MOSAIC : Mélange d'experts pour la détection de textes artificiels

Matthieu Dubois[1] Pablo Piantanida[2,3] François Yvon[1]
(1) Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, Paris, France
(2) International Laboratory on Learning Systems (ILLS), Quebec AI Institute (MILA), CNRS, Québec, Canada
(3) CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvette, France
matthieu.dubois[at]sorbonne-universite.fr,
pablo.piantanida[at]mila.quebec, francois.yvon[at]sorbonne-universite.fr

RÉSUMÉ

La diffusion auprès du grand public de grands modèles de langue facilite la production de contenus nuisibles, médisants, malhonnêtes ou falsifiés. En réponse, plusieurs solutions ont été proposées pour identifier les textes ainsi produits, en traitant le problème comme une tâche de classification binaire. Les premières approches reposent sur l'analyse d'un document par un modèle détecteur, avec l'hypothèse qu'un faible score de perplexité indique que le contenu est artificiel. Des méthodes plus récentes proposent de comparer les distributions de probabilité calculées par deux modèles. Cependant, s'appuyer sur une paire fixe de modèles peut fragiliser les performances. Nous étendons ces méthodes en combinant plusieurs modèles et en développant une approche théoriquement fondée pour exploiter au mieux chacun d'entre eux.

ABSTRACT

MOSAIC : Multiple Observers Spotting AI Content

The dissemination of Large Language Models (LLMs) has made it easier for all to produce harmful, toxic, faked or forged content. In response, various proposals have been made to automatically discriminate artificially generated from human-written texts, framing the problem as a binary classification problem. Early approaches evaluate an input document with a detector LLM, assuming that low-perplexity scores reliably signal machine-made content. More recent systems instead consider two LLMs and compare their probability distributions over the document to further discriminate when perplexity alone cannot. However, using a fixed pair of models can induce brittleness in performance. We extend these approaches to the ensembling of several LLMs and derive a new, theoretically grounded approach to combine their respective strengths.

MOTS-CLÉS : Mélange d'experts, Génération de Textes, Détection de textes artificiels, Théorie de l'information.

KEYWORDS: Mixture of experts, Text Generation, Machine-generated text detection, Information theory.

ARTICLE : **Accepté à** Findings of ACL 2025.

1 Introduction

La fluidité et la diversité des textes produits par des machines ont considérablement progressé avec le développement de très grands modèles de langue. Avec la sortie de ChatGPT, OpenAI a suscité de vives discussions sur les nouvelles possibilités offertes par ces assistants d'écriture alimentés par intelligence artificielle (IA), et notamment sur les menaces que ceux-ci peuvent induire. En effet, la production de contenus toxiques, voire malhonnêtes (Crothers *et al.*, 2023), ou encore de désinformation (Zellers *et al.*, 2019) n'a jamais été aussi simple. Actuellement, les méthodes permettant de détecter l'origine d'un texte donné afin d'atténuer la diffusion de tels contenus ou encore d'empêcher les étudiants d'utiliser ces outils lors de leurs travaux semblent avoir du mal à suivre les rapides progrès de l'IA [1].

De nombreux travaux cherchent à identifier les textes produits par des modèles de langues afin de se prémunir contre les potentielles menaces mentionnées ci-dessus. En général, ces outils utilisent un modèle *détecteur*, afin de différencier entre des écrits provenant de modèles *générateurs* et des productions humaines. Cette tâche de classification se décline en de nombreuses versions, faisant varier granularité et type de supervision (voir Section 2). Étant donnée la popularité de ce modèle, la plupart des travaux sont concentrés sur ChatGPT, pour lequel il est aisé d'obtenir de grands volumes de données.

Notre Contribution. Dans cet article, nous présentons une nouvelle méthode d'ensemble (décrite Figure 1) motivée par les principes de compression universelle de la théorie de l'information. Cet ensemble combine les forces de plusieurs modèles détecteurs pour obtenir un score qui permet de distinguer les contenus artificiels des contenus naturels. Nos expériences, qui s'appuient sur des jeux de données standards contenant plusieurs langues et plusieurs domaines, montrent que notre méthode d'ensemble permet d'obtenir un système qui identifie de nombreux générateurs. À modèles fixés, nos résultats sont comparables avec l'état-de-l'art. Nous présentons également de nombreuses ablations complémentaires, détaillées dans la section 6. Notre code et les données produites dans le cadre de cet article sont disponibles sur le github suivant : `https://github.com/BaggerOfWords/MOSAIC`.

2 État de l'Art

Aperçu du domaine. La popularité des grands modèles de langue suscite de nombreuses inquiétudes quant à leur possible détournement pour des usages malicieux et illégaux, comme la désinformation (Zellers *et al.*, 2019), la rédaction de faux articles de recherche (Liu *et al.*, 2024a), ou encore la triche aux examens (Vasilatos *et al.*, 2023). À la lecture, les écrits produits par des machines sont difficilement identifiables (Gehrmann *et al.*, 2019) ; les méthodes cherchant à les détecter de manière automatique suscitent donc un intérêt croissant. Ce problème peut être traité comme un problème de décision binaire (texte naturel ou artificiel), ou bien s'intéresser à la détection d'un modèle particulier (ChatGPT par exemple, (Mitrović *et al.*, 2023; Liu *et al.*, 2024a)) ou encore un modèle parmi d'autres (Li *et al.*, 2023). Certains travaux considèrent une granularité plus fine, visant à différencier les textes entièrement artificiels des textes « raffinés » par des machines (Abassy *et al.*, 2024; Liu *et al.*, 2024b) ; dans ce cadre, on pourra s'intéresser à identifier les fragments ou les passages artificiels

1. Comme en témoigne l'abandon par OpenAI de leur détecteur `https://openai.com/index/new-ai-classifier-for-indicating-ai-written-text/`.

disséminés au sein de textes naturels. Une autre distinction est souvent faite entre les méthodes se focalisant sur un domaine textuel (comme les contenus scientifiques (Liyanage *et al.*, 2022), académiques (Liu *et al.*, 2024c) ou les écrits du Web (Fagni *et al.*, 2021; Kumarage *et al.*, 2023)) et les méthodes plus généralistes. Lorsque le modèle générateur est connu à l'avance, plusieurs méthodes sont envisageables selon les informations renvoyées par ce dernier : seulement les textes produits, ou bien les textes et les probabilités associées à chaque token.

Méthodes supervisées. Sous l'hypothèse d'un unique modèle générateur, la détection dans un cadre supervisé semble très bien fonctionner, avec des résultats proches de la perfection (Zellers *et al.*, 2019; Guo *et al.*, 2023; Liu *et al.*, 2024c), obtenus avec des classifieurs reposant sur les représentations calculées par des grands modèles de langue comme RoBERTa (Conneau *et al.*, 2020) ou T5 (Raffel *et al.*, 2020). Cependant, ces approches sont fragiles car elles dépendent de la paire générateur-détecteur utilisée à l'entraînement (Antoun *et al.*, 2024), ce qui incite par exemple Verma *et al.* (2023) à concevoir des extracteurs de caractéristiques automatiques à partir de plusieurs détecteurs afin d'améliorer la robustesse de leur système.

Méthodes non supervisées. En l'absence d'hypothèse sur le ou les générateurs, il faut s'en remettre à des méthodes non-supervisées capables de détecter des textes sans exemple. Les approches actuelles reposent sur l'idée que les textes humains sont en général plus « surprenants » que les textes artificiels [2], ce qui entraîne ainsi une différence de log-probabilité au niveau des unités textuelles (*tokens*) [3]. C'est l'idée principale de GPTzero [4]. Ainsi, un simple seuillage sur log-vraisemblance moyenne ou la perplexité des textes permet d'avoir de bons résultats de référence (voir (Gehrmann *et al.*, 2019; Ippolito *et al.*, 2020; Mitchell *et al.*, 2023)). Ces méthodes sont cependant très dépendantes du choix d'un modèle *détecteur*, nécessaire pour le calcul des log-probabilités, qui doit également être robuste à des changements de domaine, de genre, de style, de langue (Wang *et al.*, 2024) ou encore de modèle générateur (Antoun *et al.*, 2024).

Méthodes à perturbation. Avec l'intuition que de petites modifications conduisent, en moyenne, à rendre un texte artificiel moins probable (contrairement aux textes humains), Mitchell *et al.* (2023) et Bao *et al.* (2024) développent un critère statistique fondé sur la courbure de la fonction de log-probabilité, obtenant par la même occasion des scores quasi-parfaits sur des textes produits par cinq modèle différents couvrant trois domaines. Le score *Binoculars* d'Hans *et al.* (2024) repose lui aussi sur la log-probabilité des tokens, qu'ils viennent comparer à l'entropie croisée d'un modèle auxiliaire. Ce modèle de base est détaillé à la section 3.3. Ces travaux pointent ainsi du doigt la **sur-dépendance à l'égard d'un modèle détecteur spécifique comne une limitation majeure de l'état de l'art.**. La méthode que nous proposons essaie de remedier à cette faiblesse, en utilisant un ensemble de modèles, à l'instar de méthodes supervisées (Verma *et al.*, 2023; Wang *et al.*, 2023; El-Sayed & Nasr, 2023; Liyanage & Buscaldi, 2023).

Autres méthodes. D'autres approches demandent à un modèle de réécrire les textes à analyser (Mao *et al.*, 2024; Yang *et al.*, 2024), en partant du principe qu'un modèle de langue aura tendance à peu modifier ses propres productions. Une autre stratégie communément utilisée est le filigranage (*watermarking*) (Kirchenbauer *et al.*, 2023a,b; Liu & Bu, 2024), dont l'efficacité fait encore l'objet de débats (Zhang *et al.*, 2023).

2. En supposant que la génération n'utilise pas un échantillonnage aléatoire.
3. (Mitchell *et al.*, 2023) indique que le passage au log rend les différences plus flagrantes.
4. https://gptzero.me/

3 Détection de Textes Générés avec un Ensemble de Modèles

3.1 Contexte

Nous considérons des modèles pour des tâches de génération qui définissent une distribution de probabilité sur des chaînes de caractères. Formellement, les modèles de langue sont des distributions de probabilité sur un espace de sortie $\mathcal{Y}$ qui contient toutes les séquences possibles sur un vocabulaire fini $\Omega : \mathcal{Y} \triangleq \left\{ \texttt{BOS} \circ \mathbf{y} \circ \texttt{EOS} \,|\, \mathbf{y} \in \Omega^* \right\}$, où BOS et EOS désignent respectivement les tokens de début et de fin de séquence, et Ω^* est la fermeture de Kleene de Ω.

Les grands modèles de langue actuels sont paramétrés par des poids entraînables $\theta \in \Theta$. Ces modèles sont tels que pour tout $t > 0$, $p_\theta(\cdot|\mathbf{y}_{<t})$ définit une distribution de probabilité conditionnelle sur $\bar{\Omega} = \Omega \cup \texttt{EOS}$. La probabilité d'une séquence $\mathbf{y} = \langle y_0, \ldots, y_T \rangle$ est donnée par :

$$p(\mathbf{y}) = \prod_{t=1}^{T} p_\theta(y_t|\mathbf{y}_{<t}) \tag{1}$$

avec $\mathbf{y}_{<t} = \langle y_0, \ldots, y_{t-1} \rangle$, $y_0 = \texttt{BOS}$ et $y_T = \texttt{EOS}$.

Mesure de l'information. Une relation fondamentale en théorie de l'information relie la probabilité d'un message à la quantité d'information qu'il véhicule, selon Cover & Thomas (2006) : information $= -\log(\text{probabilité})$, en supposant l'utilisation de techniques de codage telles que les codes de Huffman et les codes arithmétiques (Shields, 1996) qui permettent d'obtenir des longueurs de message approchant étroitement cette longueur idéale en notation binaire.

