DETI - 2023/2024

Identify text written by AI

Report 2

Diogo Oliveira Magalhães	102470
Leonardo Almeida	102536
Pedro Henrique Figueiredo Rodrigues	102778

Teoria Algorítmica da Informação

Professores: Armando Pinho e Diogo Pratas

May 8, 2024

Contents

Chapter 1

Introduction

In recent years, the advancement of artificial intelligence has revolutionised numerous industries. Healthcare, aviation, agriculture, and financial services have all undergone big changes with the rise of AI. Although most applications of AI have a positive impact on society, there are concerns about the potential misuse of this technology. One of the main concerns is related to AI-generated content, which can be in various forms, from news articles and social media to creative works such as photography and design. The misuse of AI-generated content can have a serious impact on society, as it can be used to spread misinformation, manipulate public opinion, and even commit fraud.

In order to face this challenge, the research community has been working on developing methods capable of distinguishing between human-generated and AI-generated content, more specifically text. One of the most popular approaches to this problem is to train complex machine learning models on large datasets of human and AI-generated text so that they can learn how to differentiate between the two. Although these models have shown promising results, they present several drawbacks, such as high computational cost, the need for large amounts of training data, and a considerably large input text to produce viable results.

In the context of "Teoria Algorítmica da Informação" (Algorithmic Information Theory) course, as part of the Master's degree in Computer Science and Engineering at the University of Aveiro, we were challenged to implement and study a classifier capable of distinguishing between human-rewritten and AI-rewritten text using finite-context models. The main goal of this project is to create two models that are "good descriptions" of the text, one for human text and another for AI text, and then use them to classify a target text as human-rewritten or AI-rewritten. This final decision will correspond to the model that requires fewer bits to encode the target text.

Figure 1.1 illustrates the main idea of the solution to implement and its components.

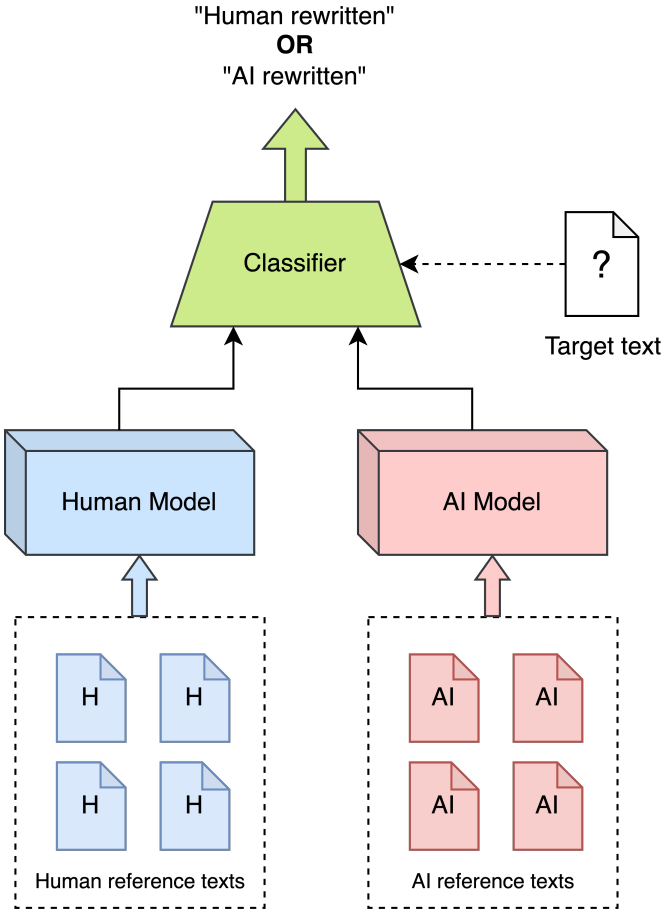


Figure 1.1: Overview of the classifier based on finite-context models

The following chapters will present the finite context models, the strategy and implementation of the solution, the datasets used, the parameter analysis, the results obtained, and the conclusions drawn from this work.

Chapter 2

Finite context models

To understand how finite-context models can be used to distinguish if a text was rewritten by a human or an AI, first, it is crucial to understand what a Markov model is and how its basic principles can be applied to text classification.

In probability theory, a Markov model is a stochastic model used to model pseudo-randomly changing systems that assume the Markov property. The Markov property states that the probability of the system transitioning to any particular state is dependent solely on the current state, not on the sequence of events that preceded it. This property is often referred to as memorylessness. Markov models are used in a wide range of applications, including speech recognition, handwriting recognition, bioinformatics, and more.

