
4	30453	The template will have 3 parts: a mustache sha...	human

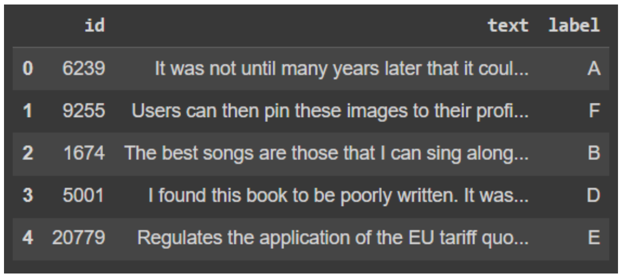
(a) First five rows of training data



(b) Classes distribution

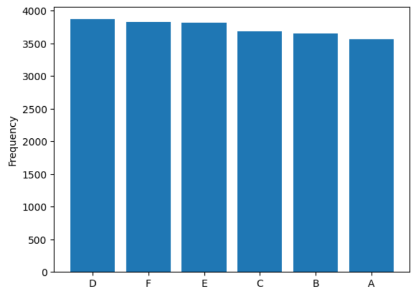
Figure 6: Structure and statistic of subtask 1's training data

For subtask 1, the team has to utilize the deployed models detect whether the text is automatically generated or not based on the text provided. This subtask is a binary classification task with 2 classes which are labeled as 'generated' and 'human', and both classes are balanced.



	id	text	label
0	6239	It was not until many years later that it coul...	A
1	9255	Users can then pin these images to their profi...	F
2	1674	The best songs are those that I can sing along...	B
3	5001	I found this book to be poorly written. It was...	D
4	20779	Regulates the application of the EU tariff quo...	E

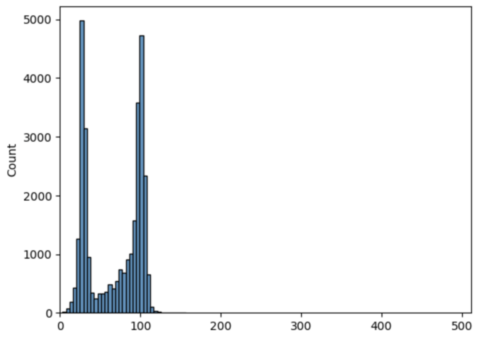
(a) First five rows of training data



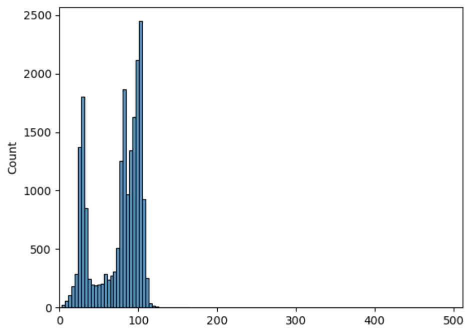
(b) Classes distribution

Figure 7: Structure and statistic of subtask 2's training data

For subtask 2, the team has to utilize the deployed models to identify which text generation model is used to generate the given automatically generated text. This subtask is a multiclass classification with 6 classes denoted as (A, B, C, D, E, and F), where each class represents a text generation model and six classes are balanced.



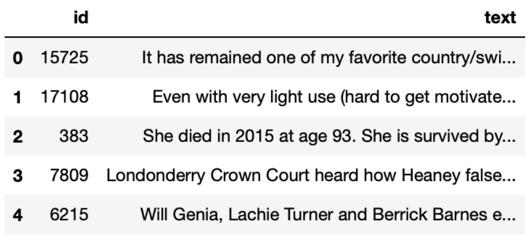
(a) Subtask 1



(b) Subtask 2

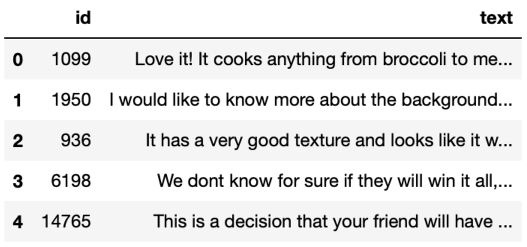
Figure 8: Histogram of the token length of the training data's texts after tokenization for both subtasks

Based on the illustration shown above, the result returned that the input_ids of the token lengths fall within the range of [0, 128] for both subtasks, which means the length of input sentences are falling within this range without inserting the special token as [PAD] until the maximum acceptable length of the model. Therefore, the team could set the maximum length for the tokenization to be 128, which resulted in saving more computation time and space.



	id	text
0	15725	It has remained one of my favorite country/swi...
1	17108	Even with very light use (hard to get motivate...
2	383	She died in 2015 at age 93. She is survived by...
3	7809	Londonderry Crown Court heard how Heaney false...
4	6215	Will Genia, Lachie Turner and Berrick Barnes e...