Explications des données. Étant donné un corpus de texte représenté par une séquence finie $\mathbf{y}_{<t} = \langle y_0, \ldots, y_{t-1} \rangle$, une « explication » de l'unité y_t est une séquence binaire qui l'encode avec une longueur *minimum* :

$$\mathcal{L}_p(y_t|\mathbf{y}_{<t}) \triangleq -\log p(y_t|\mathbf{y}_{<t}).$$

$\mathcal{L}_p(y_t|\mathbf{y}_{<t})$ est également souvent appelée la **surprise** du modèle (Samson, 1953) pour l'entrée y_t. Son espérance est appelée l'**entropie conditionnelle :**

$$\mathcal{H}_p(Y_t|\mathbf{y}_{<t}) = \sum_{y_t \in \Omega} p(y_t|\mathbf{y}_{<t}) \mathcal{L}_p(y_t|\mathbf{y}_{<t}).$$

Un autre concept important est celui d'**information mutuelle conditionnelle** (IM) entre deux variables aléatoires $\mathbb{M}$ et Y_t, étant donnée une séquence $\mathbf{y}_{<t}$, définie par (Cover & Thomas, 2006) :

$$\mathcal{I}_p(\mathbb{M}; Y_t|\mathbf{y}_{<t}) = \mathcal{H}_p(Y_t|\mathbf{y}_{<t}) - \mathcal{H}_p(Y_t|\mathbb{M}, \mathbf{y}_{<t}), \tag{2}$$

$$\mathcal{H}_p(Y_t|\mathbb{M}, \mathbf{y}_{<t}) = \mathop{\mathbb{E}}_{m \sim \mu(m|\mathbf{y}_{<t})} \mathcal{H}_p(Y_t|m, \mathbf{y}_{<t}). \tag{3}$$

L'information mutuelle conditionnelle mesure la quantité d'information que nous obtenons sur $\mathbb{M}$ en observant Y_t, connaissant déjà $\mathbf{y}_{<t}$.

3.2 Méthodes de détection multi-modèles

Pour la détection non-supervisée de textes artificiels, les méthodes les plus efficaces s'appuient sur un modèle de langue détecteur (Guo *et al.*, 2023). Cependant, les rapides progrès de l'intelligence

artificielle rendent ces approches de moins en moins efficaces. Les résultats de FastDetectGPT et *Binoculars* suggèrent que la détection peut être considérablement améliorée en utilisant simultanément deux modèles : dans cette dernière étude, les scores de détection sont obtenus en comparant la surprise d'un modèle avec l'entropie croisée par rapport à l'autre, moyennée sur tous les symboles de la séquence en entrée.

Une généralisation naturelle de ces travaux consiste à se poser la question suivante : **comment tirer parti de plusieurs modèles pour améliorer la détection ?** Une approche simple consiste à tester toutes les paires de modèles, et à retenir la paire qui conduit aux meilleurs scores sur un jeu de validation, comme indiqué dans le tableau 3 de (Hans *et al.*, 2024, p. 15) et la tableau 7 de (Bao *et al.*, 2024, p. 19). Cependant, la paire de modèles qui offre les meilleures performances peut varier en fonction de l'ensemble de validation, entraînant ainsi des fluctuations des performances lorsque le domaine ou la langue de test vient à changer. De plus, cette approche implique une augmentation exponentielle du nombre de combinaisons à explorer lorsque le nombre de modèles s'accroît.

Avant d'aborder cette question principale, nous réexaminons et reformulons *Binoculars* et FastDetectGPT [5] en utilisant les concepts introduits précédemment et explorons d'éventuelles extensions de ces modèles.

3.3 Un réexamen de la méthode *Binoculars*

Le score *Binoculars* $B_{p,q}(\mathbf{y})$ pour une séquence d'entrée $\mathbf{y} = \langle y_0, y_1, \ldots \rangle$, utilisant deux modèles q et p tels qu'exprimés dans (1), est défini par :

$$B_{p,q}(\mathbf{y}) \triangleq \frac{\sum_{t=1}^{T} \sum_{y \in \Omega} \mathbb{1}[y = y_t] \mathcal{L}_q(y_t | \mathbf{y}_{<t})}{\sum_{t=1}^{T} \sum_{y \in \Omega} p(y | \mathbf{y}_{<t}) \mathcal{L}_q(y | \mathbf{y}_{<t})}, \tag{4}$$

avec $\mathcal{L}_q(y_t|\mathbf{y}_{<t}) = -\log q(y_t|\mathbf{y}_{<t})$, et $p(y|\mathbf{y}_{<t})$ et $q(y|\mathbf{y}_{<t})$ représentant respectivement les probabilités attribuées par les modèles p et q au symbole y_t conditionné par le contexte courant $\mathbf{y}_{<t}$. Notons que l'équation (4) implique que les distributions q et p partagent un même support et la même segmentation sous-lexicale.

Du point de vue de la théorie de l'information, on peut interpréter le numérateur comme la longueur moyenne d'encodage la plus courte du texte observé selon le modèle q, tandis que le dénominateur représente la longueur d'encodage que l'on obtiendrait si le texte avait été généré par échantillonnage aléatoire à partir du modèle p. Pour cette raison, dans ce qui suit, nous appelons p le **modèle de référence**.

Il est intéressant de noter que FastDetectGPT exploite une même similaire, mais calcule une différence plutôt qu'un rapport. Bien que les deux méthodes soient équivalentes, FastDetectGPT normalise le score pour chaque unité au lieu de moyenner sur l'ensemble du texte (voir Annexe A).

Comment choisir le modèle de référence le plus prometteur ? Ces deux méthodes se basent sur l'intuition que le numérateur — la log-perplexité du texte examiné — tend à être plus faible pour les textes produits par des machines. Par le même raisonnement, le terme du dénominateur devrait être plus faible pour les textes artificiels. Cela suggère qu'avec une famille de modèles $\mathcal{P}_{\mathcal{M}}(\mathcal{Y}) = \{p_m(\mathbf{y}) : m \in \mathcal{M}\}$, et étant donné un échantillon de texte humain $\mathbf{y}_{\text{hum}}$, nous pouvons

5. Nous nous concentrons ici sur *Binoculars*, l'analyse de FastDetectGPT est fournie en Annexe A.

Algorithm 1 MOSAIC Scoring

Input : text $\mathbf{y} = \langle y_0, y_1, \ldots \rangle$, LLMs $m \in \mathcal{M}$, with $m^\star$ the reference model
for y_t in $\mathbf{y}$ **do**
  $\mu^\star(m|\mathbf{y}_{<t}) \leftarrow$ Blahut–Arimoto $(\mathcal{P}_\mathcal{M}(\mathcal{Y}); \mathbf{y}_{<t})$ ▷ On obtient les poids $\mu^\star$
  $q^\star(y_t|\mathbf{y}_{<t}) \leftarrow \sum_{m \in \mathcal{M}} \mu^\star(m|\mathbf{y}_{<t}) p_m(y_t|\mathbf{y}_{<t})$ ▷ On construit la mixture
  $s_t(\mathbf{y}) \leftarrow \mathcal{L}_{q^\star}(y_t|\mathbf{y}_{<t}) - \sum_{y \in \Omega} p_{m^\star}(y \mid \mathbf{y}_{<t}) \mathcal{L}_{q^\star}(y \mid \mathbf{y}_{<t})$ ▷ Surprise - entropie croisée
end for
$S_\mathcal{M}(\mathbf{y}) \leftarrow \frac{1}{T} \sum_t s_t(\mathbf{y})$ ▷ Score MOSAIC pour tout le texte

utiliser le critère suivant :

$$m^\star(\mathbf{y}_{\text{hum}}) \triangleq \operatorname*{argmin}_{m \in \mathcal{M}} - \sum_{t=1}^{T} \log p_m(y_t|\mathbf{y}_{<t}). \tag{5}$$

En d'autres termes, le modèle de référence $p_{m^\star}$ doit être le modèle de l'ensemble $\mathcal{P}_\mathcal{M}(\mathcal{Y})$ qui attribue la log-perplexité la plus faible aux échantillons de texte humain $\mathbf{y}$. Il s'avère qu'il s'agit généralement du plus grand modèle de l'ensemble, ce qui est en accord avec les tableaux présentés dans (Bao *et al.*, 2024; Hans *et al.*, 2024) et confirmé par nos expériences de la section 6.2. La méthode de sélection formalisée par (5) permet de choisir le modèle de référence le plus prometteur, $p_{m^\star}$ au sein d'une famille $\mathcal{P}_\mathcal{M}(\mathcal{Y})$. Malheureusement elle ne fournit pas de critère pratique pour optimiser le choix du modèle q, qui est nécessaire pour évaluer (4). Cette question est abordée dans la section suivante.

	chatgpt	**cohere-c**	**cohere**	**gpt2**	**gpt3**	**gpt4**	**llama-c**	**mistral-c**	**mistral**	**mpt-c**	**mpt**
Bino (best)	0,996	0,985	0,979	0,812	0,999	0,969	1,000	0,998	0,915	0,999	0,946
Config.	T13b/L-c	L/L-c	T13b/L-c	T13b/T7b	T13b/L-c	T13b/L-c	L/L-c	T13b/L-c	T13b/T7b	T7b/L-c	T13b/T7b
Bino (min)	0,511	0,688	0,711	0,459	0,945	0,376	0,741	0,560	0,609	0,661	0,637
Bino (avg)	0,837	0,900	0,870	0,652	0,983	0,720	0,928	0,876	0,774	0,895	0,798
Fast (best)	0,994	0,981	0,979	0,858	0,996	0,974	1,000	0,993	0,923	0,995	0,966
Config.	T13b/L-c	L/L-c	L/L-c	T7b/L	L/L-c	T13b/L-c	L/L-c	T13b/L-c	T13b/L-c	T13b/T7b	L/L-c
Fast (min)	0,505	0,673	0,705	0,501	0,914	0,363	0,740	0,552	0,606	0,647	0,636
Fast (avg)	0,802	0,853	0,860	0,684	0,961	0,691	0,918	0,869	0,796	0,866	0,830

TABLEAU 1 – AUROC de Bino(culars) & Fast(DetectGPT) sur RAID avec les modèles Llama et Tower. Les cellules « best », « avg » et « min » indiquent respectivement le score maximal, la moyenne et le score le plus faible parmi toutes les configurations. « T » et « L » désignent respectivement les modèles TowerBase et Llama-2-7b, tandis que « -c » correspond à la version « chat ».

4 Présentation de MOSAIC

S'appuyant sur la même intuition que les techniques présentées ci-dessus, nous proposons MOSAIC, une méthode conçue pour exploiter simultanément plusieurs modèles. La différence principale par rapport aux approches précédentes est qu'au lieu d'utiliser un unique modèle pour q, MOSAIC le définit comme **un mélange de modèles, différent pour chaque unité**. Les poids de ce mélange sont définis formellement dans la proposition suivante et illustrés sur la figure 1.

Proposition 1 (Modèle optimal)**.** *Le modèle optimal, $q^\star$ — qui minimise la longueur d'encodage des unités textuelles — est défini par un mélange dont les coefficients varient à chaque position t selon :*

$$q^\star(y_t|\mathbf{y}_{<t}) \triangleq \sum_{m\in\mathcal{M}} \mu^\star(m|\mathbf{y}_{<t}) p_m(y_t|\mathbf{y}_{<t}), \tag{6}$$

où la distribution $\mu^\star(\cdot|\mathbf{y}_{<t})$ de $\mathbb{M}$ sur les indices des modèles dans $\mathcal{M}$ satisfait :

$$\mu^\star(\cdot|\mathbf{y}_{<t}) \triangleq \underset{\mu\in\mathcal{P}(\Omega)}{\operatorname{argmax}}\ \mathcal{I}_p\big(\mathbb{M}; Y_t|\mathbf{y}_{<t}\big). \tag{7}$$

De plus, les poids $\{\mu^\star(m|\mathbf{y}_{<t})\}_{m\in\mathcal{M}}$ dépendent du préfixe $\mathbf{y}_{<t}$; ils peuvent être calculés efficacement en utilisant l'algorithme Blahut–Arimoto (Arimoto, 1972; Blahut, 1972).

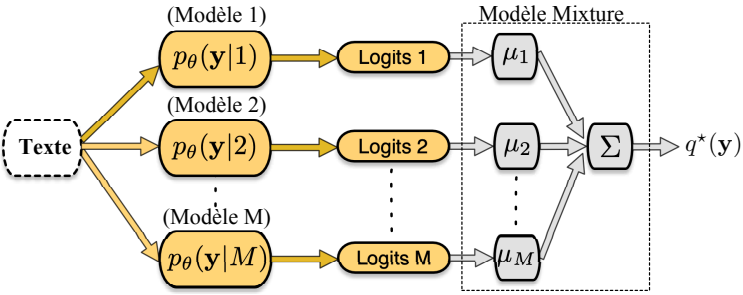

FIGURE 1 – Notre mélange, les poids $\{\mu_i\}$ sont associés à chaque token et à chaque modèle.

