UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

DÉTECTION AUTOMATISÉE DE TEXTES GÉNÉRÉS PAR L'INTELLIGENCE ARTIFICIELLE

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN SCIENCES ET TECHNOLOGIES DE L'INFORMATION

PAR

COLBERT NGOUANFOUO

et évalué par un jury composé des personnes suivantes :

Prof. Mohand Said Allili ..... Président du jury

Prof. Ana Maria Cretu ..... Membre du jury

Prof. Alan Davoust ..... Directeur de recherche

OCTOBRE 2025

Résumé

Les grands modèles de langage (LLMs) ont révolutionné le domaine de la génération de texte en langage naturel (NLG) en démontrant une capacité impressionnante à produire des textes semblables à ceux rédigés par des humains. Cependant, leur utilisation à grande échelle soulève des défis qui nécessitent une réflexion approfondie, une surveillance éthique et des pratiques responsables. Il devient donc important de détecter les textes écrits par ces grands modèles afin d'éviter toute utilisation préjudiciable et de maximiser leur potentiel. La détection de texte généré par une machine vise donc à identifier un texte comme étant écrit par une machine ou par un être humain. Pour cette tâche de détection, les travaux antérieurs utilisent principalement comme entrée un encodage du texte sous forme d'un vecteur de grande dimension, tel que celui produit par un modèle de type BERT. Cet encodage a l'inconvénient d'être peu interprétable. Ces travaux utilisent aussi certaines caractéristiques liées aux grands modèles de langage tels que la perplexité d'un modèle.

Nous posons la question si les textes générés peuvent être distingués des textes humains par leur grammaire. Pour cela nous proposons une approche qui consiste à encoder la séquence des parties du discours (*part-of-speech,* POS) dans un texte, et d'utiliser la séquence obtenue comme entrée d'un classifieur de type CNN. Dans notre contexte,ces *part-of-speech* (POS) incluent les classes syntaxiques des mots, l'espace entre les mots, la ponctuation et tout autre symbole (/,£,@ *,...).

Nous évaluons la performance de l'approche et la possibilité de la combiner avec d'autres représentations du texte, ainsi que la robustesse de l'approche. Les résultats démontrent que nous pouvons obtenir des performances très compétitives (proche de 90% d'exactitude sur le jeu de données test de la conférence SemEval 2024) seulement en regardant l'agencement des POS, et que les caractéristiques obtenues en considérant les séquences des POS sont complémentaires à une représentation neuronale (vecteur CLS) en ce sens qu'en les combinant, on obtient une amélioration des performances, mieux qu'en consi-

dérant les caractéristiques plus simples comme le comptage des POS. L'évaluation de la robustesse de notre modèle sur des textes ayant subi des attaques (l'insertion des nouveaux paragraphes, l'insertion des carctères spéciaux, la suppression des articles,...) montre que les attaques qui sont censées tromper le classifieur ne semblent pas avoir d'effet notables et que sur la longueur des textes, notre approche est moins performante pour les textes plus courts.

Abstract

Large Language Models (LLMs) have revolutionized the field of natural language generation (NLG) by demonstrating an impressive ability to produce human-like texts. However, their widespread use raises challenges that require careful consideration, ethical oversight, and responsible practices. It has thus become important to detect texts written by these large models in order to prevent harmful uses and to maximize their potential. Machine-generated text detection aims to identify whether a text was written by a machine or by a human. For this detection task, previous work primarily uses as input an encoding of the text into a high-dimensional vector, such as one produced by a BERT-type model. This encoding has the disadvantage of being difficult to interpret. These studies also use certain features related to large language models, such as model perplexity.

We ask whether generated texts can be distinguished from human texts based on their grammar. To this end, we propose an approach that consists of encoding the sequence of parts of speech (POS) in a text and using the resulting sequence as input to a CNN-type classifier. In our context, these parts of speech (POS) include the syntactic classes of words, spaces between words, punctuation, and other symbols (/,£,@,*,...).

We evaluate the performance of the approach, the possibility of combining it with other text representations, as well as the robustness of the approach. The results show that we can achieve very competitive performance (close to 90% accuracy on the SemEval 2024 conference test dataset) solely by examining the arrangement of POS, and that the features obtained from POS sequences are complementary to a neural representation (CLS vector). Combining them yields better performance than using simpler features like POS counts. The evaluation of the robustness of our model on texts subjected to adversarial attacks shows that attacks intended to deceive the classifier do not seem to have a significant effect, and that regarding text length, our approach is less effective for shorter texts.

Dédicace

Ce travail est dédié à ma famille

Remerciements

Je tiens à exprimer mes remerciements les plus sincères à toutes les personnes dont le soutien, les conseils et l'expertise ont grandement contribué à la réalisation de ce mémoire de recherche.

Je souhaite tout d'abord adresser ma profonde gratitude à tous les membres du jury pour m'avoir fait l'honneur de participer à l'évaluation de ce travail.

Je remercie très sincèrement mon encadreur Professeur Alan , qui a dirigé ce travail avec rigueur et bienveillance. Son encadrement attentif, ses orientations scientifiques précieuses ainsi que sa disponibilité constante ont été déterminants à chaque étape de ce projet.

Je tiens à saluer mes professeurs et l'ensemble de l'équipe pédagogique de l'UQO.

Je souhaite également exprimer ma reconnaissance envers ma famille, pour son soutien moral, sa patience et sa confiance inébranlable, qui ont été essentiels au cours de cette période exigeante. Leur présence, discrète mais constante, m'a permis de mener ce travail avec sérénité et détermination.

Enfin, je remercie mes amis, pour leur compréhension, leur soutien et leurs encouragements, qui m'ont accompagnés tout au long de cette importante de mon parcours universitaire.

À toutes et à tous, je renouvelle l'expression de ma gratitude la plus profonde.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations, sigles et acronymes

IA Intelligence Artifficielle

LLM Large Language Models

NLG Natural Language Generation

CNN Convolutional Neural Networks

POS Part-of-speech

NE Named-entity

CLS jeton de classification (*Classification token*)

Chapitre 1

Introduction

1.1 Motivation

L'intelligence artificielle a connu un essor important ces dernières années, notamment en ce qui concerne la génération automatique du langage (NLG). Le progrès des modèles de NLG a renforcé l'aide à l'écriture, telle que la saisie semi-automatique, et a conduit à une écriture plus complexe et contrôlable [63]. Les modèles de langage comme GPT5 sont capables de créer et de modifier (par des *prompts* par exemple) une variété de textes semblables à ceux écrits par les humains. Cette habileté à créer de façon efficiente des textes uniques et vraisemblables à ceux écrits par les humains présente des risques sociétaux comme la désinformation [60, 62, 69], les avis frauduleux sur les produits [62, 1], l'hameçonnage [5, 23], le plagiat [26, 16], et le spam toxique [36]. Afin de limiter les abus de l'utilisation des textes générés par l'IA et de maximiser le potentiel de cette technologie, il devient essentiel de détecter avec précision les textes générés. Les recherches récentes démontrent que les humains sont largement incapables de faire la distinction entre les textes écrits par l'IA et ceux écrits par l'humain [34, 10, 30, 35], donc les méthodes automatisées de détection avec apprentissage ont été développées pour tenter d'apporter une solution à ce problème.

1.2 Problématique

Plusieurs études traitent ce problème comme un problème de classification binaire [31]. Plus précisément, un classifieur est entrainé pour différencier un texte (ou une par-

tie d'un texte) généré automatiquement par l'IA de celui écrit par un humain. Dans ce contexte, les techniques les plus performantes utilisent l'apprentissage profond (basé sur les réseaux de neurones) et les différentes approches de détection utilisent différentes architectures (comme les modèles pré-entrainées) et surtout différentes caractéristiques ou différentes représentations des textes.

Des études récentes s'appuyant sur les modèles neuronaux de langage (MLN) comme RoBERTa consistent à injecter le texte brut directement dans un réseau de neurones similaire à celui utilisé pour la génération de texte. Un *fine-tuning* de ces MLN, permet de capturer implicitement les distinctions textuelles entre les textes générés et les textes humains, grâce à l'ajout d'une couche de classification au-dessus de l'architecture de ces MLN.

Les approches les plus performantes utilisent pour la plupart des représentations "neuronales" du texte comme le vecteur CLS (représentation du premier *token* spécial [CLS] après le passage dans le modèle neuronal), qui ont l'inconvénient d'être "opaques" (on ne sait pas pourquoi le modèle considère le texte comme plutôt artificiel ou humain). Ces représentations sont souvent complétées par des caractéristiques liées aux modèles de langage eux-mêmes (perplexité [1], variation de la perplexité dans le texte, probabilité des jetons,...) ou des caractéristiques linguistiques (mesures de lisibilité, éclatement, longueur des phrases, fréquence de différents mots ou classes de mots (POS, NE...)). L'ensemble de ces représentations forme un vecteur de caractéristiques qui peut être utilisé par les réseaux de neurones (RN) ou les techniques d'apprentissage classiques comme le *support-vector machine* (SVM), le *random forest* (RF) pour faire la classification. En dehors des caractéristiques liées aux modèles de langage, les caractéristiques linguistiques peuvent permettre la distinction des textes générés des textes humains ou d'améliorer les performances des classifieurs [18]. Ceci suggère que le texte produit par les LLM est (linguistiquement) différent du texte produit par des personnes, et que ces différences peuvent être, jusqu'à un certain point, capturées par des éléments linguistiques, qui présentent l'avantage d'être plus interprétable [41, 64] : tel un médecin qui fait son diagnostic, on peut déclarer un texte comme artificiel en indiquant les signes qui le révèlent. Les éléments grammaticaux (classes grammaticales des jetons) ont un fort potentiel pour différencier les textes générés et humains [41, 65, 44], et on cherche

1. La perplexité d'un modèle linguistique sur un texte est l'inverse de la probabilité du texte, normalisée par le nombre de mots.

à construire des modèles pour les exploiter.
Nous posons la question suivante :

De quelle façon peut-on exploiter les artéfacts grammaticaux pour faire de la classification des textes générés par l'IA ?

Dans la littérature, les caractéristiques issues des éléments grammaticaux sont obtenues par comptage[64, 18, 44] (nombre de noms, d'adjectifs, de verbes...).
Nous faisons l'hypothèse qu'il serait bénéfique de tenir compte de la disposition des *tokens* dans un texte, au sens de l'ordre, ou d'utiliser les *patterns* séquentiels pour discriminer les textes générés des textes humains. Pour valider cette hypothèse, il faut voir comment tenir compte de l'ordre en identifiant des *patterns* séquentiels, et deuxièmement, faire des expériences pour évaluer la pertinence de notre approche. Plus concrètement, il faudra répondre aux questions suivantes :

- **(Q1.)** Comment peut-on tenir compte de la disposition dans le texte, au sens de l'ordre ou des *patterns* séquentiels des POS, pour distinguer les écrits par les humains des textes générés par les machines ?
- **(Q2.)** Peut-on combiner l'approche issue de la question précédente (l'utilisation des *patterns* séquentiels des POS) avec les approches existantes (représentations neuronales) pour obtenir des meilleures performances ?
- **(Q3)** Comment se comparent les résultats obtenus en prenant en compte les caractéristiques obtenues par comptage des POS et avec ceux obtenus en tenant compte des caractéristiques obtenues avec les *patterns* séquentiels ?
- **(Q4)** Comment se comporte notre approche par rapport à la robustesse ? Robustesse : attaques et variation de la longueur des textes ?

.

1.3 Contributions

Dans le cadre de ce mémoire de recherche, nous avons construit plusieurs modèles de détection de textes générés en anglais par les systèmes de dialogues comme Chatgpt ou Gemini, basés à la fois sur les caractéristiques du texte et sur les modèles neuronaux. Ces modèles ont pour but de démontrer la pertinence des *patterns* grammaticaux. Pour répondre à la question **(Q1.)**, nous avons conçu une représentation du texte qui utilise

les caractéristiques linguistiques (les POS et plus), mais qui garde l'agencement séquentiel de ces éléments dans le texte. Pour ce faire, nous avons utilisé les CNN (réseaux de neurones convolutifs) à une dimension pour obtenir ces caractéristiques. Ce type de réseaux de neurones donne de bons résultats pour reconnaître les motifs spatiaux en vision. Nous avons utilisé ces caractéristiques dans notre modèle (modèle CNN+POS). Ces caractéristiques nous aident à nous concentrer sur les différences fondamentales entre les textes générés et les textes écrits par l'humain plutôt que de capturer les spécificités de modèles. Nous avons obtenu une exactitude moyenne (sur 20 entrainements) de 87,19% sur l'ensemble de test (jeu de données de la conférence SemEval 2024 [68] .) et notre meilleure performance est de 92%. Ceci démontre qu'on peut obtenir des performances très compétitives seulement en regardant des *patterns* dans l'agencement des classes de mots (POS), qu'on peut capturer à l'aide d'un CNN. Afin de situer ces performances par rapport à d'autres techniques, nous les avons comparées à deux modèles : un obtenu par *fine-tuning* de RoBERTa (*baseline* fort) et un modèle commercial réalisé par une *start-up* spécialisée (performance de pointe). Avec RoBERTa, nous avons obtenu une exactitude moyenne de 85,74% sur l'ensemble de test avec 92% comme meilleure performance, et avec GPTzero (un détecteur commercial), nous avons obtenu une exactitude de 94% [2] sur l'ensemble de test. Les performances de notre approche semblent assez compétitives par rapport à celles du *baseline*. Quoique ces performances soient assez compétitives, elles ne sont pas les meilleures au vu de l'état de l'art, ce qui conduit à poser la suivante : peut-on combiner l'approche issue de la question précédente (l'utilisation des *patterns* séquentiels des POS) avec les approches existantes (représentations neuronales) pour obtenir des meilleures performances ? (question **(Q2.)**).