(a) Subtask 1



	id	text
0	1099	Love it! It cooks anything from broccoli to me...
1	1950	I would like to know more about the background...
2	936	It has a very good texture and looks like it w...
3	6198	We dont know for sure if they will win it all,...
4	14765	This is a decision that your friend will have ...

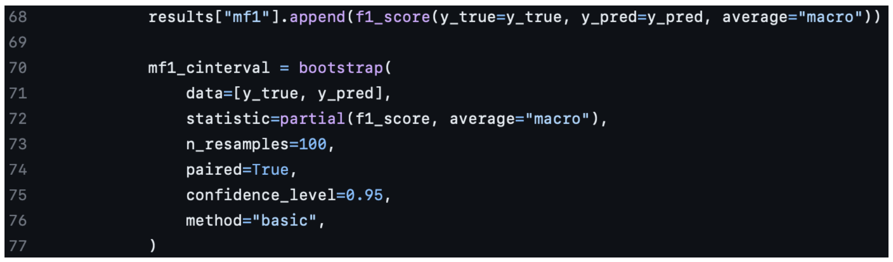
(b) Subtask 2

Figure 9: Testing set for both subtasks

As the above illustration, that is the testing set provided by the organizer and that only consists of its text and ids, which only require participants to make predictions on it. Participants have to submit the predicted result of each sentence to the organizer then they will only use the evaluated form using the Macro F1 score provided by them to evaluate the deployed model's performance.

3.2 Evaluation measures

To evaluate the performance of the implemented models, the team utilized the evaluation scripts provided by the AuTexTification organizer, which was using the Macro F1-Score and Macro F1 Confidence Interval (CI) as the range of the score would be within it (Autextification, 2023). In addition, the team would also utilize the accuracy to evaluate the performance of the models.



```
            results["mf1"].append(f1_score(y_true=y_true, y_pred=y_pred, average="macro"))

            mf1_cinterval = bootstrap(
                data=[y_true, y_pred],
                statistic=partial(f1_score, average="macro"),
                n_resamples=100,
                paired=True,
                confidence_level=0.95,
                method="basic",
            )
```

Figure 10: Evaluation Script (Autextification, 2023)

Since the provided testing set does not include the label of each sentence, therefore, the team splitted the training set into three different sets training, validation, and testing (used to evaluate the performance of deployed models by the team before the organizer released the leader board). The training and validation set will be used to execute the process of hyperparameter optimization and fine-tune the deployed models, and the testing will be only used to test the performance of the models.

4 Evaluation and discussion of the results

After fine-tuning the BERT and XLM-RoBERTa models, the team would utilize the evaluation script provided by the organizer to evaluate the performance of the model. The team would list down those generated results of each model in both subtasks.

4.1 Discussion of result

For the first subtask, these deployed models have to successfully predict those sentences' labels and the performance of BERT and XLM-RoBERTa listed below as:

Fine-tuned model	**Precision**	**Recall**	**Macro F1**	**CI of Macro F1**
BERT (Without preprocessing)	0.7842	0.9324	0.8519	(0.8381, 0.8641)
BERT (With preprocessing)	0.7813	0.9333	0.8506	(0.8385, 0.8662)
XLM-RoBERTa (Without preprocessing)	0.8466	0.9479	0.8944	(0.8857, 0.9044)
XLM-RoBERTa (With preprocessing)	0.7776	0.9426	0.8522	(0.8426, 0.8618)
BERT (leader board)	-	-	0.5559	(0.5503, 0.5620)

Table 1: Performance Metrics of Fine-tuned Models with different scenarios for subtask 1

Based on the result of these two deployed models, the team found that the XLM-RoBERTa model without translating foreign languages and converting emoji and emoticons into English and not removing stopwords, and executing the process of lemmatization has the best performance in subtask 1. As the result showed, those sentences were being removed stopwords, process lemmatization, translating those foreign languages, and converting those emojis and emoticons into English will cause the sentences to lose information on detecting the sentence was generated by text generation models or human-written and lead the F1 score to become lower even spending much time on translating them that is not worth. The returned result from both models in different preprocessing scenarios were performing well on this task since their Macro F1 score is really high and above 0.85 when the team was evaluating on our side by using the test set that split from the provided training set and didn't train the model. However, the organizer was releasing the leader board as 0.5559 for the team, which could be considered as poor performance on the BERT model.