Definition 1 (Score MOSAIC)**.** Pour une phrase d'entrée $\mathbf{y} = \langle w_0, w_1, \ldots\rangle$, et des modèles indexés par $\mathcal{M} = \{1, \ldots, M\}$ partageant un tokeniseur commun, le score MOSAIC est alors défini par :

$$S_{m^\star,\mathcal{M}}(\mathbf{y}) \triangleq \frac{1}{T}\sum_{t=1}^{T}\sum_{y\in\Omega}\Bigg[\underbrace{\mathbb{1}_{\{y=y_t\}}\mathcal{L}_{q^\star}(y_t \mid \mathbf{y}_{<t})}_{\text{(encodage pour le symbole observé)}} - \underbrace{p_{m^\star}(y_t \mid \mathbf{y}_{<t})\mathcal{L}_{q^\star}(y_t \mid \mathbf{y}_{<t})}_{\text{(encodage pour le symbole généré par le modèle } m)}\Bigg] \tag{8}$$

où $\mathcal{L}_{q^\star}(y_t|\mathbf{y}_{<t}) = -\log q^\star(y_t|\mathbf{y}_{<t})$ et $m^\star$ est le modèle de référence (5) ayant la plus faible perplexité sur les textes humains.

Cette formulation met en évidence la similarité entre les scores MOSAIC et *Binoculars*, qui ne diffèrent que par la façon dont ils calculent la surprise moyenne : avec un modèle fixe pour *Binoculars* ou avec un mélange dépendant de la position pour MOSAIC. Ce score est utilisé pour détecter les textes artificiels de la manière suivante : étant donné un seuil approprié $\delta > 0$ et un texte $\mathbf{y}$, si $S_{m^\star,\mathcal{M}}(\mathbf{y}) \geq \delta$, le texte est classé comme « humain » ; sinon, il est considéré comme « artificiel ». L'implémentation de cette méthode est formalisée dans l'algorithme 1.

5 Paramètres Expérimentaux

5.1 Jeux de données & Métriques

Nous évaluons notre méthode sur un ensemble de textes et de modèles génératifs de la littérature : RAID (Dugan *et al.*, 2024) et M4 (Wang *et al.*, 2024). Ces corpus sont décrits dans le tableau 2.

		Humain		Artificiel	
Nom du Corpus	# Générateurs	# textes	longueur moyenne	# textes	longueur moyenne
RAID (sampling)	11	1k	452	11k	373
RAID adversarial	11	1k	452	11k	658
RAID+	2	1k	452	2k	410
M4 (Multilingue)	1	15k	729	15k	649

TABLEAU 2 – Détails des jeux de données : RAID+ a été généré pour cette étude en utilisant des modèles dans nos ensembles. # Gen indique le nombre de générateurs utilisés pour les textes artificiels. La longueur moyenne des textes est exprimée en unités segmentées par Llama-2.

RAID contient environ 15 000 textes naturels en anglais issus de divers domaines ; la version artificielle comprend leurs équivalents produits avec plusieurs modèles et procédures d'échantillonnage. Nous sélectionnons un sous-ensemble aléatoire de 1 000 textes humains et leurs homologues produits par un échantillonnage ancestral. RAID inclut également un sous-corpus artificiellement bruité ; nous rapportons les résultats de différentes attaques antagonistes dans la Table 9 en Annexe.

Le corpus M4 [6] est un regroupement de textes naturels collectés à partir de sources diverses (Wang *et al.*, 2024). Des textes artificiels comparables sont produits par 6 modèles de langue, en utilisant pour instructions d'entrée des titres d'articles (de presse ou scientifique) ou des résumés selon les domaines. Dans nos expériences, nous utilisons un seul générateur multilingue (ChatGPT), avec les sous-corpus de M4 composés de 3 000 paires de textes (artificiels, naturels) en chinois, russe, bulgare, arabe et allemand.

Nous avons complété les extraits de **RAID** avec des textes produits (par échantillonage aléatoire) à l'aide de modèles présents dans nos ensembles, en utilisant les mêmes instructions que celles utilisées pour RAID. Nous appelons RAID+ ce corpus complémentaire.

Métriques. Comme dans la plupart des études, nous rapportons le score AUROC comme principale métrique d'évaluation. Dans certaines applications, le taux de vrais positifs (TPR) pour un taux de faux positifs prédéfini (par exemple, 5%) est également pertinent et est indiqué. Tous ces scores sont obtenus en utilisant scikit-learn (Pedregosa *et al.*, 2011).

5.2 Choix des modèles

Dans nos expériences, nous utilisons deux ensembles composés chacun de quatre modèles partageant un vocabulaire commun (de tailles respectives 65k pour Falcon, 32k pour Llama-2) : **La famille Falcon** : (Almazrouei *et al.*, 2023) Falcon-7b, Falcon-7b-instruct, Falcon-40b, Falcon-40b-instruct et **des variantes de Llama** : (Touvron *et al.*, 2023) Llama-2-7b, Llama-2-7b-chat, TowerBase-7b et TowerBase-13b (Alves *et al.*, 2024). Les deux familles contiennent les modèles utilisés pour l'approche *Binoculars*, que nous avons complétés afin d'analyser des ensembles de plus grande taille.

AUROC	chatgpt	cohere-c	cohere	gpt2	gpt3	gpt4	llama-c	mistral-c	mistral	mpt-c	mpt
MOSAIC-2	0.928	0.978	0.968	0.803	0.994	0.902	0.998	0.970	0.895	0.990	0.920
MOSAIC-3	0.917	0.978	0.971	0.818	0.994	0.898	0.996	0.969	0.903	0.988	0.925
MOSAIC-4	0.970	0.979	0.980	0.815	0.997	0.955	0.998	0.988	0.914	0.994	0.927
TPR@5%FPR	**chatgpt**	**cohere-c**	**cohere**	**gpt2**	**gpt3**	**gpt4**	**llama-c**	**mistral-c**	**mistral**	**mpt-c**	**mpt**
MOSAIC-2	0.674	0.923	0.893	0.365	0.984	0.636	0.999	0.881	0.577	0.960	0.643
MOSAIC-3	0.629	0.922	0.904	0.377	0.984	0.610	0.998	0.871	0.584	0.948	0.644
MOSAIC-4	0.831	0.936	0.928	0.351	0.991	0.771	1.000	0.943	0.568	0.976	0.624

TABLEAU 3 – Performances de MOSAIC sur les données RAID avec un nombre variable de modèles. « -c » indique la version chat d'un modèle. 2 modèles : Llama-2-7b et sa version chat, 3 modèles : + TowerBase-7B, 4 modèles : + TowerBase-13B.

6 Résultats

6.1 Résultats sur RAID

Dans le tableau 1, nous réalisons une recherche systématique afin d'identifier la configuration la plus performante pour chaque modèle générateur utilisé dans RAID, configuration appelée « Oracle » dans la Figure 2. Le modèle optimal varie selon l'ensemble de données, ce qui rend cette méthode incompatible avec une approche « agnostique au générateur ». Comme l'indiquent les scores moyens de *Binoculars* et FastDetectGPT sur l'ensemble des combinaisons, de nombreuses paires conduisent à des performances médiocres, soulignant le besoin de disposer d'un critère de sélection des paires.

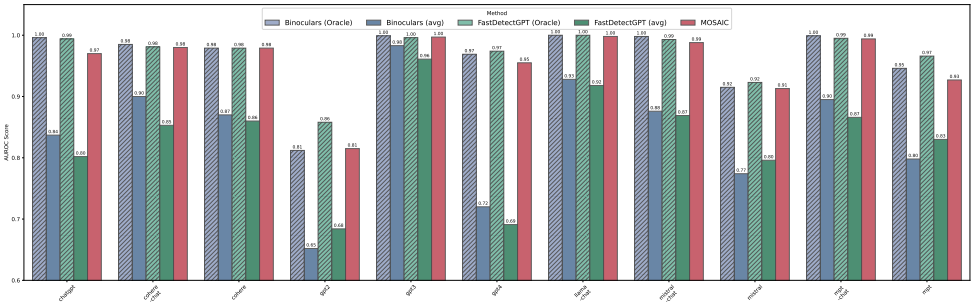

FIGURE 2 – Comparaison des AUROC de MOSAIC sur le jeu de données RAID avec les modèles Llama et Tower. Les noms des modèles représentent le modèle utilisé pour générer le jeu de données.

Dans la Figure 2, nous observons que MOSAIC **est aussi performant que les configurations oracles des autres méthodes**, à l'exception des textes produits par GPT2. Cela peut s'expliquer par la qualité moindre de ces productions. La piètre performance moyenne de toutes les combinaisons sur les données produites par ce modèle plus petit vient soutenir cet argument. En revanche, la plupart des combinaisons de deux modèles peinent à détecter correctement GPT4. Cependant, puisque les productions de GPT4 sont de meilleure qualité, MOSAIC reste efficace pour les identifier.

Agrandir notre ensemble. Pour démontrer l'efficacité de notre méthode ensembliste, nous présentons

6. Pour *Multi-Lingual, Multi-Domain, Multi-Generator Machine-Generated text*.

les résultats de MOSAIC utilisant 2, 3 et 4 modèles. Le modèle de référence est choisi selon le critère défini en section 3.3. Dans les résultats du tableau 3, nous constatons qu'ajouter le « meilleur » modèle, TowerBase-13B, conduit à une amélioration des performances, sauf pour « mistral-chat » et « mpt-chat ». L'ajout de la version 7B de Tower ne modifie guère les résultats de notre méthode, probablement parce que ses capacités en anglais sont similaires à celles des modèles Llama déjà disponibles. Agrandir l'ensemble semble avoir le moins d'effet lorsque le modèle générateur est petit.

Inclusion du générateur dans l'ensemble. À partir des 1 000 échantillons humains de RAID et des instructions associées, nous avons produit 1 000 textes par échantillonnage ancestral. MOSAIC (avec la famille Falcon) obtient un AUC de 0,965 pour les textes produits par Falcon-40b ; avec la famille Llama l'AUC est de 0,986 pour les textes produits par Llama-2-7b. Ces résultats suggèrent que la taille du générateur et la qualité de ses productions importent plus que son usage en tant que détecteur.

Attaques antagonistes. Les performances sur cet ensemble « bruité » sont présentées dans le tableau 9 en Annexe. MOSAIC (avec Llama et Tower) résiste assez bien à ces modifications, à l'exception des attaques « synonymes » et « espaces à largeur zéro », qui détériorent significativement nos performances en raison des importants changements de perplexité qu'elles provoquent.

6.2 Choix du meilleur modèle « de référence »

Comme mentionné section 3.3, le modèle m utilisé pour calculer l'entropie conditionnelle doit être celui qui présente la plus faible perplexité sur des textes humains. Le tableau 4 présente les log-perplexités de nos modèles sur les parties « humaines » des données M4. Le tableau 5 donne ensuite la performance de MOSAIC en fonction du choix de m. L'heuristique de sélection du modèle de référence (Section 3.3) donne les meilleurs résultats. Ce critère simple nécessite quelques textes de développement humains ; en leur absence, choisir le plus grand modèle de l'ensemble fera l'affaire.

	Arabe	Bulgare	Chinois	Allemand	Russe
TowerBase-13B	**1,2743**	**1,8052**	2,3047	**1,4912**	**1,5069**
TowerBase-7B	1,3929	1,9839	2,3527	1,6169	1,6084
Llama-2-7b-chat	1,7379	2,3175	2,6800	2,1189	2,2917
Llama-2-7b	1,3506	1,8291	**2,1286**	1,6117	1,7778

TABLEAU 4 – Valeurs de log-perplexité de nos modèles pour la partie « humaine » de M4

Modèle m	Arabe	Bulgare	Chinois	Allemand	Russe
TowerBase-13B	**0,9563**	**0,9888**	**0,9752**	**0,9311**	**0,9148**
TowerBase-7B	0,9111	0,9578	0,9558	0,8679	0,8569
Llama-2-7b-chat	0,7768	0,8262	0,5849	0,6751	0,4321
Llama-2-7b	0,8947	0,9762	0,9059	0,9200	0,6814

TABLEAU 5 – AUROC de MOSAIC quand on fait varier le « modèle de référence » $m^\star$.