Pour répondre à cette question, nous avons aussi comparé les performances de notre méthode avec la performance d'un modèle obtenu en combinant les caractéristiques issues de notre approche aux vecteurs CLS (modèle CNN+CLS) et avons passé l'ensemble des caractéristiques dans un réseau de neurones entièrement connectés et nous avons constaté une amélioration des performances (une exactitude moyenne de 92,87% sur 20 entrainements avec 95,16% pour notre meilleur modèle CNN+CLS). Ceci démontre que les caractéristiques grammaticales capturées par notre représentation sont complémentaires

2. Les résultats « bruts » de GPTZero sont optimisés pour éviter les faux positifs, avec une précision supérieur à 99% sur les textes générés, mais seulement 80% sur les textes humains. Par conséquent l'exactitude est assez faible, 88% envion. Nous avons utilisé les scores de confiance retournés par l'API pour évaluer la performance qui optimise l'exactitude, et celle-ci atteint alors environ 94% sur le jeu de données de test de semEval.

à celles capturées par une représentation sémantique du texte. Est-ce qu'un traitement plus simple des POS (le comptage) aurait suffi à capturer cette information complémentaire, et aurait aussi amélioré les performances obtenues à partir de la représentation sémantique ?

Cette question nous amène à répondre à la question (**(Q3)**) : comment se comparent les résultats obtenus en prenant en compte les caractéristiques obtenues par comptage des POS et avec ceux obtenus en tenant compte des caractéristiques obtenues avec les *patterns* séquentiels ?
Pour répondre à cette question, nous avons comparé les modèles CLS+CNN et CLS+comptage (obtenus en associant les vecteurs caractéristiques par comptage des POS aux vecteurs CLS) afin de confirmer la pertinence des caractéristiques issues de notre approche. Nous avons obtenu 92,27% pour le meilleur modèle CLS+Comptage contre 95,16% pour notre meilleur modèle CNN+CLS, ce qui confirme que les caractéristiques issues de notre approche sont plus pertinentes que les caractéristiques par comptage des POS.

Enfin, nous avons voulu évaluer la robustesse de notre approche et la comparer à celle des approches concurrentes (question **(Q4)**). Pour répondre à la question **(Q4)**, nous avons exploré les performances de notre approche sur la variation de la longueur des textes des jeux de données. Nous avons également évalué la robustesse de notre approche sur les textes ayant subi des attaques qui sont conçues pour tromper le classifieur. Nous avons constaté que les attaques ne fonctionnent pas avec notre approche. Cela peut paraitre surprenant et on peut aussi s'en féliciter. Cependant, nous avons constaté qu'elles ne fonctionnent pas non plus sur les modèles RoBERTa et GPTzero. Notre approche est aussi moins performante sur les textes courts.

1.4 Plan du mémoire

Le reste du mémoire est structuré comme suit : le deuxième chapitre, dans lequel nous définissons les concepts utilisés et faisons une revue de la littérature, le troisième chapitre qui présente la méthodologie utilisée et dans le quatrième qui présente la validation de notre approche. Le dernier chapitre est consacré à la conclusion et aux travaux futurs.

Chapitre 2

Rappels et état de l'art

Dans ce chapitre, nous allons rappeler les quelques notions nécessaires à la compréhension de notre travail, à la génération et à la détection des textes produits par l'IA. Nous ferons aussi le tour de quelques travaux déjà effectués dans ce domaine.
Pour faire la détection des textes écrits par l'IA, il est important de comprendre les différentes méthodes de génération de textes.

2.1 Génération automatique de texte en langage naturel

2.1.1 Définitions et exemples

Pour commencer, il est important de distinguer un langage naturel d'un langage non naturel. Nous allons ensuite définir le terme *texte généré par une machine*. Il faudra également définir la génération des textes ainsi que ses méthodes.

Définition 2.1. Un *langage naturel* est un langage humain, qui, au sens opérationnel, est acquis naturellement en association avec la parole [43], par opposition au langage non-naturel ou langage formel comme les langages de programmation en informatique.

Définition 2.2. *Un texte généré par une machine* est un texte en langue naturelle produit, modifié ou étendu par une machine. [14]

Il ressort de cette définition que la génération des textes utilise un langage non-naturel et peut couvrir différents domaines.

2.1.2 Exemples de génération de textes

La génération des textes en langage naturel dépend des données d'entrées (*inputs*) et de la tâche qu'on veut réaliser. Plusieurs modèles ont été entrainés pour différentes tâches. Le tableau 2.1 nous donne quelques exemples de tâches avec des exemples de modèles pour réaliser ces tâches. Il faut noter dans ces exemples que *les inputs*, les

TABLEAU 2.1 – Inputs, tâches et exemples de modèles

Inputs	**Tâches**	**Exemples de modèles**
Pas d'entée ou bruit aléatoire	Génération inconditionnelle	GPT2 [53], GPT3 [8] (sans *prompt*)
Séquence de texte	Génération conditionnelle de texte	GPT2 [53], GPT3 [8] (avec *prompt*), T-[54]
	Traduction automatique	FairSeq-[51],T- [54]
	Transfert de style de texte	Dictionnaire de style [59], GST [59]
	Synthèse de texte	BART-RXF [2], Fréquence des mots et des phrases [42]
	Réponse aux questions	FairSeq-[51],T-[54]
	Système de dialogue	DG-AIRL [39], DIALOGPT [70], Blenderbot3, ChatGPT [57]
Attributs discrets	Génération basée sur les attributs	MTA-LSTM [20], PPLM [15], CTRL [33]
Données structurées	Génération de texte à partir de données	DATATUNER [27],Control prefixe (T5) [11]
Données multimédia	*Image captioning*	GIT [67],ETA [37]
	Video captioning	MMS [38],Youtube2Text [24]
	Reconnaissance vocale	ARSG [9], wav2vec-U [3]

tâches et les modèles du tableau 2.1 ne sont pas exclusivement liés. Par exemple, CTRL peut prendre, en entrée, à la fois un attribut de code de contrôle discret et une invite de

texte conditionnelle dans la génération [33].
Dans notre travail, nous nous intéresserons aux textes en anglais générés par les LLM à partir des prompts.

2.2 La génération des textes

Le problème de génération de texte peut être posé comme un problème de modélisation du langage.

Définition 2.3. *Un modèle de langage* est une fonction, ou un algorithme d'apprentissage d'une telle fonction, qui capture les principales caractéristiques statistiques de la distribution des séquences de mots dans un langage naturel, permettant généralement de faire des prédictions probabilistes du mot suivant en fonction des mots précédents.[6]

La génération de texte utilise des algorithmes et des modèles de langage pour traiter les données d'entrée et générer un texte de sortie. Les modèles génératifs de langage sont entraînés sur un ensemble existant d'exemples de textes de longueur variable $(x_1, x_2, ..., x_n)$, chacun composé de symboles $(s_1, s_2, ..., s_m)$. Dans la pratique, les modèles génératifs génériques de langage fondent leurs prédictions de mots sur des mots générés précédemment, soit directement, soit sous la forme d'un état caché codant des informations contextuelles [6, 69]. Par conséquent, la probabilité conditionnelle d'une séquence $x = (s_1, s_2, ..., s_m)$ donnée est donnée par la relation de récurrence suivante, qui la relie aux probabilités conditionnelles de tous les mots précédents :

$$p(x) = p(s_1)p(s_2|s_1)p(s_3|s_1, s_2)...p(s_m|s_1, ..., s_{m-1}) = \prod_{i=1}^{m} p(s_i|s_1, s_2, ..., s_{i-1}) \qquad (2.1)$$

Ainsi, pour générer intuitivement un texte, un modèle génératif utilise le contexte précédent pour estimer une distribution de probabilité sur le vocabulaire du modèle, qu'il suffit ensuite de décoder en échantillonnant le jeton suivant à partir de cette distribution. La figure 2.1 permet de visualiser la génération de texte lors de l'échantillonnage. Le mot *'car'* est échantillonné à partir de la distribution de probabilité conditionnée $p(w|\,The)$, suivi d'un échantillonnage *'drives'* à partir de $p(w|'The',' car')$. Avant l'avènement des modèles neuronaux, une façon d'estimer les probabilités conditionnelles de l'équation (2.1) est l'utilisation de modèles *n-grammes* reposant sur l'hypothèse markovienne selon

1. Tiré du site internet `https://huggingface.co/blog/how-to-generate`

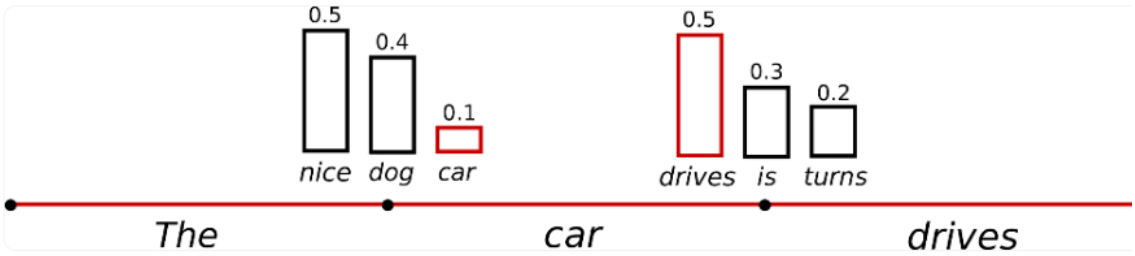

FIGURE 2.1 – Échantillonnage [1].

laquelle la probabilité d'un mot peut être approximée en calculant la probabilité conditionnelle des n mots qui le précèdent. Par exemple, dans un modèle 4-gramme (quatre éléments successifs), pour l'expression *« je vais au »*, le modèle pourrait prédire avec une forte probabilité le nom d'un endroit s'il a été entrainé sur un corpus de textes contenant les lieux.
Les modèles n-grammes présentent quelques limites. Par exemple, les 2-grammes donnent des prédictions moins précises, tandis que des modèles n plus grands diminuent la probabilité de trouver une séquence particulière de mots dans un corpus, ce qui produit des données manquantes. Un autre inconvénient des modèles n-grammes est leur tendance à négliger toute information qui n'est pas contenue dans le voisinage immédiat d'un mot cible, ignorant largement certains types de structures syntaxiques et ne parvenant pas à maintenir la continuité sémantique sur des séquences plus longues [29]. Une solution à ces limites des modèles n-grammes a été l'utilisation des approches neuronales.

2.2.1 Approches neuronales

Les modèles neuronaux de génération de textes exploitent les prouesses du *deep learning* en donnant une meilleure estimation des probabilités de l'équation (2.1) pour produire des textes plus cohérents et proches du langage humain. Nous distinguons ici deux familles de modèles neuronaux de génération de textes : les modèles sans *transformer* et les modèles avec *transformer*.

Les modèles neuronaux sans transformer

Les réseaux de neurones récurrents (RNN) [32, 49, 7] et les architectures à mémoire à long terme (LSTM) [48] se sont montrés très efficace dans la génération des textes en langage naturel. Cependant, les RNN et LSTM sont confrontés au problème de la

disparition du gradient (*vanishing gradient*) dans lequel la mise à jour des paramètres d'apprentissage devient très lente (car le gradient tend vers 0), résultant ainsi une convergence lente ou un échec de la convergence [14].

Les modèles neuronaux avec *transformer*

Les modèles de génération de langage qui prédominent actuellement sont basés sur l'architecture de *transformer* [66]. Le grand avantage des *transformers* est une mémoire plus structurée pour les dépendances à long terme et sa résilience au problème de la disparition du gradient grâce au mécanisme de *multi-head attention* [66]. Les modèles unidirectionnels gauche-droite (comme GPT-2 [53] et GPT-3 [45]) sont les plus étudiés en raison de leur performance pour générer un texte plus cohérent [58], et sont des éléments clés de la recherche sur la détection de textes générés par des machines. En plus de GPT-2 et GPT-3, il y a des modèles autoregréssifs apparentés qui utilisent une architecture similaire à GPT-2 et GPT3 et qui sont remarquables par différentes procédures d'échantillonnage et par leurs données d'entrainement. Parmi ces modèles, on peut citer : Grover, GPT-J, GPT-NeoX.
Le mécanisme de *self-attention* de l'architecture Transformer permet d'entrainer des architectures de réseaux neuronaux capables d'estimer efficacement les distributions de probabilités de l'équation (2.1), moyennant une tâche de pré-entrainement appropriée.
L'un des problèmes posés par l'échantillonnage du prochain jeton selon la distribution de probabilité est que le modèle peut être moins créatif et générer un texte incohérent. Un paramètre important pour améliorer l'échantillonnage est la "température" $T \in (0, \infty)$. La température est utilisée pour augmenter la probabilité d'avoir des jetons probables tout en réduisant celle qui ne l'est pas ($0 < T < 1$). Lorsque $T = 1$, il n'y a pas d'effet et pour $T > 1$, le modèle augmente la probabilité de sélectionner un jeton suivant moins probable afin améliorer la diversité au prix potentiel du choix d'un jeton inhabituel.
En plus de la température, il existe trois stratégies de décodage courantes utilisées pour échantillonner les probabilités de jeton m à partir des modèles *transformer* génératifs unidirectionnels contemporains [28] :

- Échantillonnage aléatoire sans troncature : il s'agit d'échantillonner la totalité de la distribution de probabilité.
- Échantillonnage Top-K (*top-k truncation*) : l'échantillonnage Top-K est utilisé pour s'assurer que les mots les moins probables ne devraient avoir aucune chance ;

– Échantillonnage du noyau (*Nucleus sampling*) : l'échantillonnage du noyau se concentre sur les plus petits ensembles possibles de mots $V(p)$ tels que la somme de leur probabilité soit supérieure ou égale à p. Ensuite, les jetons qui ne sont pas dans $V(p)$ sont mis à 0 ; les autres sont redimensionnés pour s'assurer que la somme est égale à 1.

Un aspect important des modèles produits par les modèles *transformer* est la possibilité de façonner la sortie du modèle au moyen du *prompt engineering* (en élaborant soigneusement l'entrée de texte conditionnelle pour qu'un modèle de langage puisse continuer à fonctionner [45]) ou de la fourniture des attributs discrets supplémentaires utilisés pour influencer la génération du réseau tels que le code de contrôle, le sujet ou le sentiment.