For the second subtask, these deployed models have to successfully predict those sentences' labels and the performance of BERT and XLM-RoBERTa listed below as:

Fine-tuned model	**Precision**	**Recall**	**Macro F1**	**CI of Macro F1**
BERT (Without preprocessing)	0.6993	0.4280	0.5310	(0.5129, 0.5470)
BERT (With preprocessing)	0.7017	0.4257	0.5299	(0.5113, 0.5497)
XLM-RoBERTa (Without preprocessing)	0.7016	0.4226	0.5275	(0.5101, 0.5472)
XLM-RoBERTa (With preprocessing)	0.6859	0.4272	0.5265	(0.5079, 0.5468)
BERT (leader board)	-	-	0.5352	(0.5220, 0.5484)

Table 2: Performance Metrics of Fine-tuned Models with different scenarios for subtask 2

Based on the result of these two deployed models, the team found that the BERT model without translating foreign languages and converting emoji and emoticons into English has the best performance in subtask 2. Both models in both scenarios were not performing well due to the Macro F1 score being nearly approaching 0.5. Based on the evaluated result, the team could understand that both models are not good at handling classifying those sentences into six different text generation models.

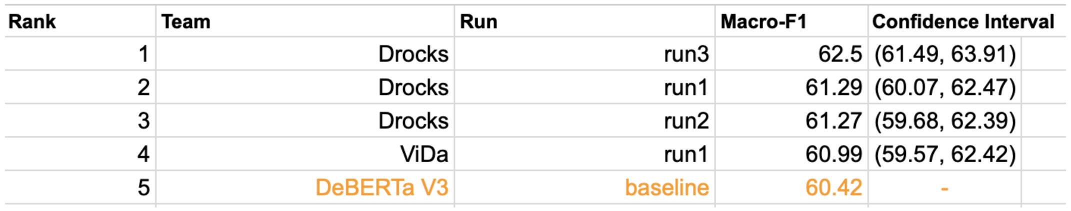
4.2 Results Comparison with Existing Solution

In the paper of (Ippolito et al., 2020), the authors also deployed the BERT model for executing the task same as subtask 1 for classifying the sentences generated by text generation model or human-written and performing fine-tuning on a large dataset extracted from popular web pages. That would make the model have more records to learn the patterns from a large set of records and be more professional on the specific downstream task. Furthermore, the authors implement the top-k decoding strategy, as mentioned in their paper, to enhance the model that does not utilize those lower likelihood tokens to fine-tune the model to increase its performance of the model. This strategy is a random sampling method to reduce the length of generated sentences and encourages pure summaries compared to greedy decoding strategies (Radford et al., 2018).

Fine-tune model	Accuracy
BERT (Top-k)	0.901

Table 3: Result from the paper of (Ippolito et al., 2020)

For subtask1 in this study, the best-performed XLM-RoBERTa pre-trained model and achieved an accuracy of 0.8505 on the validation set. However, this performance was lower compared to the referenced paper. One potential reason could be the size of the training dataset being used in the study. The authors of the paper had obtained and retrieved a large dataset from popular web pages, which allows their model to learn from more records to discover the patterns. In our study, the team has only 33,845 rows of sentences for fine-tuning, which could have limitations for the team to effectively improve the model's performance.



Rank	Team	Run	Macro-F1	Confidence Interval
1	Drocks	run3	62.5	(61.49, 63.91)
2	Drocks	run1	61.29	(60.07, 62.47)
3	Drocks	run2	61.27	(59.68, 62.39)
4	ViDa	run1	60.99	(59.57, 62.42)
5	DeBERTa V3	baseline	60.42	-

Figure 11: Leader board for subtask 2

For subtask 2, other participants who had also participated in the AuTexTification competition could generate better performance compared to the team and even above the baseline by using DeBERTa pre-trained model that was achieved by the organizer as having 0.625 Macro F1-score. That is really out-performing the team, and those participants have not released what kind of models that they deployed and implemented methods on preprocessing.

5 Conclusion

5.1 Key Takeaways

In this study, the team deployed BERT and XLM-RoBERTa pre-trained models in two different scenarios: non-preprocessing on those sentences by just using pre-trained models' tokenizers and executing the technique of lemmatization on those sentences translating foreign languages and converting emoji and emoticons into English. Based on the result, the team found that the Macro F1-score decreased when performing translation and conversion. This finding is interesting because the translation and conversion processes on those given sentences could lead the models to lose information on those sentences and more consume time on doing preprocessing. Based on this study, the team concluded that retain in the original language, emojis, and emoticons allowed the models to make more accurate predictions due to humans often using their native languages to encounter those words that they are not familiar with in English and using emoji or emoticons to express their emotions since those pre-trained models are using the Transformer architecture to give attention weight and word embedding on those tokens. Therefore, it could know how well the foreign languages and emoji or emoticons are belonging to the human-written label.

5.2 Future Direction

For future enhancement, the team decided not to implement those well-pre-trained models such as BERT or XLM-RoBERTa models for this project. These models do not really fit with our use case as detecting those sentences are generated by the text generation model or human-written, as the end-user could only fine-tune the model to specify the downstream task that the model should do and predict in the upcoming task. Therefore, the team would train and create a model and tokenizer that would utilize the Transformer architecture to design from scratch and use our own data to train it, which could allow the model to become more narrow and specific on the task (Wolf et al., 2020).