6.3 Ablations

Un seul modèle, plusieurs décodages Une implémentation plus légère de MOSAIC consiste à utiliser un unique modèle, dont les scores sont modifiés afin de simuler plusieurs distributions de probabilité. De nombreuses techniques d'échantillonnage reposent sur l'adaptation de la distribution d'un modèle sous-jacent (Meister *et al.*, 2023), telles que top-k (Fan *et al.*, 2018), top-p (Holtzman *et al.*, 2020) et η (Hewitt *et al.*, 2022). Avec ces techniques, MOSAIC ne requiert qu'un modèle en inférence. Nous explorons cette approche avec Llama-2-7b et quatre valeurs de top-p [7]. Les résultats dans en Annexe, Tableau 8. Les résultats moins bons que la configuration à 2 modèles (Tableau 3), ce

7. Dans notre implémentation, nous utilisons une version lissée de top-p, de sorte que toutes les distributions aient le même support, condition requise pour calculer le terme d'entropie croisée dans l'équation (8).

qui suggère qu'appliquer top-p de quatre manières différentes n'introduit pas autant de diversité que le modèle « *instruct* ».

Ensemble uniforme Plutôt que d'utiliser l'algorithme de Blahut-Arimoto pour combiner les modèles, nous avons naïvement attribué des poids identiques à chaque modèle de l'ensemble. Les résultats sont dans le tableau 7, ligne « MOSAIC (uniform Falcon) ». Nous constatons que ce mélange uniforme donne de bons résultats mais reste inférieur à notre méthode de calcul optimal des poids de mélange.

7 Conclusion

À travers nos expériences, nous avons démontré que MOSAIC combine efficacement les modèles de l'ensemble, obtenant d'excellents résultats sur l'ensemble des jeux de données testés. Cette approche est entièrement non supervisée et extensible à de nouveaux modèles. Les attaques antagonistes n'ont qu'un effet minimal sur les performances, bien que la méthode ne soit pas optimisée pour leur faire face. Cependant, MOSAIC est actuellement coûteuse en termes de calcul, de potentielles améliorations sont proposées Section 6.3 et en Annexe E. Enfin, il semble nécessaire de poursuivre les recherches pour comprendre pleinement comment évaluer la « distance » entre différents modèles, afin de choisir le meilleur ensemble, susceptible de couvrir toutes les stratégies de génération possibles.

Remerciements

Les auteurs remercient les personnes ayant évalué l'article pour leur remarques et critiques. Le premier auteur est financé par une bourse de la mission Internationale du CNRS. Ce travail a utilisé les ressources de calcul de l'IDRIS (allocations 2023-AD011014903 et 2025-AD011014903R1) à travers GENCI.

Références

ABASSY M., ELOZEIRI K., AZIZ A., TA M. N., TOMAR R. V., ADHIKARI B., AHMED S. E. D., WANG Y., MOHAMMED AFZAL O., XIE Z., MANSUROV J., ARTEMOVA E., MIKHAILOV V., XING R., GENG J., IQBAL H., MUJAHID Z. M., MAHMOUD T., TSVIGUN A., AJI A. F., SHELMANOV A., HABASH N., GUREVYCH I. & NAKOV P. (2024). LLM-DetectAIve : a tool for fine-grained machine-generated text detection. In D. I. HERNANDEZ FARIAS, T. HOPE & M. LI, Éds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, p. 336–343, Miami, Florida, USA : Association for Computational Linguistics. DOI : 10.18653/v1/2024.emnlp-demo.35.

ALMAZROUEI E., ALOBEIDLI H., ALSHAMSI A., CAPPELLI A., COJOCARU R., DEBBAH M., GOFFINET É., HESSLOW D., LAUNAY J., MALARTIC Q., MAZZOTTA D., NOUNE B., PANNIER B. & PENEDO G. (2023). The Falcon Series of Open Language Models. DOI : 10.48550/arXiv.2311.16867.

ALVES D. M., POMBAL J., GUERREIRO N. M., MARTINS P. H., ALVES J., FARAJIAN A., PETERS B., REI R., FERNANDES P., AGRAWAL S., COLOMBO P., DE SOUZA J. G. C. & MARTINS A.

(2024). Tower : An open multilingual large language model for translation-related tasks. In *First Conference on Language Modeling*.

ANTOUN W., SAGOT B. & SEDDAH D. (2024). From text to source : Results in detecting large language model-generated content. In N. CALZOLARI, M.-Y. KAN, V. HOSTE, A. LENCI, S. SAKTI & N. XUE, Éds., *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, p. 7531–7543, Torino, Italia : ELRA and ICCL.

ARIMOTO S. (1972). An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, **18**(1), 14–20. DOI : 10.1109/TIT.1972.1054753.

BAO G., ZHAO Y., TENG Z., YANG L. & ZHANG Y. (2024). Fast-detectGPT : Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*.

BARRON A. R., RISSANEN J. & YU B. (1998). The minimum description length principle in coding and modeling. *IEEE Trans. Inf. Theory*, **44**(6), 2743–2760.

BLAHUT R. (1972). Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, **18**(4), 460–473. DOI : 10.1109/TIT.1972.1054855.

CONNEAU A., KHANDELWAL K., GOYAL N., CHAUDHARY V., WENZEK G., GUZMÁN F., GRAVE E., OTT M., ZETTLEMOYER L. & STOYANOV V. (2020). Unsupervised cross-lingual representation learning at scale. In D. JURAFSKY, J. CHAI, N. SCHLUTER & J. TETREAULT, Éds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 8440–8451, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.acl-main.747.

COVER T. M. & THOMAS J. A. (2006). *Elements of Information Theory*. New York, NY : Wiley, 2nd édition.

CROTHERS E. N., JAPKOWICZ N. & VIKTOR H. L. (2023). Machine-generated text : A comprehensive survey of threat models and detection methods. *IEEE Access*, **11**, 70977–71002. DOI : 10.1109/ACCESS.2023.3294090.

DUGAN L., HWANG A., TRHLÍK F., ZHU A., LUDAN J. M., XU H., IPPOLITO D. & CALLISON-BURCH C. (2024). RAID : A shared benchmark for robust evaluation of machine-generated text detectors. In L.-W. KU, A. MARTINS & V. SRIKUMAR, Éds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 12463–12492, Bangkok, Thailand : Association for Computational Linguistics. DOI : 10.18653/v1/2024.acl-long.674.

EL-SAYED A. & NASR O. (2023). An ensemble based approach to detecting LLM-generated texts. In S. MURESAN, V. CHEN, K. CASEY, V. DAVID, D. NINA, I. KOJI, E. ERIK & U. STEFAN, Éds., *Proceedings of the 21st Annual Workshop of the Australasian Language Technology Association*, p. 164–168, Melbourne, Australia : Association for Computational Linguistics.

FAGNI T., FALCHI F., GAMBINI M., MARTELLA A. & TESCONI M. (2021). TweepFake : About detecting deepfake tweets. *PLOS ONE*, **16**(5), e0251415. Publisher : Public Library of Science, DOI : 10.1371/journal.pone.0251415.

FAN A., LEWIS M. & DAUPHIN Y. (2018). Hierarchical neural story generation. In I. GUREVYCH & Y. MIYAO, Éds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 889–898, Melbourne, Australia : Association for Computational Linguistics. DOI : 10.18653/v1/P18-1082.

GEHRMANN S., STROBELT H. & RUSH A. (2019). GLTR : Statistical detection and visualization of generated text. In M. R. COSTA-JUSSÀ & E. ALFONSECA, Éds., *Proceedings of the 57th Annual*

Meeting of the Association for Computational Linguistics : System Demonstrations, p. 111–116, Florence, Italy : Association for Computational Linguistics. DOI : 10.18653/v1/P19-3019.

GUO B., ZHANG X., WANG Z., JIANG M., NIE J., DING Y., YUE J. & WU Y. (2023). How Close is ChatGPT to Human Experts ? Comparison Corpus, Evaluation, and Detection. DOI : 10.48550/arXiv.2301.07597.

HANS A., SCHWARZSCHILD A., CHEREPANOVA V., KAZEMI H., SAHA A., GOLDBLUM M., GEIPING J. & GOLDSTEIN T. (2024). Spotting LLMs with binoculars : Zero-shot detection of machine-generated text. In *Forty-first International Conference on Machine Learning*.

HEWITT J., MANNING C. & LIANG P. (2022). Truncation sampling as language model desmoothing. In Y. GOLDBERG, Z. KOZAREVA & Y. ZHANG, Éds., *Findings of the Association for Computational Linguistics : EMNLP 2022*, p. 3414–3427, Abu Dhabi, United Arab Emirates : Association for Computational Linguistics. DOI : 10.18653/v1/2022.findings-emnlp.249.

HOLTZMAN A., BUYS J., DU L., FORBES M. & CHOI Y. (2020). The curious case of neural text degeneration. In *Proceedings of the International Conference on Learning Representations*, ICLR.

IPPOLITO D., DUCKWORTH D., CALLISON-BURCH C. & ECK D. (2020). Automatic detection of generated text is easiest when humans are fooled. In D. JURAFSKY, J. CHAI, N. SCHLUTER & J. TETREAULT, Éds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 1808–1822, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.acl-main.164.

KIRCHENBAUER J., GEIPING J., WEN Y., KATZ J., MIERS I. & GOLDSTEIN T. (2023a). A Watermark for Large Language Models. In *Proceedings International Conference on Machine Learning*. DOI : https ://openreview.net/forum ?id=aX8ig9X2a7.

KIRCHENBAUER J., GEIPING J., WEN Y., SHU M., SAIFULLAH K., KONG K., FERNANDO K., SAHA A., GOLDBLUM M. & GOLDSTEIN T. (2023b). On the Reliability of Watermarks for Large Language Models. DOI : 10.48550/arXiv.2306.04634.

KUMARAGE T., GARLAND J., BHATTACHARJEE A., TRAPEZNIKOV K., RUSTON S. & LIU H. (2023). Stylometric Detection of AI-Generated Text in Twitter Timelines. arXiv :2303.03697 [cs], DOI : 10.48550/arXiv.2303.03697.

LI L., WANG P., REN K., SUN T. & QIU X. (2023). Origin Tracing and Detecting of LLMs. arXiv :2304.14072 [cs], DOI : 10.48550/arXiv.2304.14072.

LIU Y. & BU Y. (2024). Adaptive Text Watermark for Large Language Models.

LIU Z., YAO Z., LI F. & LUO B. (2024a). On the Detectability of ChatGPT Content : Benchmarking, Methodology, and Evaluation through the Lens of Academic Writing.

LIU Z., YAO Z., LI F. & LUO B. (2024b). On the detectability of chatgpt content : Benchmarking, methodology, and evaluation through the lens of academic writing. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, p. 2236–2250, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3658644.3670392.

LIU Z., YAO Z., LI F. & LUO B. (2024c). On the detectability of ChatGPT content : Benchmarking, methodology, and evaluation through the lens of academic writing.

LIYANAGE V. & BUSCALDI D. (2023). An ensemble method based on the combination of transformers with convolutional neural networks to detect artificially generated text. In S. MURESAN, V. CHEN, K. CASEY, V. DAVID, D. NINA, I. KOJI, E. ERIK & U. STEFAN, Éds., *Proceedings of the 21st Annual Workshop of the Australasian Language Technology Association*, p. 107–111, Melbourne, Australia : Association for Computational Linguistics.

LIYANAGE V., BUSCALDI D. & NAZARENKO A. (2022). A benchmark corpus for the detection of automatically generated text in academic publications. In N. CALZOLARI, F. BÉCHET, P. BLACHE,

K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk & S. Piperidis, Éds., *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, p. 4692–4700, Marseille, France : European Language Resources Association.

Mao C., Vondrick C., Wang H. & Yang J. (2024). Raidar : geneRative AI detection viA rewriting. In *The twelfth international conference on learning representations*.

Meister C., Pimentel T., Malagutti L., Wilcox E. & Cotterell R. (2023). On the efficacy of sampling adapters. In A. Rogers, J. Boyd-Graber & N. Okazaki, Éds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1437–1455, Toronto, Canada : Association for Computational Linguistics. doi : 10.18653/v1/2023.acl-long.80.