2.3 Les différentes approches de détection des textes générés par l'IA

L'émergence des *Large Language Models* (LLM) a favorisé la génération des textes vraisemblables aux écrits humains. Cela vient avec une utilisation abusive de ces textes telle la désinformation, la fraude, l'hameçonnage... La détection des textes produits par l'IA devient un moyen mesurable pour réduire ces abus. La détection des textes générés par l'IA consiste à entrainer un classifieur à différencier un texte généré par une intelligence artificielle d'un texte écrit par un humain. Dans le cadre de notre travail, il s'agit donc d'une classification binaire qui consiste à détecter si un texte en anglais est généré par un LLM ou écrit par un humain. Notre méthode est transférable à d'autres langues moyennant quelques travaux supplémentaires.

Dans la littérature, nous distinguons deux grandes approches : les approches basées sur les caractéristiques (*features*) qu'il faut extraire des données au préalable et les approches basées sur le texte brut sans extraction de caractéristiques.

2.3.1 Approches récentes

Pour distinguer les textes synthétiques des textes rédigés par les humains, les approches récentes utilisent majoritairement les modèles de langage pré-entrainés comme RoBERTa ou les LLM, avec un *fine-tuning* des modèles de classification ou encore des modèles d'ensemble. Ces modèles se basent sur les caractéristiques issues des modèles de langage et/ou des caractéristiques des textes bruts.

Dans le cadre de la conférence SemEval 2024 [68], l'équipe Areg Mikael Sarvazyan, José Ángel González, et Marc Franco-Salvador [55] a obtenu la meilleure performance de la tâche de détection des textes écrits par l'IA en anglais en extrayant des caractéristiques probabilistes au niveau du jeton (probabilité logarithmique et entropie). Cette équipe a en utilisé quatre modèles LLaMA-2 : LLaMA-2-7B, LLaMA-2-7Bchat, LLaMA-2-13B, et LLaMA-2-13B-chat, LLaMA étant un LLM autoregréssif open source développé par Meta. Les caractéristiques obtenues ont ensuite été introduites dans un encodeur *transformer* entrainé de manière supervisée.
Dans le cadre de cette même conférence, Guo et al.[25] ont exploité RoBERTa pour extraire les caractéristiques, en utilisant le vecteur de sortie correspondant au jeton `[CLS]` comme représentation sémantique du texte d'entrée. Ces auteurs ont ajouté au-dessus de Roberta un perceptron multicouche pour la classification binaire (texte humain ou machine).
Dans le cadre du concours AuTexTification (*Automated Text Identification*) qui a eu lieu dans le cadre de IberLEF 2023, une équipe de chercheurs (Przybyla, Duran-Silva, et Egea-Gomez)[52] a obtenu le meilleur score (80,91% de F1) [56] en construisant un détecteur qui repose sur le modèle LSTM bidirectionnel entrainé sur une combinaison de caractéristiques probabilistes au niveau des jetons provenant de différentes versions de GPT-2, de caractéristiques linguistiques au niveau des jetons telles que la fréquence des mots ou les erreurs de grammaire, et de représentations de texte provenant d'encodeurs pré-entrainés.

2.3.2 Les approches basées sur les caractéristiques

Ces approches utilisent des caractéristiques soigneusement élaborées pour les propriétés statistiques des textes, sous l'hypothèse que ces propriétés diffèrent pour les textes humains et les textes générés par les machines. Elles consistent à utiliser les techniques du NLP (*Natural Language Processing*) pour créer les vecteurs de caractéristiques à partir d'une séquence de mots, et à utiliser ensuite les algorithmes comme le SVM, les forêts aléatoires ou les réseaux de neurones pour faire de la classification. Les catégories de caractéristiques les plus importantes qui ont été proposées sont la fréquence des mots, les caractéristiques linguistiques issues des modèles auxiliaires, la fluidité et les caractéristiques de base d'un texte telles le nombre de ponctuations, la longueur des phrases ou des paragraphes, le nombre de paragraphes [14].

Les caractéristiques de base des textes

Les caractéristiques de base sont des comptages simples de caractères, de syllabes, de mots et de phrases, à la fois en termes absolus et relatifs. Elles peuvent inclure le nombre de signes de ponctuation ou la longueur des phrases et des paragraphes qui ont été utilisés pour détecter les textes générés par une machine [22].

Caractéristiques lexicales : la fréquence des mots

Il s'agit d'utiliser la fréquence des mots dans l'échantillon à classifier sans tenir compte de leur ordre dans le texte. Il existe plusieurs caractéristiques basées sur la fréquence.

La loi de Zipf Selon la loi de Zipf, dans un texte écrit par un humain, la fréquence d'un mot est inversement proportionnelle à son rang dans la liste globale des mots après le tri par ordre décroissant de fréquence [72, 73]. Elle suit la relation :

$$f \approx \frac{\frac{1}{k^s}}{\sum_{n=1}^{N} \frac{1}{n^s}}$$

où k est le rang du mot parmi les N que compte le corpus et $s \in R$, $s \geq 1$, est un exposant qui caractérise la distribution. L'exposant s est approximativement égal à 1. En appliquant le logarithme au deux membre de la relation précédente, on obtient :

$$\log(f) \approx -s \log(k) - \log(\sum_{n=1}^{N} \frac{1}{n^s})$$

Cette dernière équation peut être modélisée comme une régression linéaire et nous pouvons vérifier que la loi de Zipf s'applique si le coefficient de la variable $log(k)$ est approximativement égal à -1 ($s \approx 1$). Ainsi, pour un texte donné à classer, il faut :

- transformer le texte en mots (*tokeniser*) afin de constituer un "sac de mots"
- déterminer le nombre d'occurrences de chaque mot du "sac de mots" et ensuite, associer des rangs aux mots (figure 2.2). Prendre les 100 premiers mots par exemple.
- créer un modèle de régression linéaire comme indiqué dans la dernière équation ci-dessus et trouver la valeur de la pente a de la régression.

2. Tiré du site internet `https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/09-text-as-vectors-bow-tfidf`

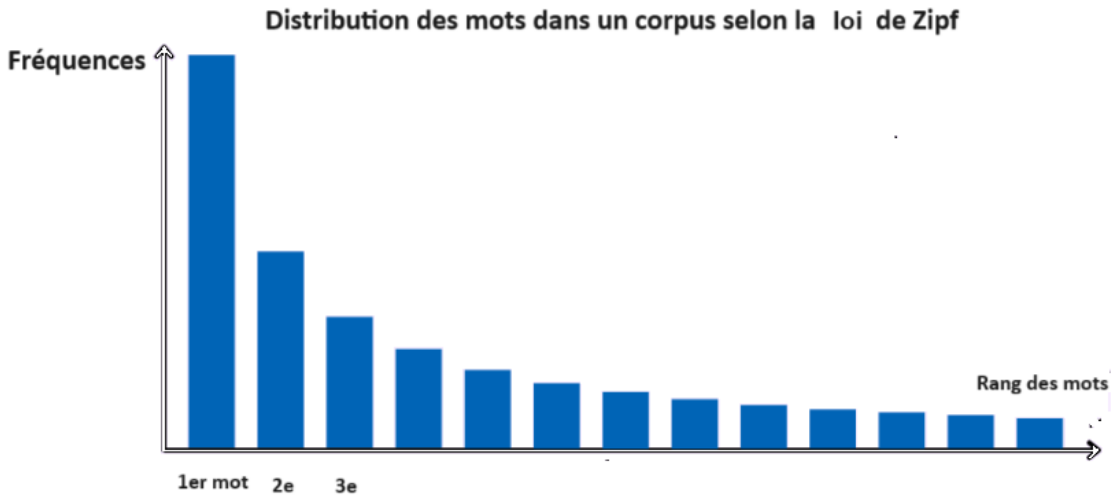

FIGURE 2.2 – Loi de Zipf[2].

La distribution des mots pour les textes générés par l'IA est différente de celle des textes humains. Cette différence est plus prononcée lorsque le volume des textes est important. La pente a devrait donc être différente pour les deux types de texte. La pente a est ainsi considérée comme une caractéristique majeure pour la détection des textes générés par une machine.

Caractéristique basée sur les lemmes Les lemmes sont utilisés comme caractéristique de classification[50, 13] à la place des mots dans un texte. Un lemme est la forme canonique ou la racine d'un mot variable. À titre d'exemples :

- Les mots parle, parles, parlons, parlez, parlent, parla,... se rapportent au lemme *parler*.
- les mots chat, chats, chatte, chattes se rapportent au lemme *chat*

Pour cette approche, pour chaque texte à classer, les lemmes sont d'abord identifiés, ensuite la fréquence de chaque lemme calculée ainsi que son rang. Afin de vérifier la compatibilité de la distribution des lemmes et la distribution de la loi de Zipf, un modèle de régression linéaire entre le log-log de la fréquence des lemmes est réalisé ($y = ax + b$, $y = log(f)$ et $x = log(k)$) et la pente calculée. On peut mesurer la perte de l'information

de la régression en calculant la fonction de coût de l'erreur quadratique moyenne :

$$C(y, \widehat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \widehat{y}_i)^2$$

Où n est le nombre de lemmes distincts, $\widehat{y}_i$ la prédiction pour le lemme de rang i, y_i est la vraie valeur $logf_i$ de la fréquence du lemme de rang i.
La pente a (comme pour la loi de Zipf) et l'erreur quadratique sont des caractéristiques pour distinguer les textes humains des textes générés par les machines [13].

Caractéristique TF-IDF Le TF-IDF (*term frequency-inverse document frequency*) est une autre caractéristique basée sur la fréquence qui permet de mesurer l'importance ou la pertinence des mots ou des phrases dans un document, parmi les documents d'un corpus. Il s'agit d'extraire dans un premier temps "le sac de mots" de tous les textes réunis (corpus) et ensuite calculer le poids de chaque mot dans chaque ligne de données. Elle est composée de deux parties : une partie TF (term frequency) et une partie IDF(inverse document frequency). Le TF et l'IDF d'un mot t dans un document d et dans un corpus D se calculent comme suit :

$$tf(t, d) = \frac{\text{nombre d'occurrences de } t \text{ dans le document } d}{\text{nombre total de mots dans le document } d}$$

$$idf(t, D) = \log\left(\frac{N}{cardinal\{d \in D : t \in d\}}\right)$$

Où N est le nombre de documents d dans le corpus D.

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Par exemple, dans le corpus suivant :

corpus = ['This is the first document.',
'This document is the second document.',
'And this is the third one.',
'Is this the first document ?']

Dans cet exemple, le vocabulaire associé est : *'and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this'*. Il faudra calculer le TF et l'IDF de chaque mot du vocabulaire pour chaque donnée (document) et ensuite calculer le TF-IDF. On pourra obtenir un

	mot_1	mot_2							mot_9	classes
d_1	$tfidf_{11}$								$tfidf_{19}$	humain
d_2										machine
d_3										
d_4	$tfidf_{41}$								$tfidf_{49}$	humain

TABLEAU 2.2 – caractéristiques TF-IDF

tableau avec tous les textes (documents) et les TF-IDF pour chaque mot du "sac de mots" (tableau 2.2). En ajoutant une colonne "classes" au tableau qui donne le caractère texte humain ou IA, on obtient un tableau complet de données.
Les caractéristiques des un-grammes et des 2-grammes de TF-IDF ont été utilisées avec un détecteur basé sur la régression logistique et ont été utilisées comme base de référence pour la détection [53, 61] ou comme les caractéristiques statistiques [22].
Une autre approche qui a été aussi utilisée est la fréquence des chevauchements de n-grammes de mots et de *POS* entre les phrases consécutives [22]. À titre d'exemple, Leon Fröhling et Arkaitz Zubiaga [22] ont évalué la répétitivité des un-grammes, des 2-grammes ou des 3-grammes autour de la conjonction *and* (dans l'expression, *the text and the text*, on a un 2-gramme autour de *and*). Pour cette approche, on s'attend à ce que les textes humains soient moins répétitifs, tant au niveau de la structure des phrases que du choix des mots [22].

Caractéristiques de fluidité ou de lisibilité

Les caractéristiques de lisibilité reflètent la complexité syntaxique, la cohésion et la sophistication du vocabulaire d'un texte [12]. Ces caractéristiques sont importantes pour les textes générés par les machines. Plus les phrases sont longues, moins les machines semblent produire des textes consistants et cohérents [28, 58]. L'indice de Gunning-Fog et l'indice de Flesch [21] fournissent respectivement les mesures statistiques de la lisibilité et de la compréhensibilité et ont un pouvoir prédictif.[13].

Autres caractéristiques linguistiques

La cohérence pour les textes générés a été mesurée dans certains travaux [50, 13] en déterminant la fréquence relative du nombre de verbes à particules et ainsi que du nombre de relations de résolution des coréférences par rapport au nombre de mots au sein d'un échantillon de texte. Ces valeurs sont considérées comme des caractéristiques

de cohérence.
Il y a une différence de distribution de *part-of-speech* (POS) et *Named-entity* (NE) entre les textes humains et les textes générés par l'IA [53, 58]. Ainsi, la distribution relative des POS et de NE, leur nombre par phrase et un certain nombre de caractéristiques simples basées sur le comptage ont utilisé pour discriminer les textes IA et les textes humains [22].
Il faut noter que dans la littérature, la différence de distribution de POS n'a pas été exploitée que dans un travail incluant un grand nombre d'autres caractéristiques, et seulement sous forme de comptage. Feng et al.[19] ont regroupé les mots en fonction de leurs étiquettes de POS. Pour chaque classe de mots, ils ont mis en œuvre cinq caractéristiques. Par exemple, pour la classe des adjectifs, ils ont implémenté les cinq caractéristiques suivantes : pourcentage d'adjectifs (tokens) par document, pourcentage d'adjectifs uniques (types) par document. Ils ont ainsi obtenu des vecteurs de nombre leur permettant de faire la classification.

2.3.3 Approches par les modèles neuronaux basées sur le texte brut

Pour ces approches, les textes bruts sont passés directement dans un RN et celui-ci trouvera lui-même les caractéristiques.
Les méthodes de détection basées sur les réseaux neuronaux, en particulier celles qui exploitent les caractéristiques extraites du modèle de langage neuronal (NLM) Transformer, font preuve d'une grande efficacité dans la détection des textes générés par des machines. Il existe deux principales approches pour les méthodes basées sur les modèles de langue naturelle (NLM) : la classification sans entraînement préalable (approche *zero-shot*) en se basant sur des modèles existants, et le *fine-tuning* en se basant sur des modèles de langage pré-entraînés. Ces derniers constituent la majorité importante des méthodes de classification automatique de textes basés sur les NLM.