For some applications, a simple Markov model may not be sufficient to capture the complexity of the system being modeled. In such cases, higher-order Markov models can be used. A higher-order Markov model considers the previous k states to determine the probability of transitioning to the next state, allowing a detailed representation of the system's dynamics.

For illustration purposes, let us consider that the output until a certain instant t of a system with states A, B, C, and D is as follows: *ABBAC*. Using a first-order Markov model, the probability of the next state being B only depends on the previous state, which means *P(B|ABBAC) = P(B|C)*. On the other hand, in a Markov model with order 3, the probability of transitioning to state B would depend on the previous three states, *P(B|ABBAC) = P(B|BAC)*.

In general, considering the sequence of outputs x_1, x_2, x_3 ..., x_n, the probability of transitioning to state x_n in a Markov model of order k is given by $P(x_n|x_{n-1}, x_{n-2}, ..., x_{n-k})$.

In lossless data compression, the most common approach is to use finite-context models, which are a type of Markov model that considers a finite number of previous symbols to estimate the probability of the next symbol. Finite-context models are used to capture the statistical properties of the data source and assign a probability estimate to each symbol in the alphabet based on a finite context of previous symbols.

In this project, for each reference text, the base idea is to collect counts that represent the number of times each symbol occurs in each context. Using these counts, we can determine the probability of a symbol occurring in a specific context, and consequently, the number of bits required

to encode that symbol. The model that represents the target text with fewer bits is considered to be the most likely source of the text.

Chapter 3

Strategy and Implementation

To achieve the goal of distinguishing between human and AI-generated text, we decided to separate the problem into two main tasks: model creation and classification. This way, the same model can be used multiple times to classify different texts, without the need to retrain it every time.

3.1 Model creation

For the model creation task, we created a program *train.cpp* with the following arguments:

- **-h**: The path to the human reference text file.
- **-g**: The path to the AI-generated text file.
- **-o**: The path to the output folder where the models and configurations will be saved.
- **-a**: The file with the alphabet to be used in the model, if not provided, the program will generate the alphabet based on the input texts.
- **-k**: The order of the Markov model to be used.

The program starts by evaluating the alphabet to be used in the model. Then, it saves the alphabet used, its number of symbols and the **k** value in a *.json* file in the output folder because these hyperparameters are essential to ensure that the model is correctly loaded in the classification task. Finally, the program creates a model for human and AI-generated texts, saving them in the output folder in a human-readable format.

To create the model, we use a *unordered_map* to store the counts of each symbol in each context. The key of the map is a string that represents the context, and the value is an array with the counts of each symbol in that context. To know which position in the array corresponds to each symbol, we create a mapping between the symbol and its index with another array of size 256. This way, to know the index of a symbol *s*, we access the mapping array with the ASCII value of *s* as the index. Symbols that are not in the alphabet are ignored. This workflow is illustrated in Figure 3.1.

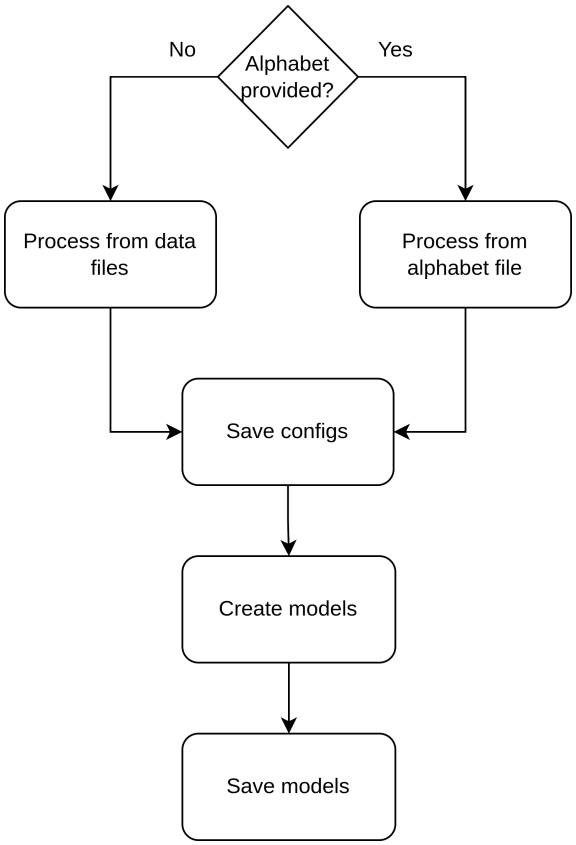


Figure 3.1: Model creation workflow.