Mitchell E., Lee Y., Khazatsky A., Manning C. D. & Finn C. (2023). DetectGPT : Zero-Shot Machine-Generated Text Detection using Probability Curvature. In *Proceedings International Conference on Machine Learning*, ICML.

Mitrović S., Andreoletti D. & Ayoub O. (2023). ChatGPT or Human ? Detect and Explain. Explaining Decisions of Machine Learning Model for Detecting Short ChatGPT-generated Text. doi : 10.48550/arXiv.2301.13852.

Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M. & Duchesnay E. (2011). Scikit-learn : Machine learning in python. *J. Mach. Learn. Res.*, **12**(null), 2825–2830.

Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W. & Liu P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, **21**(140), 1–67.

Samson E. W. (1953). Fundamental natural concepts of information theory. *ETC : A Review of General Semantics*, **10**(4), 283–297.

Shields P. (1996). *The Ergodic Theory of Discrete Sample Paths*. Graduate studies in mathematics. American Mathematical Society.

Silva J. F. & Piantanida P. (2022). On universal d-semifaithful coding for memoryless sources with infinite alphabets. *IEEE Transactions on Information Theory*, **68**(4), 2782–2800. doi : 10.1109/TIT.2021.3134891.

Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.-A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F., Rodriguez A., Joulin A., Grave E. & Lample G. (2023). LLaMA : Open and Efficient Foundation Language Models. doi : 10.48550/arXiv.2302.13971.

Vasilatos C., Alam M., Rahwan T., Zaki Y. & Maniatakos M. (2023). HowkGPT : Investigating the Detection of ChatGPT-generated University Student Homework through Context-Aware Perplexity Analysis. doi : 10.48550/arXiv.2305.18226.

Verma V., Fleisig E., Tomlin N. & Klein D. (2023). Ghostbuster : Detecting Text Ghostwritten by Large Language Models. doi : 10.48550/arXiv.2305.15047.

von Neumann J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, **100**, 295–320.

Wang P., Li L., Ren K., Jiang B., Zhang D. & Qiu X. (2023). SeqXGPT : Sentence-level AI-generated text detection. In H. Bouamor, J. Pino & K. Bali, Éds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 1144–1156, Singapore : Association for Computational Linguistics.

WANG Y., MANSUROV J., IVANOV P., SU J., SHELMANOV A., TSVIGUN A., WHITEHOUSE C., MOHAMMED AFZAL O., MAHMOUD T., SASAKI T., ARNOLD T., AJI A. F., HABASH N., GUREVYCH I. & NAKOV P. (2024). M4 : Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In Y. GRAHAM & M. PURVER, Éds., *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1369–1407, St. Julian's, Malta : Association for Computational Linguistics.

YANG X., CHENG W., WU Y., PETZOLD L. R., WANG W. Y. & CHEN H. (2024). DNA-GPT : Divergent n-gram analysis for training-free detection of GPT-generated text. In *The twelfth international conference on learning representations*, (ICLR), Vienna, Austria.

ZELLERS R., HOLTZMAN A., RASHKIN H., BISK Y., FARHADI A., ROESNER F. & CHOI Y. (2019). Defending Against Neural Fake News. In H. WALLACH, H. LAROCHELLE, A. BEYGELZIMER, F. D' ALCHÉ-BUC, E. FOX & R. GARNETT, Éds., *Advances in Neural Information Processing Systems*, volume 32 : Curran Associates, Inc.

ZHANG H., EDELMAN B. L., FRANCATI D., VENTURI D., ATENIESE G. & BARAK B. (2023). Watermarks in the Sand : Impossibility of Strong Watermarking for Generative Models.

A Analyse théorique de FastDetectGPT et Binoculars

Comme évoqué dans la Section 3.3, FastDetectGPT ressemble étroitement à *Binoculars*. Cependant, au lieu de calculer directement l'entropie croisée entre deux modèles, la méthode tire N échantillons indépendants du modèle p : $\{\tilde{y}_i \sim p(Y_t|\mathbf{y}_{<t})\}_{i=1}^{N}$, puis renvoie le score suivant :

$$S_{p,q}^{\text{Fast}}(\mathbf{y}) = \frac{-\log q(y_t|\mathbf{y}_{<t}) + \frac{1}{N}\sum_{i=1}^{N} \log q(\tilde{y}_i|\mathbf{y}_{<t})}{\tilde{\sigma}(\mathbf{y}_{<t})},$$

où

$$\tilde{\sigma}^2(\mathbf{y}_{<t}) \triangleq \frac{1}{N-1}\sum_{i=1}^{N}\Big(-\log q(y_i|\mathbf{y}_{<t}) + \frac{1}{N}\sum_{j=1}^{N}\log q(y_j|\mathbf{y}_{<t})\Big)^2 \tag{9}$$

est un terme de normalisation, utile pour la détection des symboles individuels.

B Les fondements théoriques de MOSAIC

Identifier les explications des données. Nous nous intéressons au problème de déterminer une séquence adéquate de modèles $\hat{\mathbf{m}} = \langle \hat{m}_0, \ldots, \hat{m}_T \rangle$.

Notre objectif est de dériver un algorithme qui extrait au mieux la régularité présente dans les données, ce qui revient à identifier **le modèle qui permet la meilleure compression des tokens d'entrée**. Supposons que nous disposions d'une famille de modèles $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$, avec les longueurs de code de Shannon correspondantes :

$$\mathcal{L}_{p_m}(y_t|\mathbf{y}_{<t}) \triangleq -\log p_m(y_t|\mathbf{y}_{<t}),$$

pour chaque y_t. On peut les considérer comme une collection de compresseurs de données, indexés par m. On peut mesurer la performance d'encodage de y_t à l'instant t par rapport à $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$. Si l'on

choisit d'encoder le token y_t avec le modèle $q(y_t|\mathbf{y}_{<t})$, le surplus de longueur de code attendue par rapport à n'importe quelle distribution $p_m \in \mathcal{P}_{\mathcal{M}}(\mathcal{Y})$ est donnée par :

$$\mathcal{R}_m(q\,;\mathbf{y}_{<t}) \triangleq \mathop{\mathbb{E}}_{y_t\sim p_m(y_t|\mathbf{y}_{<t})}\Big[-\log q(y_t|\mathbf{y}_{<t})\Big] - \mathcal{H}_{p_m}(Y_t|\mathbf{y}_{<t}) \tag{10}$$

qui est non négative puisque $\mathcal{H}_{p_m}(Y_t|\mathbf{y}_{<t})$ représente la longueur de code minimale attendue. $\mathcal{R}_m$ représente le nombre moyen de bits supplémentaires nécessaires pour encoder y_t en utilisant le code/LLM $q(y_t|\mathbf{y}_{<t})$, par rapport à $\mathcal{H}_{p_m}(Y_t|\mathbf{y}_{<t})$, c'est-à-dire le nombre de bits nécessaires si nous avions utilisé, rétrospectivement, le LLM le mieux adapté dans $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$. Cependant, l'encodeur ne peut pas connaître le modèle sous-jacent générant artificiellement y_t, nous adoptons donc une approche du pire cas et recherchons des LLMs universels avec une surcharge moyenne minimale, le pire cas étant évalué sur tous les modèles de $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$. $\mathcal{R}_m$ est notre mesure de qualité et, par conséquent, le LLM « optimal » par rapport à $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$, pour un contexte donné $\mathbf{y}_{<t}$, est la distribution qui minimise :

$$q^\star(y_t|\mathbf{y}_{<t}) \triangleq \mathop{\operatorname{argmin}}_{q\in\mathcal{P}(\Omega)} \max_{m\in\mathcal{M}} \mathcal{R}_m(q\,;\mathbf{y}_{<t}) \tag{11}$$

où la minimisation s'effectue sur l'ensemble de toutes les distributions sur Ω. Le minimiseur correspond au code qui présente la moindre surcharge (c'est-à-dire, le moins de bits supplémentaires) par rapport au code optimal, qui serait rétrospectivement le meilleur choix dans une sélection de modèles dans le pire des cas pour générer du texte synthétique parmi tous les LLMs de la famille $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$.

Exploiter les longueurs de code pour identifier les textes générés par l'IA. La surcharge moyenne de la longueur de code optimale $-\log q^\star(y_t|\mathbf{y}_{<t})$ obtenue en résolvant l'équation (11) apparaît comme un choix très pertinent pour construire une fonction de score robuste destinée à détecter les textes générés par l'IA, grâce aux propriétés suivantes :

- Plus le LLM le mieux adapté dans $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$ correspond aux données générées artificiellement, plus la longueur de code $\mathcal{L}_{q^\star}(y_t|\mathbf{y}_{<t}) \triangleq -\log q^\star(y_t|\mathbf{y}_{<t})$ est courte.
- Aucun LLM dans $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$ n'est favorisé par rapport à un autre, puisque $\mathcal{R}_m(q^\star\,;\mathbf{y}_{<t}) \leq \mathcal{R}_m(p\,;\mathbf{y}_{<t})$ pour tout $p \in \mathcal{P}_{\mathcal{M}}(\mathcal{Y})$, c'est-à-dire que nous traitons tous les LLMs de notre univers $\mathcal{P}_{\mathcal{M}}(\mathcal{Y})$ sur un pied d'égalité.

B.1 Preuve de la Proposition 1

Démonstration. Nous devons démontrer l'identité fondamentale :

$$\Gamma(\mathbf{y}_{<t}) \triangleq \min_{q\in\mathcal{P}(\Omega)} \max_{m\in\mathcal{M}} \mathcal{R}_m(q\,;\mathbf{y}_{<t}) = \max_{\mu\in\mathcal{P}(\mathcal{M})} \mathcal{I}(\mathbb{M};Y_t|\mathbf{y}_{<t}), \tag{12}$$

où le $q^\star(y_t|\mathbf{y}_{<t})$ optimal atteignant le minimum est caractérisé par le mélange :

$$q^\star(y_t|\mathbf{y}_{<t}) = \sum_{m\in\mathcal{M}} \mu^\star(m|\mathbf{y}_{<t}) p_m(y_t|\mathbf{y}_{<t}) \tag{13}$$

et la distribution $\mu^\star(m|\mathbf{y}_{<t})$ de la variable aléatoire $\mathbb{M}$ sur $\mathcal{M}$ est obtenue en résolvant :

$$\mu^\star(m|\mathbf{y}_{<t}) \triangleq \mathop{\operatorname{argmax}}_{\mu\in\mathcal{P}(\Omega)} \mathcal{I}\big(\mathbb{M};Y_t|\mathbf{y}_{<t}\big). \tag{14}$$

Pour ce faire, nous partons de la définition de $\mathcal{R}_m$:

$$\mathcal{R}_m(q\,;\mathbf{y}_{<t}) \triangleq \mathop{\mathbb{E}}_{y_t\sim p_m(y_t|\mathbf{y}_{<t})}[-\log q(y_t|\mathbf{y}_{<t})] - \min_{q'\in\mathcal{P}(\Omega)} \mathop{\mathbb{E}}_{y_t\sim p_m(y_t|\mathbf{y}_{<t})}\Big[-\log q'(y_t|\mathbf{y}_{<t})\Big] \tag{15}$$

$$= \mathop{\mathbb{E}}_{y_t\sim p_m(y_t|\mathbf{y}_{<t})}[-\log q(y_t|\mathbf{y}_{<t})] - \mathcal{H}_{p_m}(Y_t|\mathbf{y}_{<t}) \tag{16}$$

$$= \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big), \tag{17}$$

où $\mathcal{D}_{\mathrm{KL}}(\cdot\|\cdot)$ désigne la divergence de Kullback–Leibler. Nous pouvons ainsi formuler notre problème comme suit :

$$\begin{aligned}\Gamma(\mathbf{y}_{<t}) &= \min_{q\in\mathcal{P}(\Omega)} \max_{m\in\mathcal{M}} \mathcal{R}_m(q\,;\mathbf{y}_{<t}) \\ &= \min_{q\in\mathcal{P}(\Omega)} \max_{m\in\mathcal{M}} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) \\ &= \min_{q\in\mathcal{P}(\Omega)} \max_{\mu\in\mathcal{P}(\mathcal{M})} \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big),\end{aligned} \tag{18}$$

la minimisation étant effectuée sur toutes les distributions $q \in \mathcal{P}(\Omega)$, ce qui représente la valeur attendue du regret de q par rapport à la pire distribution sur $\mu \in \mathcal{P}(\mathcal{M})$. On remarque que cela est équivalent au « regret moyen dans le pire des cas » (Barron *et al.*, 1998; Silva & Piantanida, 2022). L'égalité dans (18) découle du fait que

$$\max_{\mu\in\mathcal{P}(\mathcal{M})} \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) \leq \max_{m\in\mathcal{M}} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) \tag{19}$$

et de plus,

$$\max_{m\in\mathcal{M}} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) = \mathop{\mathbb{E}}_{m\sim\widetilde{\mu}} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) \tag{20}$$

en choisissant la mesure $\widetilde{\mu}$ comme une probabilité uniforme sur l'ensemble $\widetilde{\mathcal{M}} \subseteq \mathcal{M}$ défini comme l'ensemble des maximisateurs :

$$\widetilde{\mathcal{M}} \equiv \operatorname*{argmax}_{m\in\mathcal{M}} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big)$$

et zéro sinon.