Approche *zero-shot*

Elle consiste à entrainer un *classifieur* sur un ensemble de données étiquetées et à faire des prédictions pour des étiquettes (*labels*) qu'il n'a jamais vues auparavant, c'est-à-dire des échantillons pas présents dans les données d'entrainement. Par exemple, un modèle entrainé pour classifier les chats et chiens peut classifier avec précision des lapins

en utilisant l'approche *zero-shot* même-si lors de l'entrainement, le modèle n'a pas connu des lapins. Ceci est possible grâce aux caractéristiques (couleurs de pelage, habitat, alimentation,...) associées aux différentes classes qui permettent de combler le fossé entre les catégories connues et inconnues.
L'approche *zero-shot* peut être appliquée au NLP. En effet, à la base, les modèles génératifs (à l'instar de GPT-2 ou de Grover [61, 53, 69]) peuvent faire de la classification. Ces modèles génératifs peuvent être utilisés sans *fine-tuning* pour identifier si un texte provient d'eux ou d'un modèle similaire.
Les modèles génératifs auto-régressifs comme GPT-2, GPT-3 et Grover sont unidirectionnels, chaque mot (*token*) a une représentation (*embedding*) dépendante de la représentation du mot précédent. La représentation d'une séquence de mots est donc un vecteur dont chaque composante est une représentation des mots de la séquence représentés l'un après l'autre. Il en résulte que la représentation d'une séquence de mots peut être obtenue en ajoutant un jeton de classification `[CLS]` à la fin de cette séquence. Il s'agit d'un jeton qui représente l'ensemble de la séquence d'entrée. La représentation vectorielle de ce jeton de classification (`[CLS]`) peut être utilisée comme vecteur de caractéristiques de toute la séquence.
Les modèles d'apprentissage *zero-shot* peuvent donc apprendre de ces vecteurs de caractéristiques sur les données étiquetées par les textes humains ou textes écrits par les machines et les associer à des classes spécifiques au cours de l'entrainement, ce au moyen d'une couche de neurones entièrement connectée et de la fonction d'activation *sigmoid*. Une fois entrainés, ces modèles peuvent projeter les classes connues et inconnues sur l'espace de représentation (*embedding space*). En mesurant la similarité entre les vecteurs caractéristiques *embedding* à l'aide de mesures de distance, le modèle peut déduire la catégorie des données inconnues.

Approche *fine-tuning*

Le *Fine-tuning* consiste à adapter un modèle pré-entrainé (GPT, BERT,...) pour exécuter une tâche spécifique (classification, génération de texte,...). Son but est d'optimiser les performances du modèle sur une nouvelle tâche (connexe à celle pour laquelle il était entrainé initialement) sans recommencer le processus d'entrainement depuis le début. Le processus du *fine-tuning* comprend plusieurs étapes clés, notamment la préparation des données, le choix du modèle pré-entrainé, l'ajustement de l'architecture du modèle (modification de la dernière couche de neurones, ajustement de la sortie) et l'initialisa-

tion des poids du modèle.
Les méthodes les plus avancées d'identification de textes générés par les machines à l'aide de réseaux neuronaux impliquent un *fine-tuning* de grands modèles linguistiques bidirectionnels [61] car ces méthodes offrent une meilleure performance [4]. Initialement, RoBERTa, une variante de BERT, a été ajusté pour évaluer les textes de GPT2 et pour distinguer les textes générés des textes humains [40]. Faire du *fine-tuning* sur BERT pour la classification de texte comprend l'ajout d'un jeton de classification `[CLS]` au début de chaque texte et dont la représentation vectorielle (*embedding*) sera le vecteur de caractéristiques de tout le texte. L'ensemble des vecteurs de caractéristiques des données d'entrainement est utilisé pour entrainer un réseau de neurones entièrement connecté avec une sortie binaire.

Resumé

En resumé, il faut noter que la plupart des travaux utilisent soit uniquement les caractéristiques globales des textes (issues des modèles auxiliaires, du comptage,...) soit les caractéristiques issues des modèles de langage (du texte brut) ou le mélange des deux types de caractéristiques. La figure 2.3 montre toutes les possibilités d'approches avec les étapes d'extraction des caractéristiques suivi de la classification.

2.4 Resultats des travaux récents

Dans cette section, nous présentons les résultats de quelques travaux récents que nous situons par rapport aux aux approches de détections de la figure 2.3. Nous présentons les resultats des travaux basés sur l'approche par les caractéristiques, ensuite des resultats des travaux basée sur l'approche neuronale et enfin des resultats des travaux basés sur les deux approches.

2.4.1 Travaux basés sur les modèles neuronaux

En 2019, Solaiman et al. [61] ont construit un détecteur de textes générés par GPT2 en faisant un *fine-tuning* de RoBERTa à l'aide des sorties du plus grand modèle GPT2 (comportant 1,5 milliard de paramètres). Avec une certaine exactitude, leur détecteur était capable de détecter si le texte était un texte généré par une machine ou non. Ils ont mené leurs travaux en utilisant RoBERTa-base avec 125 millions de paramètres et

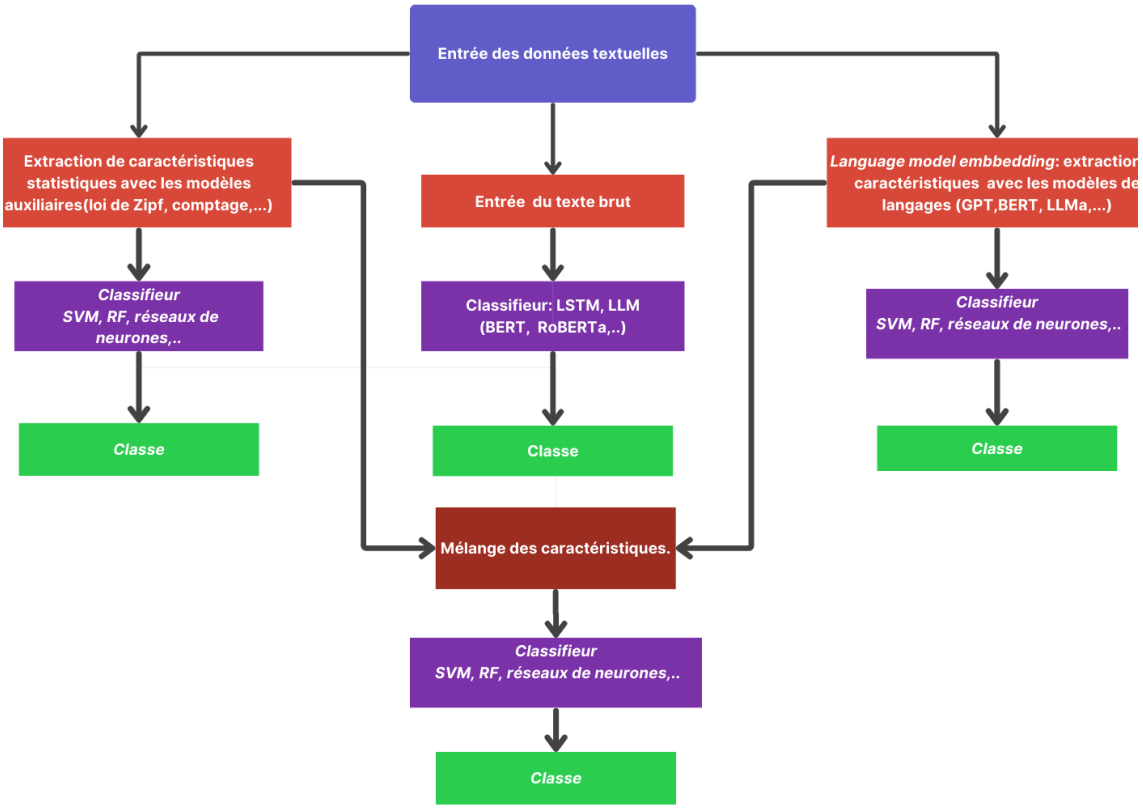

FIGURE 2.3 – Architecture d'un modèle de détection : extraction des caractéristiques issues des modèles de langage et des caractéristiques statistiques

RoBERTa-large avec 356 millions de paramètres comme base de leur classifieur. Ces auteurs ont abtenu un *accurary* de 95% sur le dataset des sorties de GPT-2 et les échantillons de textes de WebText, de longueur fixe de 510 *tokens*.
SemEval est une série d'ateliers internationaux de recherche sur le traitement du langage naturel (NLP) dont la mission est de faire progresser l'état de l'art en matière d'analyse sémantique. Avec les données du SemEval tâche 8, la sous-tâche A consistait à déterminer s'il s'agit d'un texte écrit par un être humain ou généré par une machine [68]. Les données d'entrainement contiennent 119757 textes en anglais issus des domaines tels que Wikipedia, WikiHow, Reddit, arXiv et PeerRead, dont $63,351$ écrits par les humains. L'équipe Areg Mikael Sarvazyan, José Ángel González, et Marc Franco-Salvador a obtenu la meilleure performance (avec une exactitude de 96.88%) dans cette sous-tâche en extrayant des caractéristiques probabilistes au niveau du jeton (probabilité logarithmique et entropie) en utilisant quatre modèles LLaMA-2 : LLaMA-2-7B, LLaMA-2-7Bchat, LLaMA-2-13B, et LLaMA-2-13B-chat, LLaMA étant grand modèle linguistique autore-

gréssif open source développé par Meta. Ces caractéristiques ont ensuite été introduites dans un encodeur transformateur entrainé de manière supervisée.

2.4.2 Travaux basés sur les caractéristiques syntaxiques, grammaticales ou autres caractéristiques lingustiques (notamment les POS)

Leon Fröhling et Arkaitz Zubiaga [22] ont proposé un classifieur basé sur plusieurs de ces caractéristiques pour distinguer les textes écrits par les humains des textes écrits par les machines. Ces auteurs ont entrainé leur classifieur avec un perceptron multicouche sur les données issues de GPT-2 et ont testé sa transférabilité sur les données issues de GPT-3 et Grover. Ils ont trouvé qu'en termes de performance individuelle, le sous-ensemble de caractéristiques le plus important est composé de la syntaxe, de la diversité linguistique et des caractéristiques de base. Ces auteurs ont obtenu une exactitude (*accuracy*) de 78,5% pour le modèle entrainé et testé sur les données de GPT-2 et 75,5% lorsqu'ils ont ajouté les données issues de GPT-3 et Grover. Sur un modèle d'ensemble, les meilleures performances de ces auteurs sont de 96,6% et de 92% pour les modèles entrainés et testés sur les données de GTP-2 small-k40, et GPT-2 xl-k40 respectivement.
Joel Thomas, Gia Bao Hoang et Lewis Mitchell ont entrainé un modèle d'ensemble dans lequel les étiquettes de POS ont été transformées en représentations vectorielles à l'aide de la fréquence des termes (TF) [65] et ces vecteurs leur ont servi de caractéristiques pour entrainer un modèle KNN(*K-nearest neighbors*).
Johan Lokna, Mislav Balunovic et Martin Vechev identifient les *patterns* de POS dont les occurences sont hautement prédictives en entrainant une regression logistique pour distinguer les textes humains des textes générés [41]. ces auteurs ont obtenu une amélioration de la précision à 86% (exactitude de leur modèle).

2.4.3 Travaux basés sur les deux approches

La tâche partagée AuTexTification (*Automated Text Identification*) qui a eu lieu dans le cadre de IberLEF 2023, le 5 atelier sur le forum d'évaluation des langues ibériques lors de la conférence SEPLN 2023. Sur le jeu de données AuTexTification 2023 qui contient 55677 textes en anglais, dont $27,989$ les textes écrits par les humains et $27,688$ générés par les LLM dans cinq domaines (tweets, critiques, articles pratiques,

actualités et documents juridiques), une équipe de chercheurs (Przybyla, Duran-Silva, et Egea-Gomez)[52] ont obtenu le meilleur score avec un Macro-F1 de $80,91\%$ [56]. Leur détecteur repose sur le modèle LSTM bidirectionnel entrainé sur une combinaison de caractéristiques probabilistes au niveau des jetons provenant de différentes versions de GPT-2, de caractéristiques linguistiques au niveau des jetons telles que la fréquence des mots ou les erreurs de grammaire, et de représentations de texte provenant d'encodeurs pré-entrainés.

2.4.4 Résumé

Dans cette section, nous avons présenté quelques résultats des travaux portant sur les différentes approches ainsi que l'utilisation des deux approches ensemble dans une même tâche. Ceci nous a permis d'obtenir la performance de chacune des approches, quoique cela soit sur les données différentes.
Nous pouvons constater qu'aucune des méthodes de détection des textes générés automatiquement n'utilise les convolutions. Étant donné que notre approche va exploiter cette architecture, il est important de faire une revue des réseaux de neurones convolutifs dans la section suivante.

2.5 Revue des réseaux neuronaux convolutifs

Les CNN sont des réseaux neuronaux artificiels multicouches capables de détecter des caractéristiques complexes dans les données. Il s'agit d'une architecture du *deep learning* initialement associée à des applications de traitement d'images, mais qui a également trouvé des applications dans le contexte du NLP et, en particulier, de la classification des textes [71].

2.5.1 Composition d'un CNN

En général, un CNN se compose de deux blocs principaux :

1. Premier bloc : extraction des caractéristiques en appliquant des opérations de filtrage par convolution. Dans ce bloc, la première couche filtre l'image avec plusieurs noyaux de convolution et renvoie des *feature maps*, qui sont ensuite normalisées (avec une fonction d'activation) et/ou redimensionnées grâce au *pooling*. Ce procédé peut être réitéré plusieurs fois : on filtre les features maps obtenues avec

de nouveaux noyaux, ce qui nous donne de nouvelles features maps à normaliser et redimensionner, et qu'on peut filtrer à nouveau, et ainsi de suite. Finalement, les valeurs des dernières feature maps sont concaténées dans un vecteur. Ce vecteur définit la sortie du premier bloc, et l'entrée du second.