3.2 Classification

For the classification task, we created a program *was_chatted.cpp* with the following arguments:

- **-m**: The path to the models folder.
- **-d**: The data file to evaluate (1 sample per line), if not provided, the iterative mode is enabled.
- **-a**: The smoothing factor (alpha) value to be used to calculate the probability of a symbol in a context.
- **-v**: To enable verbose mode.

The program starts by loading the models and configurations from the models folder. Then, there are two possible modes of operation: iterative and from a file. In the iterative mode, the program reads a sample from the standard input, classifies it, and prints the result. In the other mode, the program reads the samples from the file, classifies them, and prints the results. This workflow is illustrated in Figure 3.2.

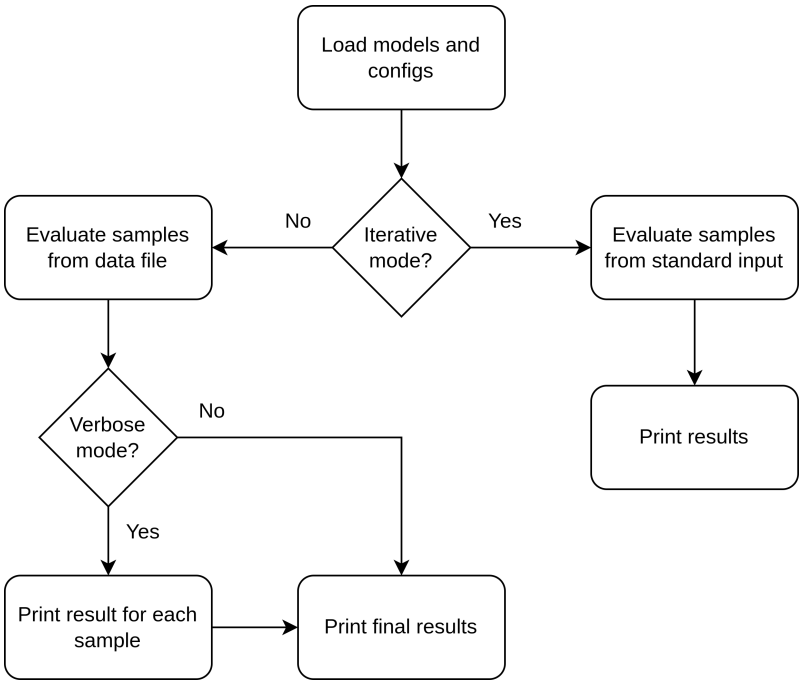


Figure 3.2: Classification workflow.

As mentioned in Chapter 2, to classify a text, we need to estimate the number of bits required to encode it using the human and AI-generated models and choose the model that requires fewer bits. This number of bits is calculated by

$$\sum_{i=1}^{n} -\log_2(P(x_i|x_{i-1},x_{i-2},...,x_{i-k})) \tag{3.1}$$

where x_i is the symbol at position i in the text, and n is the length of the text. The probability of a symbol in a context is given by

$$P(x_i|x_{i-1},x_{i-2},...,x_{i-k}) \approx \frac{count(x_{i-1},x_{i-2},...,x_{i-k},x_i)+\alpha}{\sum_{j=0}^{alphabet\_size} count(x_{i-1},x_{i-2},...,x_{i-k},j)+\alpha*alphabet\_size} \tag{3.2}$$

where *count* is the number of times a symbol appears in the model for a given context, *alpha* is the smoothing factor, and *alphabet_size* is the number of symbols in the alphabet.

Since the models only accept symbols from the alphabet used in the training, the other symbols in the samples to classify are ignored.

Chapter 4

Datasets

Similarly to what happens in machine learning, in this case, to create models that are a good description of human and AI text, it is crucial to use good datasets that are representative of each class. In this project, we used two public datasets available on Hugging Face Datasets repository: the "HC3"[1] dataset and the "AI-human-text"[2] dataset.