La convexité de la divergence KL nous permet de réécrire l'expression (18) comme suit :

$$\min_{q\in\mathcal{P}(\Omega)} \max_{\mu\in\mathcal{P}(\mathcal{M})} \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) = \max_{\mu\in\mathcal{P}(\mathcal{M})} \min_{q\in\mathcal{P}(\Omega)} \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) \tag{21}$$

Cela découle de l'examen d'un jeu à somme nulle avec une fonction concave-convexe définie sur un produit d'ensembles convexes. Les ensembles de toutes les distributions de probabilité $\mathcal{P}(\mathcal{M})$ et $\mathcal{P}(\Omega)$ sont deux ensembles convexes non vides, bornés et de dimension finie. De plus, la fonction $(\mu, q) \to \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big)$ est concave-convexe, c'est-à-dire que

$$\mu \to \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big)$$

est concave et,

$$q \to \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big)$$

est convexe pour tout (μ, q). Par le théorème classique du min-max (von Neumann, 1928), l'expression (21) est ainsi établie.

Il reste à montrer que :

$$\begin{aligned}\min_{q\in\mathcal{P}(\Omega)} \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) \\ = \mathcal{I}(\mathbb{M}; Y_t|\mathbf{y}_{<t})\end{aligned} \tag{22}$$

pour toute variable aléatoire $\mathbb{M}$ distribuée selon $\mu \in \mathcal{P}(\mathcal{M})$ et pour chaque distribution $p_m(y_t|\mathbf{y}_{<t})$.

Nous commençons par montrer que :

$$\mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) \geq \mathcal{I}(\mathbb{M}; Y_t|\mathbf{y}_{<t})$$

pour toute distribution $q(\cdot|\mathbf{y}_{<t})$ et pour tout $p_m(y_t|\mathbf{y}_{<t})$. À cette fin, on considère les identités suivantes :

$$\begin{aligned}\mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\big(p_m(\cdot \mid \mathbf{y}_{<t}) \,\|\, q(\cdot \mid \mathbf{y}_{<t})\big) &= \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\big(p_m(\cdot \mid \mathbf{y}_{<t}) \,\|\, p_m(\cdot \mid \mathbf{y}_{<t})\big) + \mathcal{D}_{\mathrm{KL}}\big(p_m(\cdot \mid \mathbf{y}_{<t}) \,\|\, q(\cdot \mid \\ &= \mathcal{I}(\mathbb{M}; Y_t \mid \mathbf{y}_{<t}) + \mathcal{D}_{\mathrm{KL}}\big(p_m(\cdot \mid \mathbf{y}_{<t}) \,\|\, q(\cdot \mid \mathbf{y}_{<t})\big) \\ &\geq \mathcal{I}(\mathbb{M}; Y_t \mid \mathbf{y}_{<t}),\end{aligned} \tag{23}$$

où $p_m(\cdot|\mathbf{y}_{<t})$ désigne la distribution marginale obtenue via μ, et la dernière inégalité découle de la non-négativité de la divergence KL. Finalement, il est aisé de vérifier qu'en choisissant :

$$q^\star(y_t|\mathbf{y}_{<t}) = \mathop{\mathbb{E}}_{m\sim\mu}\big[p_m(y_t|\mathbf{y}_{<t})\big], \tag{24}$$

la borne inférieure dans (23) est atteinte :

$$\min_{q\in\mathcal{P}(\Omega)} \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q(\cdot|\mathbf{y}_{<t})\Big) = \mathop{\mathbb{E}}_{m\sim\mu} \mathcal{D}_{\mathrm{KL}}\Big(p_m(\cdot|\mathbf{y}_{<t}) \,\|\, q^\star(\cdot|\mathbf{y}_{<t})\Big) \tag{25}$$

pour tout $\mu \in \mathcal{P}(\mathcal{M})$, ce qui prouve l'identité dans l'expression (22).

La revendication dans (12) découle en prenant le maximum sur toutes les mesures de probabilité $\mu \in \mathcal{P}(\mathcal{M})$ des deux côtés de (22), et en combinant l'identité obtenue avec les expressions (21) et (18). Le mélange dans (14) découle de l'expression (24), qui est une condition nécessaire pour résoudre le problème min-max. □

C Algorithme de Blahut–Arimoto

C.1 Algorithme

Notre canal peut être spécifié à l'aide de deux variables aléatoires discrètes $(\mathbb{M}, Y_t)$ avec des alphabets $(\mathcal{M}, \Omega)$ et des distributions de probabilité μ et $p_m(y_t|\mathbf{y}_{<t})$, respectivement, conditionnées sur $\mathbf{y}_{<t}$.

Le problème à résoudre est la maximisation de l'information mutuelle, consistant en :

$$\Gamma(\mathbf{y}_{<t}) \triangleq \max_{\mu \in \mathcal{P}(\mathcal{M})} \mathcal{I}_m\big(\mathbb{M}; Y_t | \mathbf{y}_{<t}\big). \tag{26}$$

Si nous notons que la cardinalité $|\mathbb{M}| = M$ et $|\Omega| = N$, alors $p_m(y_t|\mathbf{y}_{<t})$ est une matrice de taille $M \times N$, dont l'entrée à la i-ème ligne et j-ème colonne est notée w_{ij}. Dans le cas de la capacité d'un canal discret, l'algorithme a été introduit dans (Blahut, 1972; Arimoto, 1972) pour résoudre (26). Ils ont trouvé l'expression suivante pour la capacité d'un canal discret avec une loi de canal w_{ij} :

$$\Gamma(\mathbf{y}_{<t}) = \max_{\mu} \max_{Q} \sum_{i=1}^{M} \sum_{j=1}^{N} \mu_i w_{ij} \log \left(\frac{q_{ji}}{\mu_i} \right),$$

où μ et Q sont maximisés sous les contraintes suivantes :

— $\mu \triangleq (\mu_1, \ldots, \mu_M)$ est une distribution de probabilité sur $\mathcal{M}$, c'est-à-dire $\sum_{i=1}^{M} \mu_i = 1$.

— $Q = (q_{ji})$ est une matrice de taille $N \times M$ qui se comporte comme une matrice de transition de Ω vers $\mathcal{M}$ par rapport à la loi de canal. Autrement dit, pour tout $1 \leq i \leq M$ et $1 \leq j \leq N$:

$$q_{ji} \geq 0, \quad q_{ji} = 0 \Leftrightarrow w_{ij} = 0,$$

et chaque ligne doit sommer à 1, c'est-à-dire

$$\sum_{i=1}^{M} q_{ji} = 1.$$

Ensuite, en initialisant une mesure de probabilité $\mu^0 = (\mu_1^0, \mu_2^0, \ldots, \mu_M^0)$ sur $\mathcal{M}$, nous pouvons générer itérativement une séquence $(\mu^0, Q^0, \mu^1, Q^1, \ldots)$ selon :

$$(q_{ji}^t) = \frac{\mu_i^t w_{ij}}{\sum_{k=1}^{M} \mu_k^t w_{kj}} \tag{27}$$

et

$$\mu_k^{t+1} = \frac{\prod_{j=1}^{N} (q_{jk}^t)^{w_{kj}}}{\sum_{i=1}^{M} \prod_{j=1}^{N} (q_{ji}^t)^{w_{ij}}} \tag{28}$$

pour $t = 0, 1, 2, \ldots$.

Ensuite, grâce à la théorie de l'optimisation, en particulier la descente par coordonnées, il a été démontré que cette séquence converge effectivement vers le maximum recherché. Autrement dit,

$$\lim_{t \to \infty} \sum_{i=1}^{M} \sum_{j=1}^{N} \mu_i^t w_{ij} \log \left(\frac{q_{ji}^t}{\mu_i^t} \right) = \Gamma(\mathbf{y}_{<t}).$$

Ainsi, compte tenu d'une loi de canal $p_m(y_t|\mathbf{y}_{<t})$, l'équation (26) peut être estimée numériquement avec une précision arbitraire.

C.2 Complexité computationnelle

La complexité computationnelle de l'algorithme de Blahut-Arimoto se caractérise comme suit :

— **Nombre d'itérations.** L'algorithme converge généralement de manière linéaire, de sorte que le nombre d'itérations requis, noté T, est proportionnel à la précision souhaitée.

— **Opérations par itération.** Chaque itération implique la mise à jour des mesures de probabilité selon (27) et (28), ainsi que l'évaluation de l'information mutuelle, ce qui nécessite des manipulations matricielles. Soient M et N les cardinalités des alphabets d'entrée et de sortie, respectivement. Chaque itération effectue des opérations sur toutes les paires entrée-sortie, nécessitant $\mathcal{O}(M \times N)$ opérations.

Ainsi, la complexité globale de l'algorithme de Blahut-Arimoto est $\mathcal{O}(T \times M \times N)$, reflétant sa dépendance aux tailles de M (nombre de LLMs dans la famille considérée) et N (le vocabulaire), ainsi qu'au nombre d'itérations nécessaires à la convergence, qui dépend intrinsèquement des distributions sous-jacentes.

D Résultats complémentaires

Cette section est dédiée aux résultats complémentaires. La Figure 3 vient compléter la Figure 2 en donnant les résultats obtenus par MOSAIC avec les modèles Falcon sur le jeu de données RAID. La table 6 vient compléter la table 1, pour indiquer que la meilleure combinaison des modèles falcon change selon le jeu de données utilisé. La Table 7 présente les résultats de MOSAIC avec les deux ensembles de modèles sur le jeu de données RAID, ainsi que le mélange uniforme évoqué dans la Section 6.3.

La Table 8 rapporte les résultats de la Section 6.3, utilisant un seul modèle dont les logits sont modifiés via l'échantillonnage nucleus afin de créer différentes distributions de probabilité.

La Table 9 montre la performance de MOSAIC sous les différentes attaques adversariales disponibles dans le jeu de données RAID.

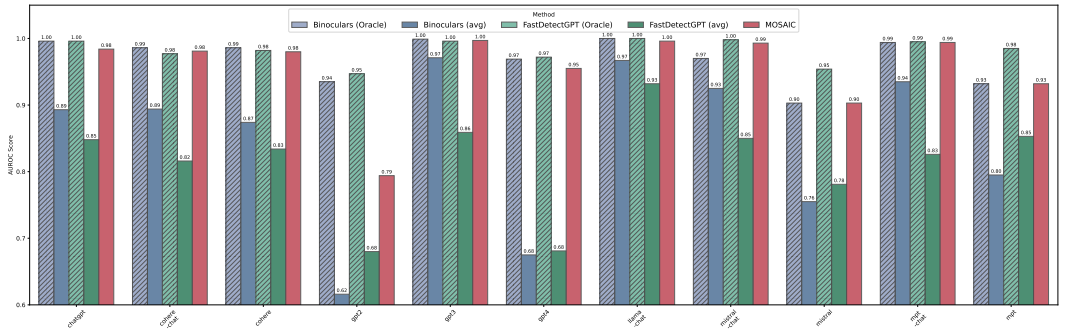

FIGURE 3 – Comparaison des AUROC de MOSAIC sur le jeu de données RAID avec les modèles Falcon. Les noms des modèles représentent le modèle utilisé pour générer le jeu de données.

	chatgpt	cohere-c	cohere	gpt2	gpt3	gpt4	llama-c	mistral-c	mistral	mpt-c	mpt
Bino (best)	0,996	0,986	0,986	0,935	0,999	0,969	1,000	0,999	0,953	0,997	0,976
Config.	7b/7b-i	40b/7b-i	40b/7b-i	7b/7b-i	40b/7b-i	40b/40b-i	7b/7b-i	40b/7b-i	40b/7b-i	7b/7b-i	40b/7b-i
Bino (min)	0,650	0,671	0,606	0,207	0,871	0,293	0,826	0,668	0,436	0,698	0,473
Bino (avg)	0,893	0,894	0,874	0,616	0,971	0,675	0,967	0,925	0,755	0,935	0,795
Fast (best)	0,996	0,977	0,982	0,947	0,996	0,972	1,000	0,998	0,954	0,995	0,985
Config.	40b/40b-i	40b/7b-i	40b/7b-i	7b/7b-i	40b/40b-i	40b/40b-i	40b/40b-i	40b/7b-i	40b/7b-i	40b/40b-i	40b/7b-i
Fast (min)	0,512	0,433	0,504	0,366	0,477	0,343	0,693	0,438	0,522	0,360	0,641
Fast (avg)	0,848	0,816	0,834	0,680	0,859	0,681	0,932	0,850	0,781	0,826	0,853

TABLEAU 6 – AUROC de Bino(culars) & Fast(DetectGPT) sur RAID avec la famille Falcon. Les cellules "best", "avg" et "min" rapportent respectivement le meilleur, la moyenne et le pire score parmi toutes les configurations. Les suffixes “-i” et “-c” désignent respectivement la version instruct et la version chat du modèle.