2. Deuxième bloc : les valeurs du vecteur en entrée sont transformées (avec plusieurs combinaisons linéaires et fonctions d'activation) pour renvoyer un nouveau vecteur en sortie. Ce dernier vecteur contient autant d'éléments qu'il y a de classes.

Comme pour les autres réseaux de neurones, les paramètres des couches sont mis à jour par rétropropagation et par descente du gradient. Dans le cas des CNN, ces paramètres désignent en particulier les caractéristiques des images. Les deux blocs précédents se déploient sur différents types de couches d'un CNN.

2.5.2 Les différents types de couches d'un CNN

Il existe quatre types de couches pour un réseau de neurones convolutif :

- une couche convolutive pour obtenir des caractéristiques à partir des données
- une couche de mise en commun (*pooling layer*) pour réduire la taille de la carte de caractéristiques
- la couche de correction ReLU
- une couche entièrement connectée

La couche convolution

L'opération de convolution est responsable de la détection des caractéristiques les plus importantes. Elle nécessite quelques composants, à savoir des données d'entrée, un détecteur de caractéristiques, encore appelé noyau (*kernel*) ou filtre. Dans les tâches liées au traitement de l'image, le filtre se déplacera sur les champs réceptifs de l'image pour vérifier si la caractéristique est présente. Ce processus est appelé convolution. Plus précisément, le filtre est une matrice de poids bidimensionnelle (2D) de taille d $\times$ d (généralement d=3), qui représente une partie de l'image, détermine également la taille du champ réceptif. Le filtre est ensuite appliqué à une zone de l'image et un produit scalaire est calculé entre les pixels d'entrée et le filtre. Ce produit scalaire est ensuite intégré dans une matrice de sortie. Puis, le filtre se décale d'un cran pour répéter le processus jusqu'à ce que le filtre ait balayé toute l'image. La sortie finale de la série de produits scalaires issus de l'entrée et du filtre est connue sous le nom de carte de

caractéristiques, carte d'activation, *feature map* ou caractéristique convoluée (figure 2.4).
Les pondérations dans le filtre restent fixes lorsqu'il se déplace sur l'image et désignent

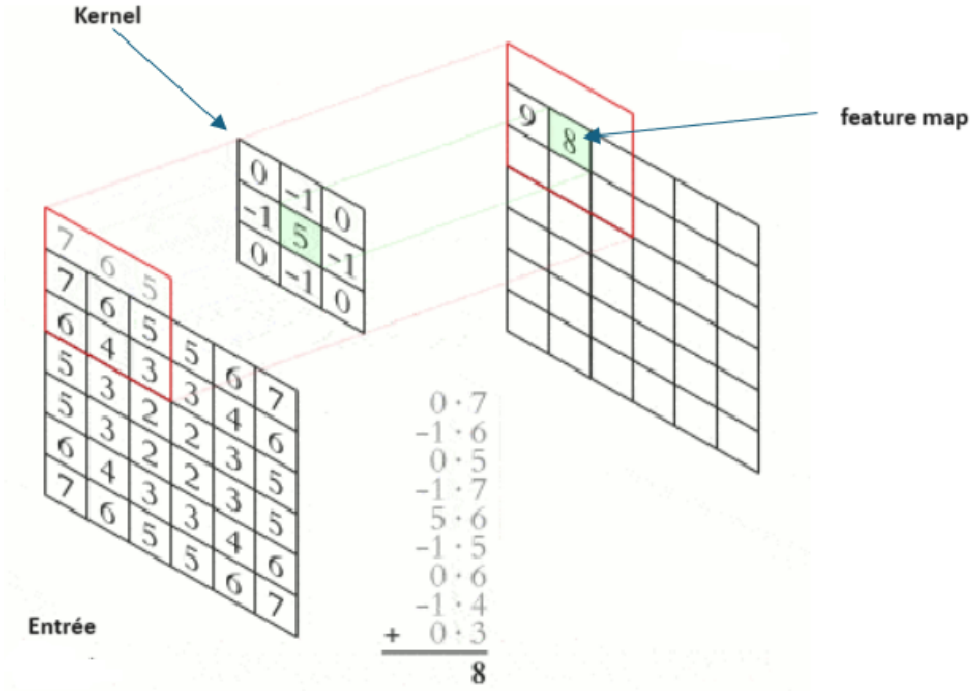

FIGURE 2.4 – Convolution entre les données d'entrée et le noyau : *feature map*[3].

les poids de la couche de convolution. Elles sont initialisées puis mises à jour pendant l'entraînement grâce au processus de rétropropagation et de descente de gradient.

Couche de non-linéarité : ReLu – Unité linéaire rectifiée

Étant donné que la convolution est une opération linéaire et que les images sont loin d'être linéaires, des couches de non-linéarité sont souvent placées directement après la couche de convolution pour introduire la non-linéarité dans la carte d'activation. Il existe plusieurs types de fonctions non-linéaires qui sont appliquées. Les plus populaires sont : la fonction sigmoïde, *tanh*, *ReLu*.
Après chaque opération de convolution, un CNN applique une transformation ReLU (unité de rectification linéaire) sur la carte d'activation, ce qui permet d'introduire une non-linéarité dans le modèle. Elle est définie par :

$$f(y) = max(0, y)$$

3. Tiré du site internet `https://cnvrg.io/cnn-sentence-classification/`

Elle remplace chaque valeur négative par un 0. Elle est très utilisée dans les réseaux de neurones à convolution car elle est rapide à calculer et sa performance est donc meilleure que d'autres fonctions où des opérations coûteuses doivent être effectuées.
La couche de convolution peut donc être résumée par la figure 2.5 où b est le biais d'entrainement.

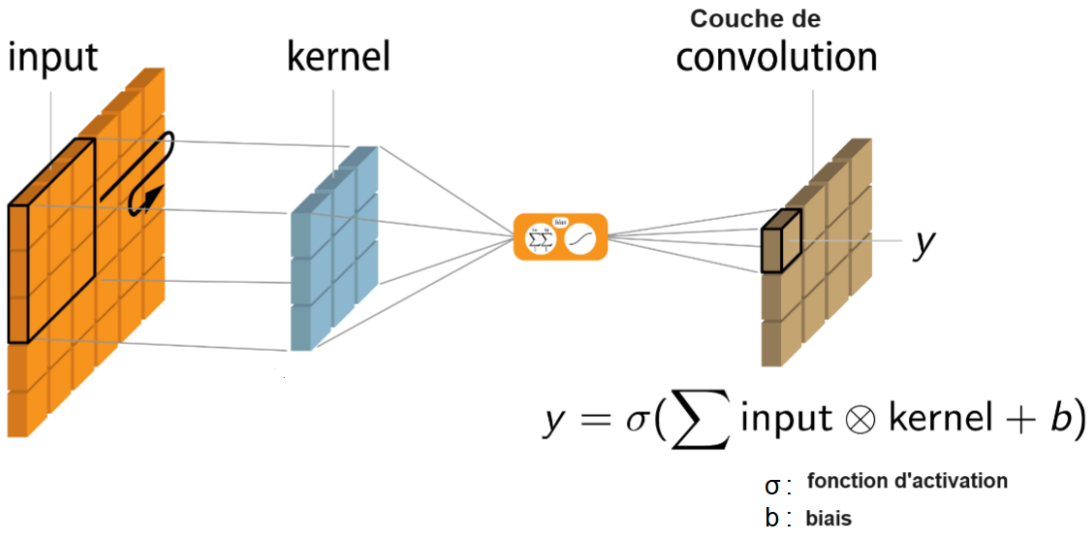

FIGURE 2.5 – couche de Convolution avec une couche en entrée ou de profondeur 1 [4]

Le nombre de couches de données d'entrée qui correspond à la taille de l'espace des caractéristiques, appelée nombre de *channels* (pour les images, le nombre de *channels* est 3 pour les couleurs RBG). Différents filtres peuvent être appliqués à chaque *channels*, puis fusionnés en un seul vecteur, figure 2.6.

La couche de *pooling*

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de mise en commun (*pooling*). L'opération de pooling consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Comme pour la couche de convolution, l'opération de pooling balaie un filtre sur toute l'entrée, à la différence que ce filtre n'a

4. Tiré du site internet `https://cloud.univ-grenoble-alpes.fr/index.php/s/wxCztjYBbQ6zwd6?dir=undefined&openfile=953832173`
5. Tiré du site internet `https://cloud.univ-grenoble-alpes.fr/index.php/s/wxCztjYBbQ6zwd6?dir=undefined&openfile=953832173`

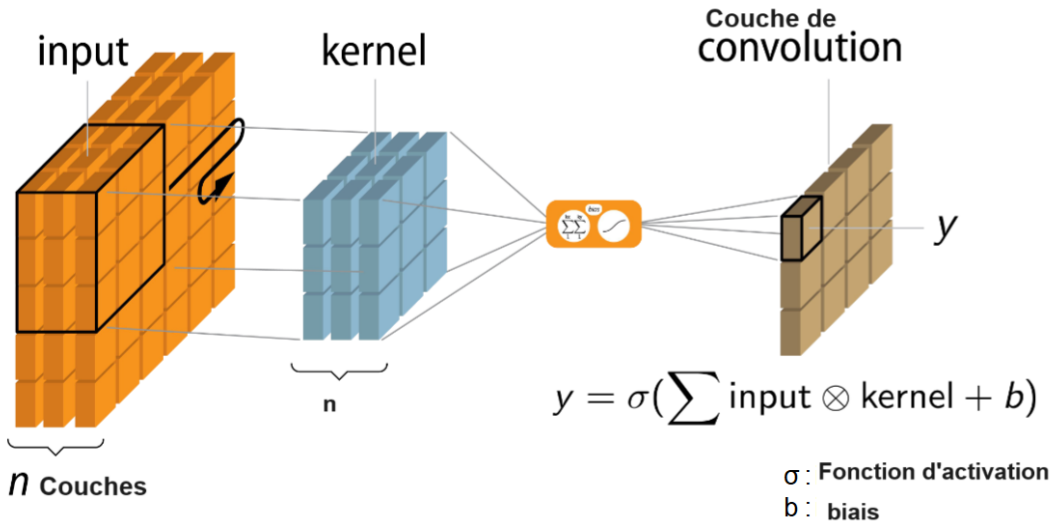

FIGURE 2.6 – Convolution avec entrée multicouche [5]

aucun poids. Au lieu de cela, le noyau applique une fonction d'agrégation aux valeurs du champ réceptif, remplissant ainsi la matrice de sortie. Il existe deux principaux types de pooling :

- Max pooling : à mesure que le filtre se déplace sur l'entrée, il sélectionne le pixel ayant la valeur maximale pour l'envoyer à la matrice de sortie.
- Average pooling : à mesure que le filtre se déplace sur l'entrée, il calcule la valeur moyenne dans le champ réceptif pour l'envoyer à la matrice de sortie.

Bien que de nombreuses informations soient perdues dans la couche de pooling, celle-ci présente malgré tout un certain nombre d'avantages pour le CNN. Elle permet de réduire la complexité, d'améliorer l'efficacité et de limiter le risque de sur-apprentissage.

Couche entièrement connectée(*fully-connected*)

Cette couche constitue toujours la dernière couche d'un réseau de neurones. À la fin du premier bloc d'un CNN, les valeurs des dernières *features maps* (redimensionnés grâce au *pooling*) sont concaténées (*flatten*) dans un vecteur qui définit l'entrée de cette couche. Elle effectue la classification en utilisant le vecteur précédent constitué des caractéristiques extraites à partir des couches précédentes et de leurs différents filtres, et renvoie un vecteur de taille N (N étant le nombre de classes). Pour cela, elle applique une combinaison linéaire des composantes de ce vecteur, puis éventuellement une fonction d'activation aux valeurs reçues en entrée. Les couches entièrement connectées

exploitent généralement une fonction d'activation *softmax* pour classer les entrées de manière appropriée, produisant une probabilité de 0 à 1.
En définitive, l'architecture d'un CNN peut être définie comme suit figure 2.7 :

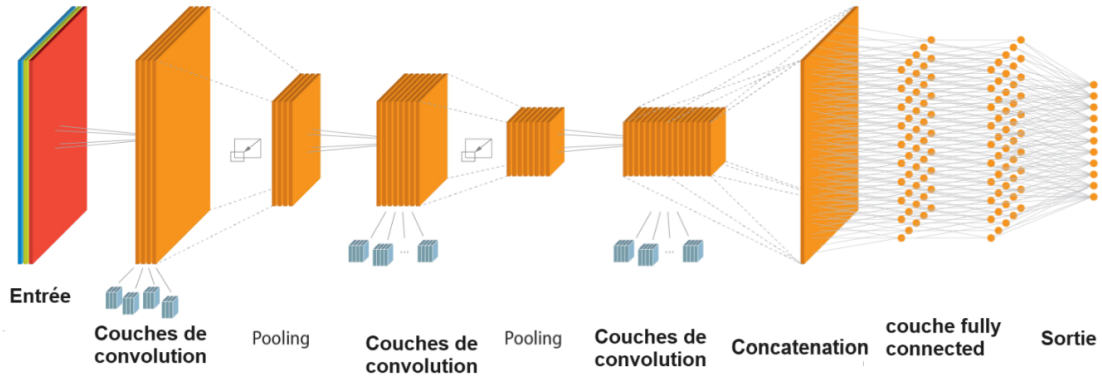

FIGURE 2.7 – Architecture d'un CNN

Le paramétrage des couches

Un réseau de neurones convolutif se distingue par la façon dont les couches sont empilées, mais également paramétrées. Les couches de convolution et de *pooling* possèdent en effet des hyperparamètres, c'est-à-dire des paramètres d'ensemble dont on doit préalablement définir la valeur. La couche de convolution possède quatre hyperparamètres :

1. le nombre de filtres
2. la taille des filtres
3. le pas avec lequel on fait glisser la fenêtre correspondant au filtre sur l'image (*Stride*)
4. la marge, ou zero-padding met à zéro tous les éléments qui se trouvent en dehors de la matrice d'entrée, produisant une sortie plus grande ou de taille égale.