The HC3 (Human-ChatGPT3) dataset was introduced in [1] where the authors conducted comprehensive human evaluations and linguistic analysis of ChatGPT-generated content compared with that of humans. The authors also conducted an extensive analysis on how to effectively detect whether a certain text is generated by ChatGPT or humans. Regarding the dataset itself, this dataset is composed of 24 321 samples and each sample is formed by three features: a question, a list of human answers, and a list of ChatGPT answers. These questions were extracted from several online sources, including online forums, and general question-answering websites.

For this project, we needed to pre-process the HC3 dataset to create a dataset that was suitable for the task at hand. As we did not have an interest in the questions themselves, we decided to extract the different answers and consider them as individual samples. After performing this extraction, we discarded all the repeated texts and samples with lengths smaller than 100 characters. Finally, to guarantee that the dataset was balanced in terms of total number of characters, we discarded a specific number of samples of the class with more characters to make sure that the number of characters in each class was nearly the same. When balancing the data in terms of total number of characters, in the end, the number of samples for each class could be different, as human texts can have different lengths than AI texts. Given this, to measure the performance of the classifier, it is important to use metrics other than accuracy.

The "AI-human-text" dataset is a dataset of human and AI text with roughly 400k rows. This dataset is a processed version of a dataset available on Kaggle[3] and is composed of two features: the text and the label, which indicates if the text was generated by a human or an AI. Similarly to what we made with the HC3 dataset, we discarded all the repeated texts, and samples with lengths

[1] https://huggingface.co/datasets/Hello-SimpleAI/HC3
[2] https://huggingface.co/datasets/andythetechnerd03/AI-human-text
[3] https://www.kaggle.com/datasets/shanegerami/ai-vs-human-text

smaller than 100 characters, and balanced the dataset in terms of the general number of characters in each class.

To train the models, we split the datasets into training, validation, and test sets. The training set corresponds to 90% of the samples, the validation set corresponds to 10%, and the testing set to the remaining 10%. The training set was used to train the models, the validation set was used to find the best parameters (*k*, *alpha* and alphabet) for the models, and the test set was used to evaluate the classifiers' performance.

In this project, the collection of samples are considered the reference text used to train the model.

Chapter 5

Parameter Analysis

For the finite-context models to be effective in distinguishing between human and AI-generated rewritten text and have a good time performance, we need to make a study of what are the best values for the hyperparameters that will be used for generating and using the models.

In the section Strategy and Implementation 3, we can identify 3 hyperparameters that can be fine-tuned, being them:

k: The order of the Markov model.

alphabet: The alphabet used by the models for generating the tables and for classifying the target texts.

alpha: The smoothing factor.

The first two hyperparameters, (k) and (alphabet), were used in the creation of the models and the last hyperparameter (*alpha*) was used to classify the target texts.

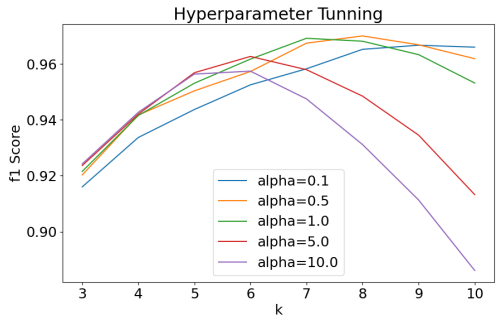
In order to determine the best values for the hyperparameters, we performed an exhaustive search where we varied the values of each parameter and analysed the performance of the algorithm using as metrics the f1 score, as the number of samples for each type was unbalanced, and the time performance. For this, we followed a systematic approach where we first tested a combination of different values for the hyperparameters k and *alpha* using grid search and then, already using the selected values for those hyperparameters, we tested different values for the alphabet. All of these tests and fine-tuning were done using the validation set of the datasets to ensure that the models were not using previously seen data to make the predictions.

The values tested for the first step were:

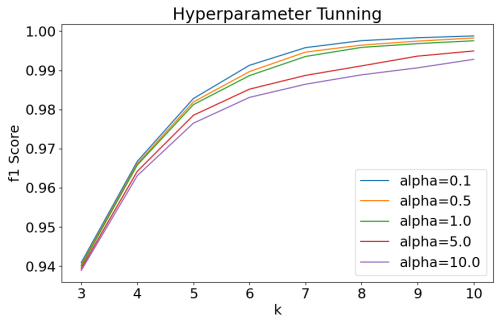
k: [3, 4, 5, 6, 7, 8, 9, 10]

alpha: [0.1, 0.5, 1, 5, 10]

Figure 5.1 shows the results of the metric f1 score for the grid search of the hyperparameters k and *alpha* using both datasets.