AUROC	**chatgpt**	**cohere-c**	**cohere**	**gpt2**	**gpt3**	**gpt4**	**llama-c**	**mistral-c**	**mistral**	**mpt-c**	**mpt**
MOSAIC (Falcon family)	0,984	0,981	0,980	0,794	0,997	0,955	0,996	0,993	0,903	0,994	0,932
MOSAIC (uniform Falcon)	0,969	0,947	0,942	0,675	0,987	0,876	0,988	0,970	0,774	0,983	0,796
MOSAIC (Llama and Tower)	0,970	0,980	0,979	0,815	0,997	0,955	0,998	0,988	0,913	0,994	0,927

TABLEAU 7 – Performances de MOSAIC sur les données RAID avec les deux familles de modèles.

E Améliorations de la complexité

Notre algorithme traite actuellement chaque texte en environ 10 secondes sur des GPU NVIDIA V100 32G, et deux fois plus rapidement sur des A100 80G. L’optimisation du temps d’exécution constitue un axe d’amélioration pour de futurs travaux. Ci-dessous, nous décrivons les limitations de notre système et proposons des améliorations potentielles :

Dans MOSAIC, les textes sont traités un par un par les LLMs. Chaque modèle est chargé sur un GPU distinct, et les logits sont transférés vers un dispositif central pour effectuer des opérations telles que Blahut-Arimoto, le calcul de la perplexité et de l’entropie croisée, après quoi le score final est calculé. Cette configuration présente plusieurs inefficacités. Par exemple, le transfert des logits vers un dispositif central constitue un goulot d’étranglement important. De plus, tandis que les calculs s’exécutent sur un seul GPU, les autres restent inactifs, entraînant ainsi une utilisation sous-optimale des ressources.

Une méthode plus efficace consisterait à calculer les logits pour tous les textes en parallèle, à les stocker sur différents GPU, et à réaliser les calculs subséquents de manière simultanée. Une solution

	chatgpt	cohere-chat	cohere	gpt2	gpt3	gpt4	llama-chat	mistral-chat	mistral	mpt-chat	mpt
p = 0,7	0,792	0,665	0,577	0,356	0,733	0,503	0,869	0,763	0,465	0,731	0,488
p = 0,8	0,789	0,665	0,577	0,354	0,732	0,502	0,867	0,760	0,462	0,731	0,485
p = 0,9	0,781	0,663	0,578	0,358	0,725	0,503	0,860	0,752	0,457	0,726	0,480
p = 0,95	0,764	0,656	0,573	0,370	0,711	0,497	0,848	0,736	0,456	0,713	0,476

TABLEAU 8 – MOSAIC sur RAID utilisant Llama-2-7b avec quatre valeurs différentes d’échantillonnage nucleus, sur le jeu de données RAID. Chaque ligne correspond à la valeur choisie de p pour définir m, le « modèle de référence ».

	homoglyph	number	suppression d'article	insertion de paragraphes	fautes d'orthographe	majuscule minuscule	espacement	espace zero-width	synonyme	paraphrase	orthographe alternative
AUROC	0,961	0,936	0,920	0,952	0,948	0,928	0,927	0,754	0,681	0,944	0,947
TPR@5%FPR	0,749	0,736	0,693	0,785	0,771	0,699	0,707	0,007	0,315	0,752	0,771

TABLEAU 9 – AUROC et TPR@5%FPR de MOSAIC pour les différentes attaques effectuées sur RAID, en utilisant les modèles habituels Llama et Tower dans ce scénario. Pour référence, MOSAIC obtient un AUC moyen de 0,956 sur tous les générateurs sans attaques adversariales.

encore plus optimisée impliquerait de charger tous les modèles sur un seul GPU en utilisant des versions quantifiées ou distillées, ce qui éliminerait le besoin de transférer les logits entre dispositifs.

Bien que ces optimisations soient prometteuses, elles n'ont pas été mises en œuvre dans ce travail, puisque nous nous sommes concentrés sur la méthodologie algorithmique plutôt que sur l'efficacité du temps d'exécution.

F Étude systématique de toutes les combinaisons

Les Tables 10, 11, 12 and 13 présentent les résultats de *Binoculars* et FastDetectGPT avec toutes les combinaisons de deux modèles parmi les quatre à notre disposition dans chaque ensemble.

Method	chatgpt	cohere-chat	cohere	gpt2	gpt3	gpt4	llama-chat	mistral-chat	mistral	mpt-chat	mpt	**Average**
Binoculars 0 1	0,99559	0,97838	0,97799	0,93460	0,99786	0,95868	0,99996	0,99919	0,91932	0,99733	0,95045	0,97358
FastDetectGPT 0 1	0,99487	0,96940	0,97366	0,94662	0,99375	0,96133	0,99922	0,99662	0,92746	0,99365	0,96846	0,97500
Binoculars 0 2	0,94267	0,87411	0,85786	0,38375	0,97858	0,68632	0,98387	0,92952	0,59812	0,95399	0,64461	0,80304
FastDetectGPT 0 2	0,86155	0,74420	0,77979	0,49724	0,82475	0,70021	0,94406	0,80315	0,64382	0,77470	0,75213	0,75687
Binoculars 0 3	0,98818	0,92769	0,90545	0,41349	0,98959	0,85435	0,99703	0,98274	0,66670	0,98825	0,69790	0,85558
FastDetectGPT 0 3	0,94627	0,82562	0,83573	0,52014	0,87708	0,82838	0,98374	0,91869	0,69212	0,89539	0,78809	0,82830
Binoculars 1 0	0,70925	0,79631	0,73389	0,49105	0,93632	0,35652	0,88678	0,83358	0,66604	0,82795	0,71482	0,72296
FastDetectGPT 1 0	0,61166	0,61442	0,64173	0,59703	0,68329	0,37331	0,82227	0,65597	0,68932	0,55185	0,80423	0,64046
Binoculars 1 2	0,65047	0,67101	0,60557	0,20704	0,87062	0,29262	0,82634	0,66793	0,43608	0,69845	0,47293	0,58173
FastDetectGPT 1 2	0,51206	0,43272	0,50416	0,36579	0,47719	0,34305	0,69285	0,43790	0,52221	0,35996	0,64190	0,48089
Binoculars 1 3	0,85179	0,76976	0,66409	0,22159	0,90261	0,44492	0,93861	0,85577	0,46290	0,88676	0,49854	0,68158
FastDetectGPT 1 3	0,64419	0,50397	0,52859	0,37228	0,49602	0,42459	0,76787	0,56583	0,52658	0,49533	0,64131	0,54241
Binoculars 2 0	0,97183	0,97627	0,97972	0,82349	0,99761	0,89893	0,99819	0,99345	0,94065	0,99375	0,96866	0,95841
FastDetectGPT 2 0	0,97335	0,96636	0,97593	0,85763	0,99305	0,91412	0,99851	0,98910	0,94400	0,98469	0,98023	0,96154
Binoculars 2 1	0,99459	0,98569	0,98585	0,91758	0,99925	0,96204	0,99993	0,99942	0,95250	0,99732	0,97588	**0,97910**
FastDetectGPT 2 1	0,99470	0,97686	0,98179	0,93533	0,99594	0,96306	0,99993	0,99799	0,95402	0,99441	0,98457	**0,97987**
Binoculars 2 3	0,99417	0,98038	0,97641	0,68528	0,99709	0,96864	0,99807	0,99378	0,86997	0,99491	0,90954	0,94257
FastDetectGPT 2 3	0,99618	0,97637	0,97700	0,72250	0,99601	0,97237	0,99997	0,99405	0,88457	0,99509	0,93609	0,95002
Binoculars 3 0	0,85527	0,93737	0,95387	0,79709	0,99687	0,52522	0,99323	0,96276	0,89972	0,96142	0,94402	0,89335
FastDetectGPT 3 0	0,86263	0,93726	0,95519	0,81322	0,99419	0,52588	0,99398	0,96045	0,91163	0,95339	0,95770	0,89687
Binoculars 3 1	0,91470	0,92920	0,92923	0,88144	0,99245	0,60599	0,99881	0,98299	0,87891	0,97938	0,92556	0,91079
FastDetectGPT 3 1	0,92361	0,93678	0,93578	0,89008	0,99040	0,62426	0,99792	0,98310	0,88995	0,97919	0,93025	0,91648
Binoculars 3 2	0,84538	0,90754	0,91513	0,64079	0,99382	0,54105	0,98113	0,89614	0,76557	0,93707	0,83912	0,84207
FastDetectGPT 3 2	0,85196	0,90944	0,91961	0,64439	0,99140	0,53714	0,98236	0,89891	0,78080	0,93265	0,85672	0,84594

TABLEAU 10 – AUROC sur le jeu de données RAID avec Binoculars et FastDetectGPT utilisant toutes les paires possibles de Falcon. Les modèles utilisés sont représentés par leurs index : Falcon-7b[0], Falcon-7b-instruct[1], Falcon-40b[2], Falcon-40b-instruct[3].