2.5.3 CNN pour le NLP

Pour une tâche classique de NLP, chaque mot dans un texte est représenté dans l'espace d'intégration (*embbedding space*). Chaque ligne de cet espace est appelée *channel*. Lors des opérations de convolution, les filtres se déplacent dans la direction (d'où

5. Tiré du site internet `https://cloud.univ-grenoble-alpes.fr/index.php/s/wxCztjYBbQ6zwd6?dir=undefined&openfile=953832173`

convolution 1D) qui a un sens au niveau de la phrase, en tenant toujours compte de l'intégration complète pour chaque mot (déplacement le long des *channels*). La figure 2.8 illustre le déplacement d'un filtre de longueur 3 en longueur de la séquence : *tentative deal reached to keep government open*. Pour configurer un réseau de manière à ce qu'il

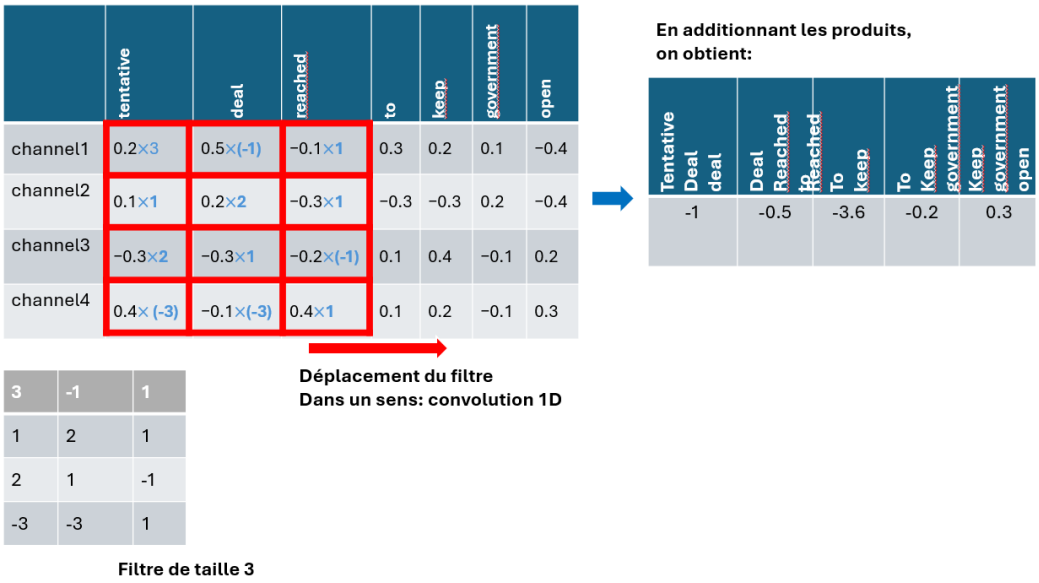

FIGURE 2.8 – Convolution 1D

soit capable d'apprendre une variété de relations différentes entre les mots, on a besoin de nombreux filtres de différentes longueurs. La méthode de pooling la plus courante est le *maxpooling*, qui consiste à sélectionner l'élément maximal dans la fenêtre de pooling. Cette méthode permet d'extraire les caractéristiques les plus significatives et de supprimer les caractéristiques relativement moins importantes.
Dans cette section, nous avons rappelé les connaissances nécessaires à la compréhension de notre approche qui sera présentée au chapitre 3.

2.6 Conclusion

Dans ce chapitre, nous avons présenté les notions liées à la génération automatique des textes avec quelques exemples de tâches de génération de textes et décrit les approches de génération de textes. Nous avons aussi rappelé l'état de l'art sur les approches de détection des textes générés par les machines, notamment les approches qui nécessitent une extraction préalable des caractéristiques (statistiques) et les approches qui utilisent

directement le texte brut sans extraction de caractéristiques. Nous avons aussi revu les réseaux neuronaux convolutifs ainsi leur application dans une tâche de NLP. Nous avons rappelé les connaissances sur les CNN, nécessaires à la compréhension de notre approche. Dans le chapitre suivant, nous allons présenter notre approche qui tient compte des caractéristiques linguistiques et de leur répartition séquentielle dans les textes pour faire la classification entre les textes générés et les textes écrits par les humains.

Chapitre 3

Méthodologie

Dans le chapitre précédent, nous avons brièvement exposé la génération des textes avec un accent sur les méthodes de génération avec l'architecture des *transformers* et l'état de l'art sur la détection des textes générés par les machines. Nous avons constaté que les méthodes de classification des textes utilisent des caractéristiques extraites des textes soit au moyen des modèles auxiliaires (modèles de Zipf, TF-IDF,...), du comptage des caractéristiques linguistiques des textes ou des modèles neuronaux (GPT, BERT, LLMa,...). Comme annoncé à l'introduction, nous allons extraire les caractéristiques issues *patterns* séquentiels des POS au moyen des réseaux de neurones convolutifs. Pour chaque texte, nous associons aux caractéristiques précédentes un autre vecteur de caractéristiques, qui est une représentation compacte (*pooler output*) basée sur le premier jeton [CLS] (le jeton spécial utilisé pour représenter l'ensemble de la séquence). Nous utilisons l'ensemble de ces caractéristiques pour déduire une classification.
Dans notre démarche, nous allons dans un premier temps produire un nouveau jeu de données avec les jeux de données SemEval 2024 tâche 8. Nous allons ensuite nous servir de ces données pour entrainer deux modèles : un classsifieur binaire de type *feed-forward* entrainé avec les caractéristiques *pooler output* et un autre classsifieur binaire de type *feed-forward* entrainé avec les caractéristiques *pooler output* couplées aux caractéristiques issues du comptage de chaque POS par texte. Les résultats obtenus nous servirons de base d'évaluation de notre approche et nous pourrons ainsi comparer nos résultats avec ceux obtenus de ces deux modèles. Nous présenterons enfin notre approche qui utilise les convolutions, les caractéristiques linguistiques et les caractéristiques issues du jeton CLS pour chaque texte pour distinguer des textes humains de ceux écrits par les machines.
Dans notre approche, nous nous intéressons aux caractéristiques linguistiques (classe

syntaxique), non pas en termes de fréquences comme dans la littérature, mais à la façon dont elles sont disposées (donc en termes de motifs) dans un texte. Notre méthodologie comprend l'extraction de balises fines des POS (*POS tags*) de chaque texte, leur encodage par les nombres, suivi de l'entrainement d'un modèle de convolution afin d'extraire les caractéristiques qui sont des motifs séquentiels, l'association de ces caractéristiques aux vecteurs CLS pour en déduire la classification.

3.1 Extraction des caractéristiques linguistiques

En anglais, il existe neuf catégories grossières de classes syntaxiques (*part-of-speech*,POS) dans une phrase (les noms, les pronoms, les adjectifs, les verbes, les adverbes, les prépositions, les conjonctions, les articles et les interjections) qui peuvent être étiquetés de manière fine en plusieurs sous-classes en incluant la ponctuation,les espaces d'autres symboles tels que : (/,*,@,... Afin de simplifier l'écriture, POS dans la suite inclura les classes syntaxiques, la ponctuation, les espaces et tout autre symbole significatif dans une phrase. La figure 3.1 montre les exemples d'étiquettes (colonne "Tag") utilisées dans le projet Penn Treebank [47]. Ces étiquettes sont principalement conçues pour être

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	*+,%, &*	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

FIGURE 3.1 – Étiquettes des POS utilisées dans le projet Penn Treebank [47] (y compris la ponctuation)

de bonnes caractéristiques pour les modèles d'analyse syntaxique. Nous attribuons un

nombre unique à chaque balise de POS . Nous utilisons la librairie Python *SpaCy* pour extraire les balises de POS de granularité fine dans chaque texte. Pour chaque texte, nous obtenons un vecteur de balises de POS. La figure 3.2 montre les POS pour le texte : *'tentative deal reached to keep government open'*. Nous convertissons les balises

```
TOKEN: tentative
          =====
          token.tag_ = 'JJ'
TOKEN: deal
          =====
          token.tag_ = 'NN'
TOKEN: reached
          =====
          token.tag_ = 'VBN'
TOKEN: to
          =====
          token.tag_ = 'TO'
TOKEN: keep
          =====
          token.tag_ = 'VB'
TOKEN: government
          =====
          token.tag_ = 'NN'
TOKEN: open
          =====
          token.tag_ = 'JJ'
```

FIGURE 3.2 – POS du texte : *'tentative deal reached to keep government open'*

en nombre et nous obtenons un vecteur de nombres. À titre d'exemple, le texte : *'tentative deal reached to keep government open'* est encodé par le vecteur suivant : [19, 25, 44, 39, 41, 25, 19] pour les POS. On peut constater que :

- la balise JJ des POS est représentée par le nombre 19. Les jetons *tentative* et *open* qui sont étiquetés comme des adjectifs (balise JJ) sont représentés par le même nombre 19.
- on remarque que la longueur des vecteurs correspond à la longueur des séquences des mots ou des *tokens*, contrairement aux approches classiques où la longueur est fixe (indépendante du texte), de plus l'ordre des composantes du vecteurs correspond à l'ordre des éléments dans le texte, ce qui permet de repérer les motifs séquentiels (perdus dans une approche classique).

L'ensemble des vecteurs peut être utilisé comme entrée d'un CNN.

3.2 Description du modèle

Dans notre approche, nous voulons distinguer les textes humains des textes générés par l'IA par la façon dont les mots sont agencés. Pour cela, nous nous intéressons aux POS. Nous donnons comme hypothèse que les textes générés par l'IA sont syntaxiquement semblables. La relation syntaxique entre les mots devrait être distribuée de manière identifiable pour les textes écrits par l'IA. Pour vérifier cela, nous avons mené nos premières expériences avec le modèle LSTM, mais les résultats n'étaient pas encourageants. Nous avons adopté pour notre approche les réseaux neuronaux convolutifs (CNN) à une dimension. Étant donné que les convolutions sont des opérations linéaires, lorsqu'un filtre convolutif est appliqué à des vecteurs similaires de distributions de POS il produira des valeurs de sortie similaires.
Nous avons choisi des séquences de textes de longueur maximale de 1500 mots. Si un texte a moins de 1500 jetons, nous complétons le vecteur numérique correspondant par les zéros. Après l'obtention des vecteurs numériques, nous appliquons plusieurs couches convolutives à ces vecteurs afin de détecter les caractéristiques (successions de n POS et/ou de n-grammes liés syntaxiquement), chaque filtre se spécialisant dans une famille de n-POS ou de n-grammes étroitement liée. Pour chaque donnée textuelle, nous remplaçons d'abord les *channels* du *word embbedding* (voir section 2.5.3) par le vecteur décrit par la section (3.1). Nous appliquons ensuite du *maxpooling* pour extraire les n-POS ou les n-grammes significatifs. La concaténation des caractéristiques (*flatten*) (suivi de l'ajout par concaténation des caractéristiques issues du jeton CLS de chaque texte pour le modèle CNN+CLS) et d'une de la technique de régularisation *dropout* (utilisé pour prévenir le sur-apprentissage) permet d'avoir un vecteur qui sera passé à un réseau de type *feed forward*, dont la couche de sortie effectue une classification binaire. L'architecture générale de notre modèle est inspirée de celle d'un CNN (cf 2.7) et représentée par la figure 3.3. Pour illustrer notre approche, considérons le vecteur de POS numérisé de la section 3.1.1 : [19, 25, 44, 39, 41, 25, 19]. La figure 3.4 illustre notre modèle pour les POS. Dans cet exemple, nous avons suivi les étapes de la figure 3.3 avec trois filtres de longueur 3.
Dans ce chapitre, nous avons présenté notre approche basée sur CNN à une dimension. Dans le chapitre suivant, nous allons l'expérimenter sur différents jeux de données et nous allons aussi comparer les résultats.

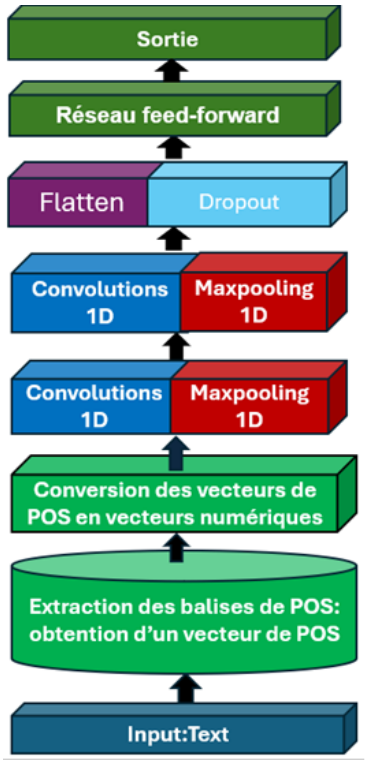

FIGURE 3.3 – Architecture de notre modèle

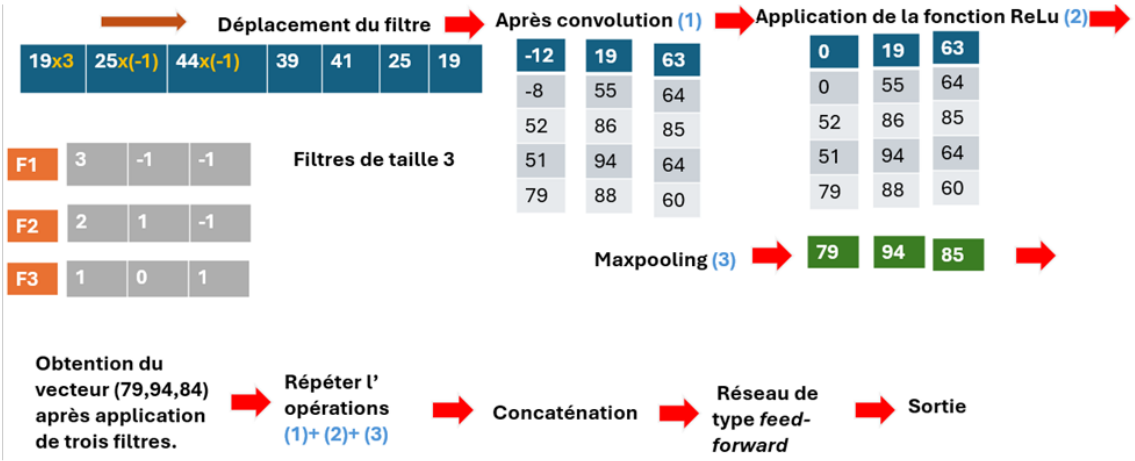

FIGURE 3.4 – Notre modèle appliqué à un vecteur de POS non complété par les zéros.