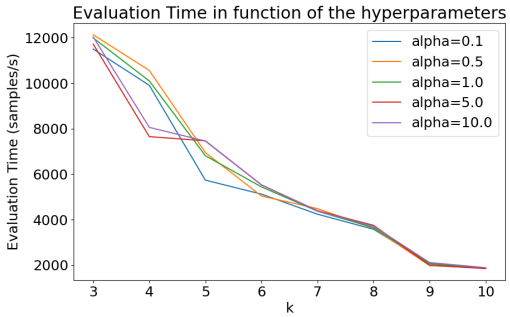
(a) Grid search results for the dataset HC3



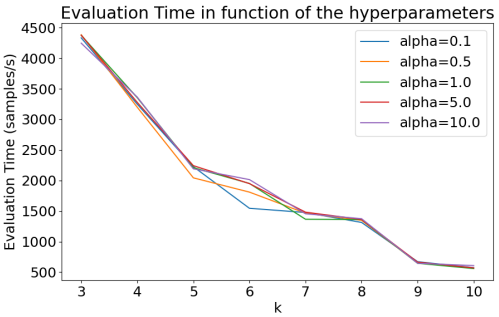
(b) Grid search results for the dataset AI-human-text

Figure 5.1: F1 score for the grid search of the hyperparameters *k* and *alpha* for the datasets HC3 (On the left) and AI-human-text (On the right)

Not only from hits and misses is a model evaluated, but also from its time performance. Figure 5.2 shows the time performance for the grid search of the hyperparameters *k* and *alpha* using both datasets.



(a) Time performance for the dataset HC3



(b) Time performance for the dataset AI-human-text

Figure 5.2: Time performance for the grid search of the hyperparameters *k* and *alpha* for the datasets HC3 (On the left) and AI-human-text (On the right)

After an analysis of the results of both tests, we ended up selecting as hyperparameters *k* and *alpha* the values 8 and 0.5, respectively. The reasons for the selection of these values were:

- The *k* value was selected because it was the value that was almost always near the maximum, or even at the maximum, f1 score performance while not being too slow in terms of time performance. Starting from $k = 9$, the time performance of the algorithm started to decrease significantly, and the f1 score performance was not significantly improving and was at times, especially in the dataset HC3, decreasing.

- The *alpha* value was selected because it was the value that was almost always near the maximum, or even at the maximum, f1 score performance, especially in the dataset HC3 where for lower values of *k* the performance was significantly better. The time performance of the algorithm was not significantly affected by any value of *alpha*.

Using these parameters we were able to test several alphabets and check which was better. The values tested for the second step were:

alphabet 1: `"1234567890 abcdefghijklmnopqrstuvwxyz"`

alphabet 2: `"1234567890 abcdefghijklmnopqrstuvwxyz.,!?'\"\"/\\;:_-"`

alphabet 3: `"1234567890 abcdefghijklmnopqrstuvwxyz.,!?'\"\"/\\;:_-@#$%^&*()[]{}<>"`

alphabet 4: `" abcdefghijklmnopqrstuvwxyz"`

Figure 5.3 shows the results of the metric f1 score for the search of the alphabet using both datasets.

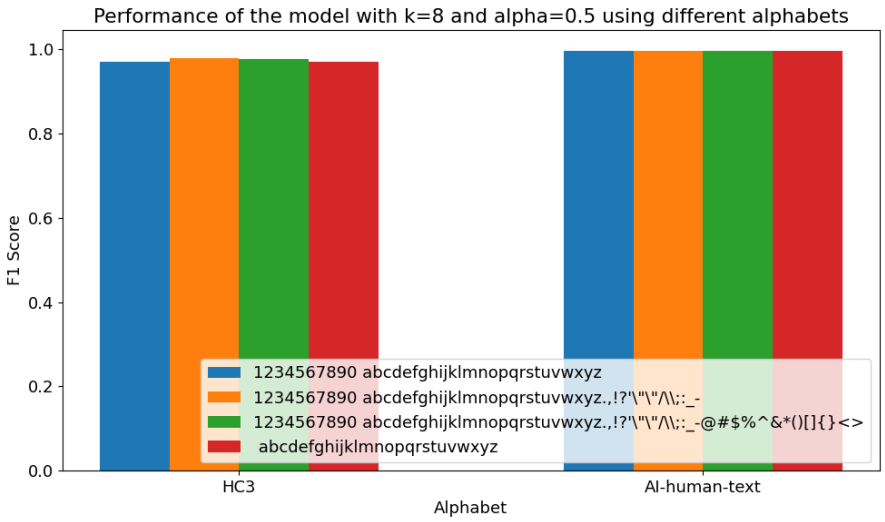


Figure 5.3: Comparison of the F1 score metric for the search of the alphabet for the datasets HC3 (On the left) and AI-human-text (On the right)