Method	chatgpt	cohere-chat	cohere	gpt2	gpt3	gpt4	llama-chat	mistral-chat	mistral	mpt-chat	mpt	**Average**
Binoculars 0 1	0,98900	0,91600	0,90500	0,69000	0,99200	0,80400	1,00000	0,99600	0,62100	0,99400	0,72900	0,87600
FastDetectGPT 0 1	0,99000	0,88800	0,89700	0,78700	0,98100	0,84400	1,00000	0,99000	0,71500	0,98400	0,85000	0,90236
Binoculars 0 2	0,70200	0,55800	0,38200	0,00900	0,94900	0,08000	0,97600	0,62700	0,00500	0,80600	0,01300	0,46427
FastDetectGPT 0 2	0,26200	0,12900	0,18500	0,00200	0,21400	0,03200	0,62800	0,10600	0,00800	0,10600	0,01900	0,15373
Binoculars 0 3	0,97900	0,72900	0,57400	0,01000	0,98000	0,35000	0,99900	0,95900	0,02400	0,98000	0,03000	0,60127
FastDetectGPT 0 3	0,72500	0,33100	0,29500	0,00200	0,32900	0,21000	0,96200	0,49900	0,01900	0,42200	0,04100	0,34864
Binoculars 1 0	0,10100	0,31800	0,14200	0,01000	0,68700	0,00800	0,54300	0,21500	0,02000	0,28400	0,03200	0,21455
FastDetectGPT 1 0	0,04700	0,07400	0,11400	0,01400	0,10500	0,00600	0,37900	0,05400	0,05300	0,04800	0,10200	0,09055
Binoculars 1 2	0,03100	0,08000	0,03200	0,00600	0,34500	0,00500	0,35700	0,01700	0,00300	0,07900	0,00700	0,08745
FastDetectGPT 1 2	0,00800	0,00100	0,01000	0,00000	0,01000	0,00400	0,00600	0,00000	0,00100	0,00200	0,00300	0,00409
Binoculars 1 3	0,30600	0,29200	0,07200	0,00500	0,50200	0,01400	0,66700	0,23400	0,00300	0,46300	0,01000	0,23345
FastDetectGPT 1 3	0,01600	0,00500	0,01200	0,00000	0,01400	0,00400	0,04100	0,00200	0,00100	0,00200	0,00400	0,00918
Binoculars 2 0	0,89200	0,92800	0,94100	0,35000	0,99600	0,57700	0,99900	0,98500	0,74700	0,98600	0,85900	0,84182
FastDetectGPT 2 0	0,89500	0,88200	0,92200	0,51400	0,97300	0,69400	0,99700	0,96100	0,78600	0,94400	0,91200	0,86182
Binoculars 2 1	0,98700	0,95000	0,95300	0,58700	0,99700	0,85900	1,00000	0,99700	0,78400	0,99500	0,87700	**0,90782**
FastDetectGPT 2 1	0,98500	0,90900	0,92800	0,71900	0,98100	0,85300	1,00000	0,99200	0,79700	0,98300	0,91800	**0,91500**
Binoculars 2 3	0,99300	0,93100	0,92400	0,14300	0,99400	0,89700	1,00000	0,98800	0,52800	0,99500	0,60100	0,81764
FastDetectGPT 2 3	0,99500	0,90800	0,91600	0,28600	0,98500	0,91300	1,00000	0,98500	0,63400	0,98700	0,76400	0,85209
Binoculars 3 0	0,49300	0,82800	0,84400	0,39300	0,99200	0,05500	0,97800	0,86700	0,62400	0,85700	0,77400	0,70045
FastDetectGPT 3 0	0,53400	0,78700	0,83700	0,48700	0,98000	0,09600	0,97500	0,83800	0,68500	0,79500	0,83700	0,71373
Binoculars 3 1	0,71400	0,81700	0,80800	0,56500	0,96900	0,15100	0,99500	0,94600	0,59400	0,94000	0,73600	0,74864
FastDetectGPT 3 1	0,70100	0,78800	0,77800	0,61100	0,95200	0,16400	0,99300	0,91900	0,62900	0,90700	0,77400	0,74691
Binoculars 3 2	0,48200	0,73100	0,72300	0,13800	0,98900	0,04300	0,93800	0,66900	0,30000	0,77800	0,39800	0,56264
FastDetectGPT 3 2	0,51000	0,70700	0,72900	0,21200	0,97200	0,09400	0,93500	0,65400	0,38900	0,71500	0,55800	0,58864

TABLEAU 11 – TPR@5%FPR sur le jeu de données RAID avec Binoculars et FastDetectGPT utilisant toutes les paires possibles de Falcon. Les modèles utilisés sont représentés par leurs index : Falcon-7b[0], Falcon-7b-instruct[1], Falcon-40b[2], Falcon-40b-instruct[3].

Method	chatgpt	cohere-chat	cohere	gpt2	gpt3	gpt4	llama-chat	mistral-chat	mistral	mpt-chat	mpt	**Average**
Binoculars 0 1	0,98868	0,98521	0,97708	0,77221	0,99868	0,95950	1,00000	0,99325	0,87059	0,99826	0,89373	**0,94884**
FastDetectGPT 0 1	0,98843	0,98087	0,97859	0,80717	0,99594	0,96678	1,00000	0,98990	0,89063	0,99503	0,92696	0,95639
Binoculars 0 2	0,78407	0,95574	0,96148	0,80034	0,99533	0,75383	0,98365	0,93690	0,89692	0,96825	0,93239	0,90626
FastDetectGPT 0 2	0,78205	0,95023	0,96258	0,82922	0,99061	0,76525	0,98419	0,93287	0,91102	0,95666	0,95267	0,91067
Binoculars 0 3	0,82575	0,90990	0,91668	0,50181	0,98496	0,62274	0,97900	0,88801	0,76507	0,94356	0,81524	0,83207
FastDetectGPT 0 3	0,77696	0,83897	0,88423	0,57974	0,93333	0,64056	0,95657	0,80964	0,79026	0,82991	0,87509	0,81048
Binoculars 1 0	0,58200	0,79002	0,75954	0,70742	0,97505	0,49584	0,83473	0,69413	0,72506	0,78803	0,74389	0,73597
FastDetectGPT 1 0	0,58753	0,80557	0,76156	0,66687	0,97223	0,45940	0,82525	0,71417	0,70701	0,81873	0,70547	0,72944
Binoculars 1 2	0,56796	0,75465	0,76687	0,72228	0,96559	0,50900	0,80903	0,70925	0,72264	0,74014	0,75186	0,72902
FastDetectGPT 1 2	0,54410	0,77559	0,77060	0,68639	0,96576	0,45812	0,78384	0,71097	0,70689	0,77309	0,71989	0,71775
Binoculars 1 3	0,51082	0,68785	0,71171	0,51711	0,94521	0,37595	0,74094	0,55979	0,60934	0,66094	0,63689	0,63241
FastDetectGPT 1 3	0,50535	0,67341	0,70523	0,50097	0,91377	0,36301	0,73958	0,55247	0,60627	0,64651	0,63589	0,62204
Binoculars 2 0	0,95832	0,93661	0,92122	0,64083	0,99498	0,80933	0,99431	0,97987	0,78711	0,99023	0,78671	0,89087
FastDetectGPT 2 0	0,95451	0,91550	0,92841	0,71961	0,98027	0,84700	0,99345	0,96768	0,84736	0,96719	0,87606	0,90882
Binoculars 2 1	0,99466	0,97298	0,95712	0,66708	0,99810	0,95333	0,99997	0,99584	0,79541	0,99853	0,79537	0,92076
FastDetectGPT 2 1	0,99318	0,95637	0,95732	0,73729	0,98816	0,96584	0,99996	0,99052	0,83830	0,99041	0,86718	0,93496
Binoculars 2 3	0,93911	0,90875	0,87909	0,45877	0,98512	0,71763	0,98668	0,95586	0,69021	0,97678	0,69611	0,83583
FastDetectGPT 2 3	0,90842	0,84184	0,85347	0,55726	0,91478	0,75533	0,97874	0,90606	0,75215	0,89647	0,81150	0,83418
Binoculars 3 0	0,95895	0,95099	0,95582	0,73085	0,99658	0,83482	0,99527	0,98233	0,88173	0,98795	0,90806	0,92576
FastDetectGPT 3 0	0,94743	0,92731	0,94888	0,78517	0,98916	0,86043	0,99348	0,96569	0,90673	0,96440	0,95181	0,93095
Binoculars 3 1	0,99569	0,98342	0,97868	0,74037	0,99948	0,96954	0,99999	0,99752	0,87714	0,99773	0,89556	0,94865
FastDetectGPT 3 1	0,99418	0,96904	0,97333	0,79378	0,99494	0,97381	0,99999	0,99281	0,90178	0,99130	0,94086	**0,95689**
Binoculars 3 2	0,93622	0,95978	0,95667	0,81153	0,99729	0,84474	0,99411	0,98110	0,91501	0,98647	0,94575	0,93897
FastDetectGPT 3 2	0,93160	0,94640	0,95190	0,84095	0,99159	0,86386	0,99445	0,97235	0,92340	0,97124	0,96553	0,94121

TABLEAU 12 – AUROC sur le jeu de données RAID avec Binoculars et FastDetectGPT utilisant toutes les paires possibles de Llama et Tower. Les modèles utilisés sont représentés par leurs index : Llama-2-7b[0], Llama-2-7b-chat[1], TowerBase-7B-v0.1[2], TowerBase-13B-v0.1[3].

Method	chatgpt	cohere-chat	cohere	gpt2	gpt3	gpt4	llama-chat	mistral-chat	mistral	mpt-chat	mpt	**Average**
Binoculars 0 1	0,95300	0,93700	0,89800	0,17300	0,99300	0,77200	1,00000	0,97200	0,38900	0,99500	0,42500	**0,77336**
FastDetectGPT 0 1	0,94900	0,91600	0,90900	0,32400	0,98500	0,84300	1,00000	0,95800	0,55400	0,98300	0,64800	**0,82445**
Binoculars 0 2	0,46000	0,85900	0,87500	0,31200	0,99000	0,28000	0,94300	0,78100	0,55700	0,88500	0,69000	0,69382
FastDetectGPT 0 2	0,44700	0,83100	0,87800	0,44800	0,97300	0,35300	0,94400	0,76400	0,65600	0,82000	0,80900	0,72027
Binoculars 0 3	0,37100	0,66500	0,63700	0,01200	0,97500	0,13700	0,93500	0,51100	0,09400	0,73500	0,10800	0,47091
FastDetectGPT 0 3	0,21900	0,40900	0,55900	0,04800	0,68700	0,11800	0,78400	0,26400	0,22700	0,27900	0,40200	0,36327
Binoculars 1 0	0,07400	0,38000	0,29600	0,17300	0,90300	0,01600	0,22600	0,17500	0,19600	0,27100	0,20600	0,26509
FastDetectGPT 1 0	0,05800	0,34500	0,27900	0,15200	0,88500	0,01500	0,21600	0,14600	0,18600	0,22800	0,19200	0,24564
Binoculars 1 2	0,06600	0,30200	0,29200	0,17400	0,84700	0,02400	0,15200	0,16900	0,19000	0,18100	0,21500	0,23745
FastDetectGPT 1 2	0,06400	0,28700	0,28800	0,16500	0,83700	0,02100	0,16000	0,16000	0,20000	0,16300	0,21900	0,23309
Binoculars 1 3	0,01300	0,11900	0,13500	0,01300	0,71500	0,00700	0,05500	0,02000	0,02600	0,05400	0,03300	0,10818
FastDetectGPT 1 3	0,01200	0,08000	0,12300	0,01800	0,49900	0,00600	0,05900	0,01300	0,03500	0,01900	0,06800	0,08473
Binoculars 2 0	0,76200	0,71000	0,54000	0,03100	0,99000	0,23200	0,99000	0,89700	0,08200	0,96300	0,06200	0,56900
FastDetectGPT 2 0	0,74800	0,66300	0,66400	0,11700	0,89900	0,36900	0,98200	0,84400	0,28500	0,83200	0,36000	0,61482
Binoculars 2 1	0,98900	0,85400	0,76300	0,03200	0,99300	0,71100	1,00000	0,98600	0,12800	0,99700	0,08600	0,68536
FastDetectGPT 2 1	0,97100	0,79600	0,77500	0,11400	0,93900	0,81400	1,00000	0,96400	0,25500	0,96300	0,33900	0,72091
Binoculars 2 3	0,69700	0,63100	0,45000	0,01400	0,97500	0,17000	0,98600	0,76700	0,04000	0,91400	0,02300	0,51518
FastDetectGPT 2 3	0,50400	0,43000	0,43400	0,01500	0,58300	0,17100	0,94100	0,50800	0,11900	0,48200	0,16600	0,39573
Binoculars 3 0	0,80100	0,81800	0,81400	0,11700	0,99700	0,32500	0,99300	0,92500	0,35600	0,97700	0,44000	0,68755
FastDetectGPT 3 0	0,75500	0,74300	0,80400	0,28300	0,96600	0,45100	0,97900	0,85300	0,57000	0,85400	0,76600	0,72945
Binoculars 3 1	0,98800	0,92500	0,90100	0,11000	0,99900	0,83900	1,00000	0,99300	0,32000	0,99400	0,37100	0,76727
FastDetectGPT 3 1	0,97700	0,85900	0,88400	0,27200	0,98100	0,88200	1,00000	0,96900	0,54000	0,97200	0,70000	0,82145
Binoculars 3 2	0,70000	0,85700	0,86600	0,26900	0,99400	0,35600	0,98100	0,91200	0,54500	0,96400	0,69700	0,74009
FastDetectGPT 3 2	0,69200	0,79100	0,83400	0,41200	0,96900	0,45800	0,97400	0,86700	0,64900	0,85300	0,81300	0,75564

TABLEAU 13 – TPR@5%FPR sur le jeu de données RAID avec Binoculars et FastDetectGPT utilisant toutes les paires possibles de Llama et Tower. Les modèles utilisés sont représentés par leurs index : Llama-2-7b[0], Llama-2-7b-chat[1], TowerBase-7B-v0.1[2], TowerBase-13B-v0.1[3].