Chapitre 4

Validation

4.1 Méthodologie

Afin de valider notre approche de détection des textes générés par les machines, nous avons mené une série de tests sur des jeux de données composés de textes générés par les machines et de textes écrits par les humains. Plus précisément, nous avons évalué les performances de notre approche dans l'absolu, ensuite, nous verrons si les caractéristiques de notre approche peuvent être complémentaires à d'autres caractéristiques pour former un modèle plus performant et quelle est la pertinence de ces caractéristiques par rapport aux caractéristiques obtenues par comptage des POS et enfin, nous allons évaluer la robustesse de notre approche sur les attaques et sur la longueur des textes. Nous procéderons par des questions de validation, suivies des expériences et des conclusions.

4.1.1 Données utilisées

Nous avons choisi le jeu de données de SemEval 2024 tâche 8. Ces données ont l'avantage d'avoir les résultats de plusieurs méthodes de détection de textes générés par les machines. SemEval est une série d'ateliers de recherche internationaux sur le traitement du langage naturel (NLP). La tâche 8 se concentre sur la détection de textes générés par des machines en anglais. Ces textes peuvent être issus des générateurs différents, des domaines différents et/ou de plusieurs langues. La sous-tâche A consistait à déterminer s'il s'agissait d'un texte écrit par un être humain ou généré par une machine et la sous-tâche B consistait à attribuer les textes non seulement à leur nature générée par une

Split	Humain	Machine	Total
Ensemble d'entrainement	65387	68405	133792
Ensemble de validation	2500	2500	5000
Ensemble Test	18000	16272	34272

TABLEAU 4.1 – Taille des données

machine, mais aussi à des générateurs spécifiques, ce qui ressemble à l'attribution de la "paternité" d'un texte [68].

4.1.2 Description des données

Nous nous intéressons ici uniquement qu'à la sous-tâche A. L'ensemble des données d'entrainement initial provient des domaines tels que Wikipedia, WikiHow, Reddit, arXiv et PeerRead. Pour la classification des textes en anglais, l'ensemble des données comprenait un total de 119757 avec 56400 textes générés par des machines et 63351 textes écrits par des humains. Les textes générés proviennent des générateurs suivants : chatGPT, cohere, davinci, dolly. Les données de validation contiennent 2500 textes générés par des machines (du générateur BLOOMZ) et 2500 textes écrits par des humains. Les données de l'ensemble test sont issues du domaine OUTFOX (une plateforme de lecture et de co-écriture d'histoires) qui ne figure pas parmi les domaines des données d'entrainement, et sont composées de 18000 textes générés par des machines (dont 3000 de GPT4 qui ne figure pas parmi les générateurs des données d'entrainement) et de 16272 textes écrits par des humains.
Nos premières analyses des erreurs ont révélé que le sur-apprentissage est un défi majeur. Afin de résoudre ce problème, nous utilisons une stratégie adoptée par une équipe participant à la conférence SemEval 2024 et qui consistait à augmenter la taille de nos ensembles de données d'entraînement pour la sous-tâche A en ajoutant l'ensemble de données de la sous-tâche B et en supprimant les éléments dupliqués([46]). Nous avons ainsi obtenu un ensemble de 133792 textes, dont 68405 textes générés par des machines et 65387 textes écrits par des humains. Le tableau 4.1 donne le résumé des données que nous avons utilisées dans nos modèles. Nous avons considéré les textes ayant au maximum 1500 jetons et pour les textes de plus de 1500 mots, nous avons fait une troncature pour garder au plus 1500 jetons en conservant la structure du texte. Les données sont étiquetées 1 pour les textes générés par les machines et 0 pour les textes écrits par les

humains. La figure 4.1 nous donne un aperçu des données. *labels* désigne la classe du texte.

	texts	labels
0	If you're a photographer, keep all the necessa...	0
1	See the image for how this drawing develops st...	0
2	It is possible to become a VFX artist without ...	0
3	Some entire movies are improvised, some plays ...	0
4	Use your friends' conversations to figure out ...	0
...	...	...
103739	During the Cold War, the United States was por...	1
103740	The "continuity thesis" is the idea that there...	1
103741	In the early Middle Ages, the pagan Norse were...	1
103742	There are many similarities between the langua...	1
103743	News of Christopher Columbus' voyage to the Ne...	1

FIGURE 4.1 – Aperçu des données.

4.1.3 Métrique d'évaluation

Dans l'évaluation de nos modèles, sauf mention contraire, nous utilisons l'exactitude comme principale mesure d'évaluation. L'exactitude est le nombre de prédictions correctes, divisé par le nombre total d'échantillons :

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

Où :
TP : nombre de vrais positifs (*True positive*)
TN : nombre de vrais négatifs (*True negative*)

FP : nombre de faux négatifs (*false positive*)
FN : nombre de faux négatifs (*false negative*)
Nous avons également calculé le rappel, la précision et le score F1, mais ceux-ci se sont avérés peu informatifs dans ce cas précis, notamment parce que les ensembles de données sont équilibrés et que les modèles ne sont pas conçus pour optimiser un type d'erreur particulier. Nous notons en particulier que la précision est utilisée comme principale mesure pour évaluer les performances des attaques, étant donné que l'ensemble de données se compose uniquement de textes générés par des machines.

4.2 Résultats

4.2.1 Question 1 : Quelles sont les performances de notre technique dans l'absolu ?

Expérimentation 1 : Évaluation de notre technique appliquant les CNN aux vecteurs numériques de POS (modèle CNN+POS)

Le but de cette expérience est d'évaluer les performances de notre approche seule. Pour ce faire, nous avons d'abord transformé chaque donnée textuelle de notre jeu de données en vecteur numérique tel qu'indiqué dans la section 3.1 du chapitre précédent. Nous avons obtenu un nouveau jeu de données (pour les POS) constitué chacun de vecteurs de nombres avec les classes correspondantes, ce, pour les données d'entraînement, de validation et de test.
La bibliothèque *keras* nous a fourni les fonctions *conv1D*, *MaxPooling1D*, *flatten* et *dropout* et *Dense* que nous utilisons lors de l'entrainement de chaque modèle. Nous avons utilisé une configuration A100 GPU de Google colab.
Pour notre modèle basé sur les POS et utilisant les CNN, tout en suivant l'architecture décrite dans la section 3.2 (3.3), nous avons configuré la taille de lot de 256. Nous avons appliqué deux fois 256 des convolutions exécutées en parallèle avec les filtres de taille 5, suivies de 256 et de 256 convolutions en parallèle toujours avec les filtres de taille 5. Nous avons fixé le *dropout* à 0.5. Les caractéristiques issues de ces convolutions sont concaténées et ensuite passées à travers une couche entièrement connectée avec 256 sorties, suivie d'une autre couche entièrement connectée avec seule une sortie. La figure C nous présente un résumé de notre modèle. Ce modèle est entraîné sur un total de 25 époques.

Avec les données décrites à la section 4.1.2, notre modèle nous a permis d'obtenir une moyenne d'exactitude de 87,19% avec un écart-type de 2,0%, sur l'ensemble de test ce, après 20 entrainements. Notre meilleur modèle des 20 entrainements nous a donné une exactitude de 91,1%, ce qui est une performance assez compétitive au regard du classement des performances de la conférence SemEval tâche 8, sous-tâche A [68]. Cependant, cette performance n'est pas la meilleure au regard des performances de la conférence SemEval tâche 8, sous-tâche A (dont la meilleure performance est de 96,88% d'exactitude). Peut-on exploiter les *features* issues de notre approche pour entraîner un modèle plus performant ?

4.2.2 Question 2 : Peut-on combiner les caractéristiques issues de notre approche aux caractéristiques neuronales afin d'obtenir un modèle plus performant ?

Pour répondre à cette question, nous allons faire l'expérience suivante :

Expérimentation 2 : modèle par concaténation des représentations neuronales et des caractéristiques *partterns* de POS (modèle CNN+CLS)

.

Pour le modèle CNN+CLS, nous avons appliqué les mêmes couches de convolutions que dans le modèle (CNN+POS) afin d'extraire les motifs caractéristiques. Tout comme dans le premier modèle (CNN+POS), ces caractéristiques sont concaténées et pour chaque texte, nous avons associé à chacune de ces caractéristiques extraites, la caractéristique neuronale du même texte. Et l'ensemble des caractéristiques est transmis à un réseau de type *feed forward*, dont la couche de sortie effectue une classification binaire. La figure 4.2 nous donne l'architecture de ce modèle et la figure D est son résumé.

Ce modèle prend en entrée, deux entrées (*inputs*) séparés (les vecteurs numériques de POS des textes d'une part et les vecteurs CLS associés à ces mêmes textes d'autre part) et l'ensemble est entrainé une seule fois dans un réseau *feed-forward*. Avec le même jeu de données, le modèle CNN+CLS nous a permis d'obtenir une moyenne d'exactitude de 92,87% avec un écart-type de 1,03% après 20 entrainements. Notre meilleur modèle nous a donné une exactitude de 95,4%.
On constate une augmentation des performances du modèle résultant de la combinaison

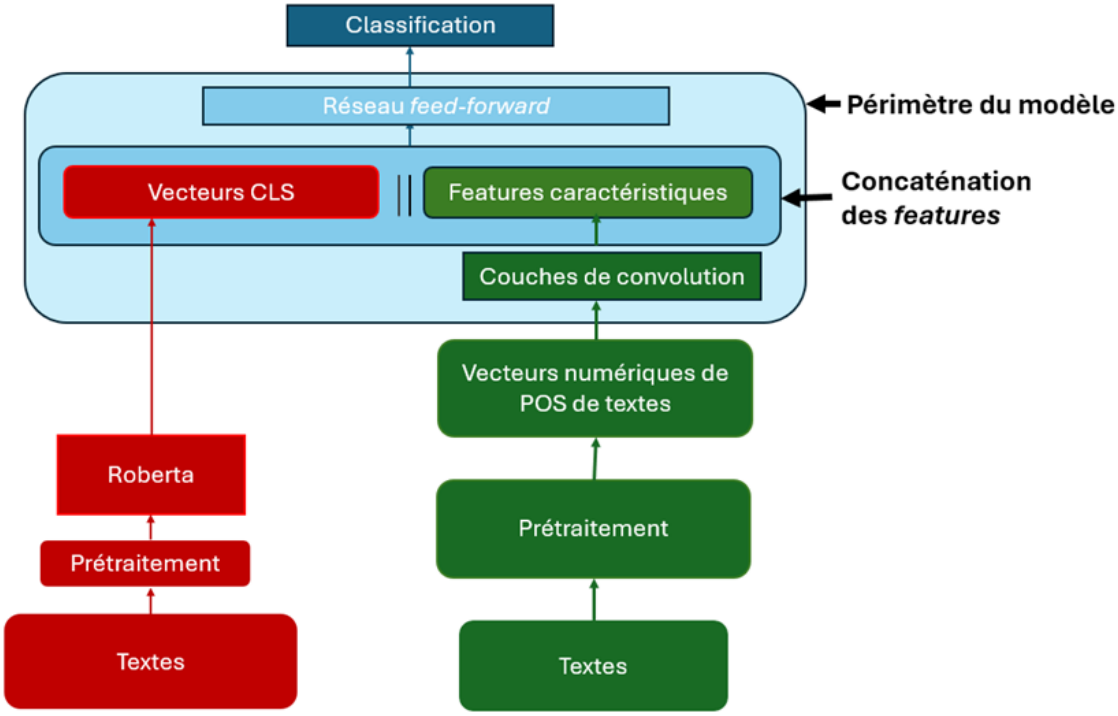

FIGURE 4.2 – Architecture du modèle CNN+CLS

des caractéristiques issues de notre approche et les caractéristiques neuronales. De plus, ce modèle hybride est plus stable au regard des écarts-types. Le tableau 4.3 confirme que ce modèle (CNN+CLS) est meilleur que notre modèle (CNN+POS). Ces résultats démontrent que notre approche peut être complémentaire aux approches neuronales afin d'obtenir un modèle plus performant.

4.2.3 Question 3 : Quelle serait la place des *features* de notre approche par rapport au comptage des POS ?

Pour répondre à cette question, nous allons comparer résultats des performances du modèle obtenu par ajout des caractéristiques par comptage des POS aux caractéristiques neuronales (modèle CLS+comptage) et du modèle par concaténation des représentations neuronales et des caractéristiques *partterns* de POS (modèle CNN+CLS). Nous allons partir d'un modèle avec les représentations neuronales (modèle CLS *full connected*) pour comparer l'effet de la combinaison des différentes caractéristiques.

Expérimentation 3 : modèle avec la représentation neuronale des textes

Nous utilisons *RoBERTa-base* comme source de représentation neuronale des textes. Nous avons trois autres jeux de données obtenus à partir des représentations neuronales de chaque texte, des données d'entraînement, de validation et de test. Nous entrainons un réseau de neurones *feed-forward* avec deux couches cachées accompagnées de la fonction d'activation ReLU, avec le taux d'apprentissage 5.10^{-5}, la norme L2 des poids $0,01$ et un arrêt précoce après 25 époques. Chaque couche cachée a une normalisation par lots et un *dropout* de $0,5$. Ce réseau de neurones *feed-forward* entrainé sur les vecteurs ici de la représentation neuronale nous donne une exactitude moyenne de 84,65% avec un écart-type de 2,87% sur l'ensemble test.