From the results obtained, we can see that on the dataset HC3 the alphabet 2 was the one that performed better even though it was by a very small margin. On the other hand, we can see that on the dataset "AI-human-text" the usage of the different tested alphabets did not have any impact on the performance of the algorithm, this is due to this dataset having a low number of punctuation symbols and special characters. With this in mind, we decided to use the alphabet 2 for both datasets.

In the previous steps, we have selected the same values for the hyperparameters *k*, *alpha* and *alphabet* for both datasets. This decision was made as we saw that the performance gain of the algorithm was similar for both datasets. This way, we tried to ensure that we are generating more general models that will be able to be trained with different datasets and still perform, not overfitting, and not needing to redo a fine-tuning of the hyperparameters.

Chapter 6

Results

Using the best parameters previously found, this chapter presents a comprehensive analysis of the results obtained by the classifier and how its performance changes in different scenarios. The main goal is to understand the relationship between the length of the reference texts used to train the models and the performance of the classifier when classifying the target texts. Moreover, we also study the impact of the size of the target texts on the algorithms' performance.

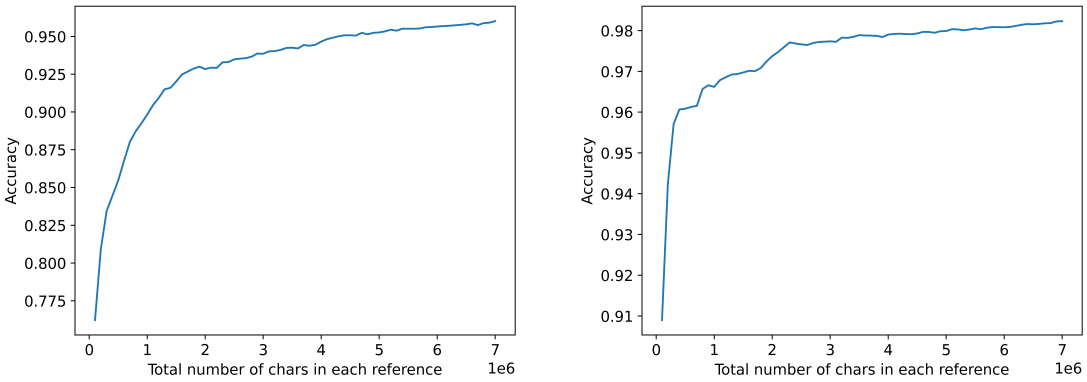
6.1 Influence of reference length

Since the performance of the classifier heavily depends on the reference texts used to train the models, we conducted an analysis to understand how the length of the references used affects the classifier's performance. With that in mind, we started by fixing the best parameters found in the last chapter and trained the models using different lengths for both references (different total number of characters in the reference). The lengths considered were from 100 thousand to 7 million characters, with increments of 100 thousand characters. Then for each classifier we determined its accuracy on the test set. Figure 6.1a and 6.1b illustrate the results obtained for the HC3 and AI-human-text datasets, respectively.

From both figures, it is clear that as the overall length of the reference text increases, the classifiers' performance also increases. This behaviour is expected since the models have more information to learn from and can better distinguish between human and AI text. However, it is also important to note that the performance tends to increase in a slower pace after a certain length. For the case of the HC3 dataset, the performance seems to decrease its growth rate after 2 million characters, while for the AI-human-text dataset, the performance seems to decrease its growth rate after roughly 3 million characters. Although the biggest growth is in the first one or two million characters, the classifier's performance seems to be always increasing at slow rates.

From these results, it is also clear that the performance of the models heavily depends on the training dataset used.