References

[1] AuTexTification. (2023). https://sites.google.com/view/autextification/home?authuser=0

[2] Fine-tuning with custom datasets — transformers 3.2.0 documentation. (n.d.). https://huggingface.co/transformers/v3.2.0/custom_datasets.html

[3] Trainer. (n.d.). https://huggingface.co/docs/transformers/main_classes/trainer

[4] Probst, P., Boulesteix, A.-L., & Bischl, B. (2019). Tunability: Importance of Hyperparameters of Machine Learning Algorithms. Journal of Machine Learning Research, 20, 1–32. https://jmlr.org/papers/volume20/18-444/18-444.pdf

[5] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2018). Language Models are Unsupervised Multitask Learners. https://life-extension.github.io/2020/05/27/GPT

[6] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf

[7] Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the Dangers of Stochastic Parrots. https://doi.org/10.1145/3442188.3445922

[8] Massarelli, L., Petroni, F., Piktus, A., Ott, Rocktäschel, T., Plachouras, V., Silvestri, F., & Riedel, S. (2020). How Decoding Strategies Affect the Verifiability of Generated Text. https://arxiv.org/pdf/1911.03587.pdf

[9] Dugan, L., Ippolito, D., Kirubarajan, A., & Callison-Burch, C. (2020). RoFT: A Tool for Evaluating Human Detection of Machine-Generated Text. https://arxiv.org/pdf/2010.03070.pdf

[10] Real or Fake Text. (n.d.). www.roft.io. http://www.roft.io

[11] Camacho-Collados, J., & Pilehvar, M. (2018). On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. https://arxiv.org/pdf/1707.01780.pdf

[12] Yuqiao, W., Ye, W., Zhao, M., & Wattenhofer, R. (2021). Emoji for Natural Language Processing. https://pub.tik.ee.ethz.ch/students/2020-HS/MA-2020-31.pdf

[13] Wu, S., & Dredze, M. (2020). Are All Languages Created Equal in Multilingual BERT? https://doi.org/10.18653/v1/2020.repl4nlp-1.16

[14] Alammar, J. (2018). The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer/

[15] Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. https://arxiv.org/pdf/1706.03762v5.pdf

[16] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/pdf/1810.04805.pdf

[17] Lample, G., & Conneau, A. (2019). Cross-lingual Language Model Pretraining. https://arxiv.org/pdf/1901.07291.pdf

[18] XLM-RoBERTa. (n.d.). https://huggingface.co/docs/transformers/model_doc/xlm-roberta

[19] Autextification. (2023). AuTexTificationEval/evaluate_submissions.py at main · autextification/AuTexTificationEval. GitHub. https://github.com/autextification/AuTexTificationEval/blob/main/evaluate_submissions.py

[20] Ippolito, D., Duckworth, D., Callison-Burch, C., & Eck, D. (2020). Automatic Detection of Generated Text is Easiest when Humans are Fooled (pp. 1808–1822). Association for Computational Linguistics. http://aclanthology.lst.uni-saarland.de/2020.acl-main.164.pdf

[21] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2018). Language Models are Unsupervised Multitask Learners. https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[22] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., & Drame, M. (2020). Transformers: State-of-the-Art Natural Language Processing (pp. 38–45). https://aclanthology.org/2020.emnlp-demos.6.pdf

6 Appendix

1. Code for the project: https://github.com/KokTeng00/AuTexTification_Challenge
2. Leader board for the AuTexTification challenge: https://sites.google.com/view/autextification/resu lts?authuser=0
3. Access link to download training set: https://zenodo.org/record/7692961?token=eyJhbGciOiJIUzU xMiIsImV4cCI6MTY4Mjg5MTk5OSwiaWF0IjoxNjgwMjc2NzY3fQ.eyJkYXRhIjp7InJlY2lkIjo3Njk yOTYxfSwiaWQiOjMxNTAxLCJybmQiOiJiMmJkNzViMSJ9.UzfQsrtuBGJ6Ev0uBP0IazQsLvcRb 1HrHrYb2aAIe4kJN3kn6HiHoJy-2LsPHt2og_TsIDFJFpPdZ75tE6_aVA
4. Access link to download testing set: https://zenodo.org/record/7846000?token=eyJhbGciOiJIUzUxM iIsImV4cCI6MTY4NDcwNjM5OSwiaWF0IjoxNjgyMDcwMzE1fQ.eyJkYXRhIjp7InJlY2lkIjo3ODQ2 MDAwfSwiaWQiOjMyMjk0LCJybmQiOiI1NDIyODZhMyJ9.hZH0uMRd5iDmhC3IcVyEDUHOnmI Tlxb27N5dxY0dUi-yK4zzxn-4dD1wokSgJMRbZsY-wdrMUOhcefJ77FhdBA