Figure 6.1: Accuracy in function of the overall length of the references



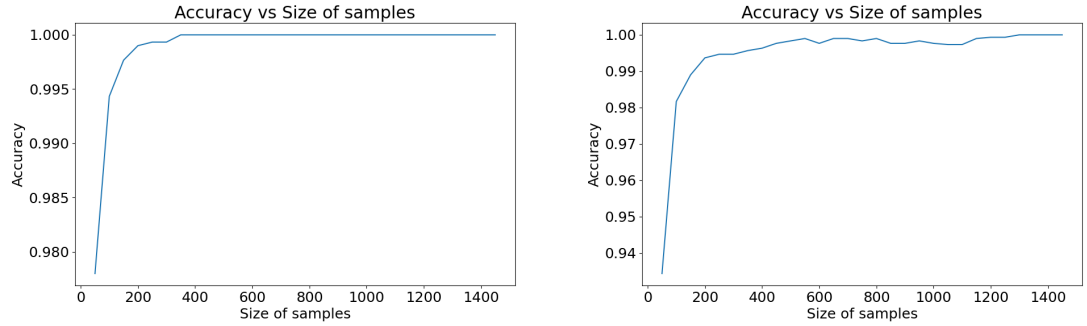
(a) Results for HC3 dataset

(b) Results for AI-human-text dataset

6.2 Influence of target sample

The algorithms' performance also depends on the target texts used to classify. To verify how the size of the target texts affects the classifiers' performance, we made an analysis where we fixed the best parameters found in the last chapter, trained the models, and then tested different target texts with different lengths.

In this analysis, we took 1500 samples of the test set, for each type of Human and AI, with lengths of at least and near 1500 characters. With these 3000 samples, we ran the classification algorithm and calculated its accuracy but only using the first Y characters of the target texts, with Y being the lengths considered for the target texts that were from 50 to 1500 characters, with increments of 50 characters. The results obtained for each dataset are illustrated in Figure 6.2.



(a) Results for HC3 dataset

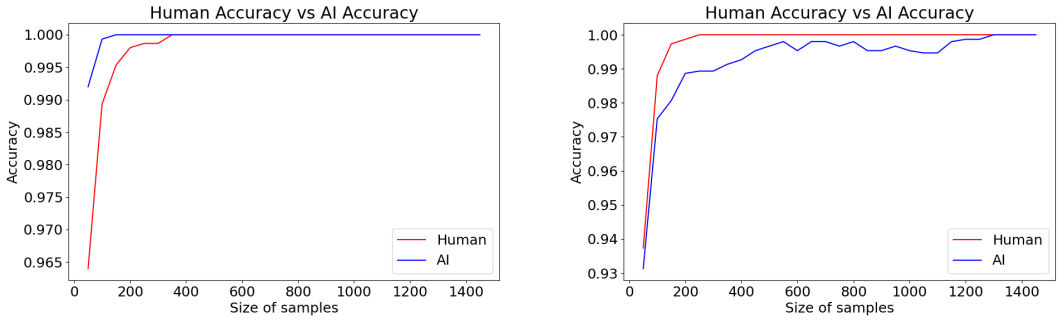
(b) Results for AI-human-text dataset

Figure 6.2: Accuracy in function of the length of the target texts for the datasets HC3 (On the left) and AI-human-text (On the right)

From the results in Figure 6.2, we can see that the classifier's performance keeps increasing as the number of chars of the target text also increases. This behaviour is expected since the models have more information to better distinguish between human and AI texts. After a certain length,

the performance tends to stabilise, and the growth rate decreases or even stops. For the case of the HC3 dataset, the algorithm's performance seems to stabilise its growth rate after 400 characters and for the AI-human-text dataset, the performance seems to stabilise its growth rate after 600 characters.

Additionally, we studied the impact of the size of the target texts on each of the types of texts separately. The results obtained are illustrated in Figure 6.2a.



(a) Results for HC3 dataset

(b) Results for AI-human-text dataset

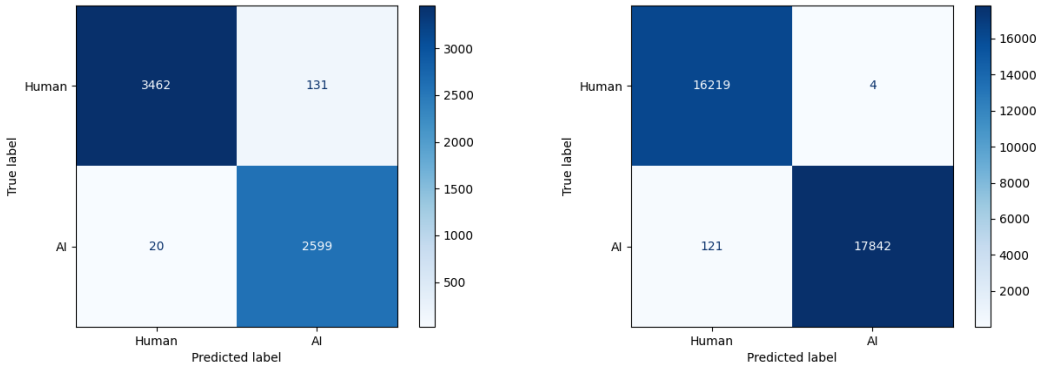
Figure 6.3: Accuracy in function of the length of the target texts for the datasets HC3 (On the left) and AI-human-text (On the right) for each type of text

From figure 6.3, we can see that different datasets might have an impact on the bias of the classifier. In the case of the HC3 dataset, the algorithm seems to be a bit more AI-biased, while in the case of the AI-human-text dataset, it seems to be a bit more human-biased. This is an interesting result that shows that the classifier is not biased by itself but by the dataset used to train it. Overall, this bias is almost irrelevant as the algorithm is able to classify the target texts with extremely high accuracy.

6.3 Classifier testing

To test the algorithm's performance we used the best parameters found in chapter 5 and used the test set of the datasets for the classification. The results obtained for each dataset are illustrated in the confusion matrices in Figure 6.4.

From the results in Figure 6.4, we can see that the classifier performed very well and had low quantities of false positives and false negatives for each type. This is an excellent result because it shows that the target texts are being classified with extremely high accuracy and F1 score.



(a) Confusion matrix for HC3 dataset (b) Confusion matrix for AI-human-text dataset

Figure 6.4: Confusion matrix for the datasets HC3 (On the left) and AI-human-text (On the right)

Table 6.1 shows the metrics results using the algorithm for each dataset. As we can see from the results in the table, the classifier performed very well, outperforming our expectations.

Dataset	**Accuracy**	**F1 score**
HC3	0.97569	0.97521
AI-human-text	0.99634	0.99633

Table 6.1: Results of the classifier for each dataset

6.4 Existent Solutions

Nowadays the existent solutions to face the problem of classifying the type of rewritten text are mostly based on Machine Learning models. Although these models have shown promising results, they present several drawbacks, such as:

- High computational costs in the training steps.
- The need for large amounts of data for training.
- A limitation on the number of tokens that are passed to the model at the same time for evaluation.
- An evaluation time that is considerably larger than using our model.
- Usually Black-box models. It is hard to understand why the model made a certain decision.

We found some machine learning models that used the same datasets as we did for training. Their results, as shown in the paper [1] as an example, were very similar to ours but they had to use more tricks to be able to achieve these results. In this specific example, they used some pre-defined sentences that were more likely to be AI-generated or human-generated to decide the type of the sample without even needing to use the model to make the prediction if any of these sentences were present in the sample.

Chapter 7

Conclusions

In this project, we implemented and studied a classifier capable of distinguishing between human-rewritten and AI-rewritten text using finite-context models. The main goal was to create two models that are "good descriptions" of the text, one for human text and another for AI text, and then use these models to classify a target text. To achieve this goal, we used two publicly available datasets. We pre-processed these datasets to create two reference texts that were suitable for this project.

Before analysing the behaviour and performance of the classifier, we performed a grid search analysis to find the best values for the hyperparameters k and *alpha*, and with those values, we tested several alphabets to find the one able to produce the best results. Using these values we then studied the performance of the classifier in different scenarios, such as varying the length of the reference texts and the length of the targets. From the results, we concluded that as the overall length of the reference text increases, the algorithms' performance also increases, but at a slower pace after a certain length. Similarly, its performance also increases with the length of the target texts, with shorter texts being harder to classify.

In terms of the final performance, for both datasets, the classifier outperformed our expectations, achieving an accuracy of over 97% on the test set of the HC3 dataset and over 99% on the test set of the AI-human-text dataset. These results show that finite-context models can be a promising approach to classify human and AI text. However, it is important to note that the performance of the classifier has a considerable dependency on the data used for training, meaning that when testing with text from a different dataset, the performance could decrease. Therefore, it is crucial to use representative datasets to achieve the best results. When compared to other ML-based approaches, the proposed classifier has the advantage of being transparent and not a complete black-box model and requires fewer computing power.

References

[1] Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection, 2023.