Expérimentation 4 : modèle par combinaison des caractéristiques par comptage des POS et de la représentation neuronale de chaque texte (modèle CLS+comptage)

Nous avons compté les POS pour chaque texte. Nous avons ainsi obtenu un vecteur de longueur 57 pour chaque texte. Nous avons ensuite associé à ce vecteur, le vecteur issu de la représentation neuronale du texte correspondant qui est de longueur 768. Nous obtenons, pour chaque, un vecteur de longueur 825. Nous entrainons un réseau de neurones *feed-forward* ayant les mêmes paramètres que celui de l'expérimentation 3. Nous avons une exactitude moyenne de 87,44% avec un écart-type de 5,59%. La figure 4.3 représente l,architecture du modèle CLS+comptage. Le tableau 4.2 nous présente les moyennes et les écart-types des exactitudes que chaque modèle. Nous pouvons constater que notre modèle est plus stable que les modèles *CLS+full connected* et *CLS + comptage*. Nous pouvons également constater que le modèle CNN+CLS est plus performant et plus stable que le modèle CLS+comptage. Le test de Student (tableau 4.3) et le tableau 4.2 nous montrent que le modèle CNN+CLS est meilleur que notre modèle et le modèle CLS+comptage. Étant donné les performances fluctuantes de nos modèles, nous pouvons confirmer la comparaison de ces modèles au moyen de la comparaison des distributions des exactitudes avec des boîtes à moustaches. Nous avons fait 20 entrainements pour notre modèle (modèle CNN+POS), pour le modèle par combinaison des représentations neuronales et des caractéristiques par comptage des POS (modèle CLS+comptage), le modèle par combinaison des représentations neuronales et des caractéristiques *patterns*

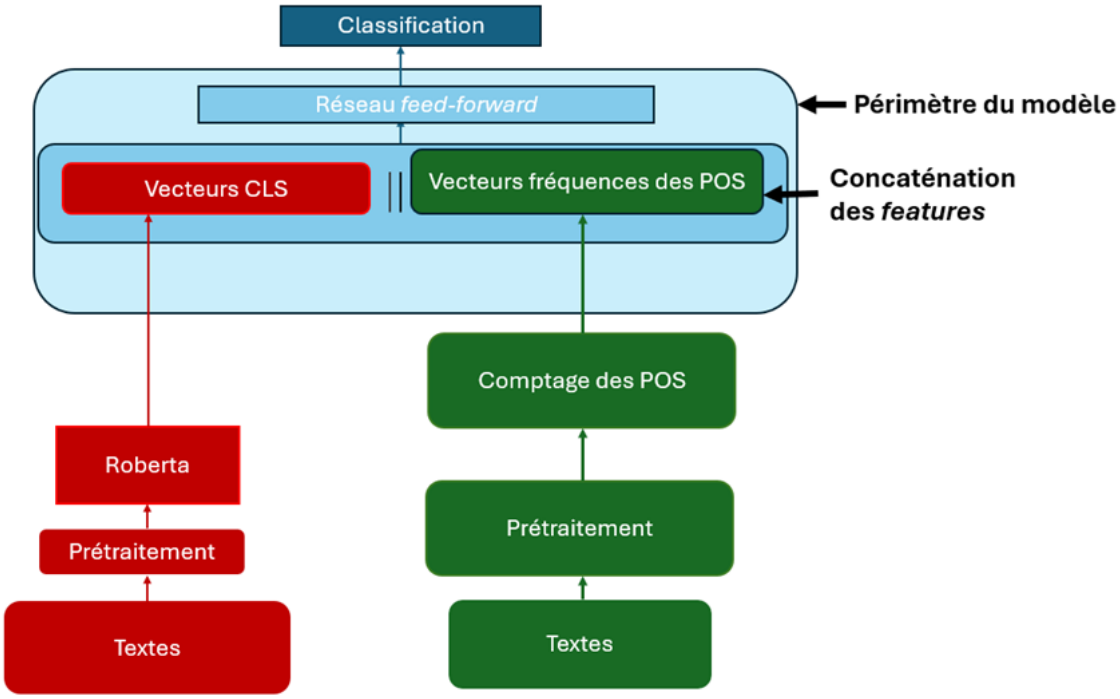

FIGURE 4.3 – Architecture du modèle CLS+comptage

Modèles	Moyenne des exactitudes	Écart-type des exactitudes
Notre modèle	87,19%	2%
CLS+full connected	84,65%	2,87%
CLS+comptage	87,44%	5,59%
CNN+CLS	92,87%	1,03%

TABLEAU 4.2 – Moyennes et écart-type des exactitudes des modèles

de POS (modèle CNN+CLS). La figure 4.4 nous montre les boites à moustaches des distributions des exactitudes des différents modèles.

La combinaison des caractéristiques issues de notre approche aux caractéristiques neuronales permet d'obtenir de meilleures performances par rapport à l'ajout des caractéristiques par comptage des POS. Ceci démontre que l'information "grammaticale" (l'ordre des POS) est complémentaire à celle des caractéristiques (*features*) neuronales (vecteurs CLS) et que les vecteurs caractéristiques obtenus par comptage des POS ne capturent pas cette information de manière adéquate.

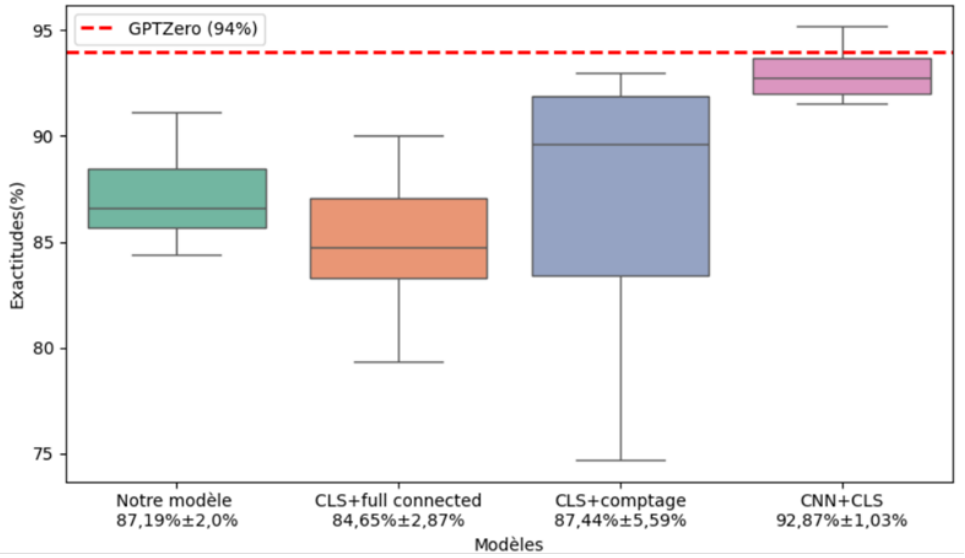

FIGURE 4.4 – Comparaison des distributions des performances des modèles

Comparaison de modèles	**Statistique**	**p-valeur**	**Conclusion**
Notre modèle vs Modèle CNN +CLS	-10.0547	$< 10^{-4}$	Modèle CNN +CLS est meilleur que notre modèle
Modèle CNN +CLS vs modèle CLS+comptage	-12.5201	$< 10^{-4}$	Modèle CNN +CLS est meilleur que le modèle CLS+comptage

TABLEAU 4.3 – Test de Student pour la comparaison des modèles

4.2.4 Question 4 : Quelle est la robustesse de notre approche ?

Nous allons évaluer la robustesse de notre modèle en testant l'influence des attaques et l'influence de la longueur des textes sur les performances. Nous ferons une comparaison avec les performances de RoBERTa et de GPTzero. Nous allons faire un fine-tuning de RoBERTa à l'expérience 5 et utiliser le meilleur modèle pour faire de la comparaison.

Expérimentation 5 : entraînement avec RoBERTa

Nous avons entraîné le texte brut avec RoBERTa afin de distinguer les textes humains des textes générés. Avec comme hyperparamètres, un taux d'apprentissage 2.10^{-5}, un poids de $0,01$ et après 5 époques, nous avons obtenu une exactitude de 89%

Expérimentation 6 : test de l'influence des attaques de notre modèle

Nous avons testé notre modèle sur des textes ayant subi 10 attaques adversariales (orthographe alternative, suppression d'article, ajout de paragraphes, majuscule-minuscule, espace de longueur nulle, ajout des espaces entre les caractères, homoglyphe, mélange aléatoire de chiffres, Insertion de fautes d'orthographe courantes, synonyme) qui des modifications sensées tromper le classifieur,et nous avons fait un test avec un détecteur commercial GPTzero et RoBERTa. Ces attaques sont issues du benchmark RAID [17] ainsi que les codes pour réaliser ces attaques. Nous les avons initialement tester pour des échantillons de texte pour confirmer les attaques. La figure 4.5 nous fournit les résultats sur un échantillon de 1000 textes générés par une machine. La figure 4.5 nous montre que

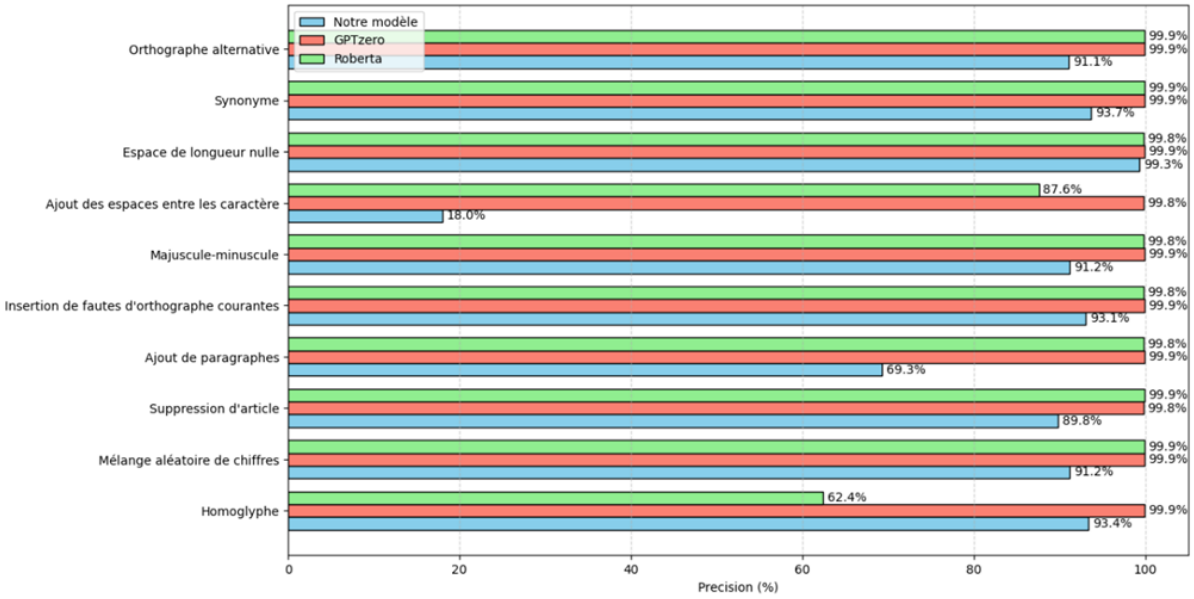

FIGURE 4.5 – Précision sur chaque échantillon de 1000 textes issu de l'attaque de 1000 textes générés par une machine. La précision sur l'échantillon initial de 1000 textes générés est de 91,10% avec une précision moyenne des 10 attaques de 83,01% .

certaines attaques diminuent le nombre de faux positifs (homoglyphe, insertion de fautes d'orthographe courantes, espace de longueur nulle, synonyme). Cela est aussi observable pour GPTzero. On observe, notre modèle est sensible à l'ajout des espaces entre les caractères (la précision chute de 91,10% à 18%).Cela s'explique facilement par le fait que les espaces affectent la tokenisation et l'analyse syntaxique du texte, ce qui affecte à son tour l'identification des balises POS, qui sont à la base de notre approche. Cependant, le modèle est moins performant sur les attaques que GPTzero et RoBERTa.

Il faut noter que les attaques ne semblent pas trop fonctionner pour notre modèle (dans certains cas, les performances ont augmenté sur les textes modifiés). En dehors de l'ajout des espaces entre les caractères (dont la précision est de 18%) et de l'insertion des paragraphes (dont la précision est de 69,3%), la précision moyenne sur les 8 autres attaques est 92,85% avec un écart-type de 2,93%. Elles ne semblent non plus fonctionner sur GPTzero et RoBERTa qui sont les deux systèmes avec lesquels nous faisons la comparaison. Ceci va à l'encontre des conclusions de l'article sur RAID[17] et du *leaderboard* accessible en ligne, où on voit des performances assez médiocres de plusieurs détecteurs sur les textes ayant subi les attaques. Nous n'avons pas utilisé les mêmes données, car nous voulions évaluer les effets des attaques sur les textes que nous avions déjà, mais nous avons utilisé leur code pour transformer nos textes, ce qui nous donne confiance dans nos résultats. Des travaux additionnels d'expérimentation et d'analyse seraient nécessaires pour mieux comprendre ceci.

Expérimentation 7 : influence de la longueur des textes

Dans cette expérience, nous avons testé l'influence de la longueur des textes sur les performances de notre modèle et fait une comparaison avec GPTzero et RoBERTa.
À partir de l'échantillon initial (les 1000 textes générés) et des 1000 textes humains, soit au total 2000 textes, nous avons créé cinq jeux de données. Pour le premier, chaque texte représente 20% des phrases du texte initial, le deuxième, 40%, le troisième, 60% et le quatrième, 80% et le dernier, 100%. La figure 4.6 nous résume l'évolution de l'exactitude en fonction des pourcentages de phrases dans chaque texte. L'expérience 7 nous démontre que notre modèle est sensible à la longueur des textes : plus le texte est court, moins il est performant (voir figure 4.6). Sur les textes courts, il est globalement moins performant que GPTzero et RoBERTa (car les séquences caractéristiques sont moins fréquents sur les textes plus courts). Cependant, il est plus performant sur les textes plus longs : 91,65% contre 88,55% pour GPTzero et 88,35% pour RoBERTa.
Les expériences nous ont permis de conclure que peut tenir compte des artéfacts grammaticaux (les POS) pour faire de la classification des textes humains et des textes générés. Elles nous démontrent qu'il est pertinent de prendre en compte de la disposition dans le texte, au sens de l'ordre ou des patterns séquentiels des POS pour distinguer les textes humains des textes générés. La combinaison des caractéristiques issues des *patterns* séquentiels des POS avec d'autres caractéristiques permet d'avoir des meilleures performances. Nous avons également évalué l'impact de la prise en compte des caracté-

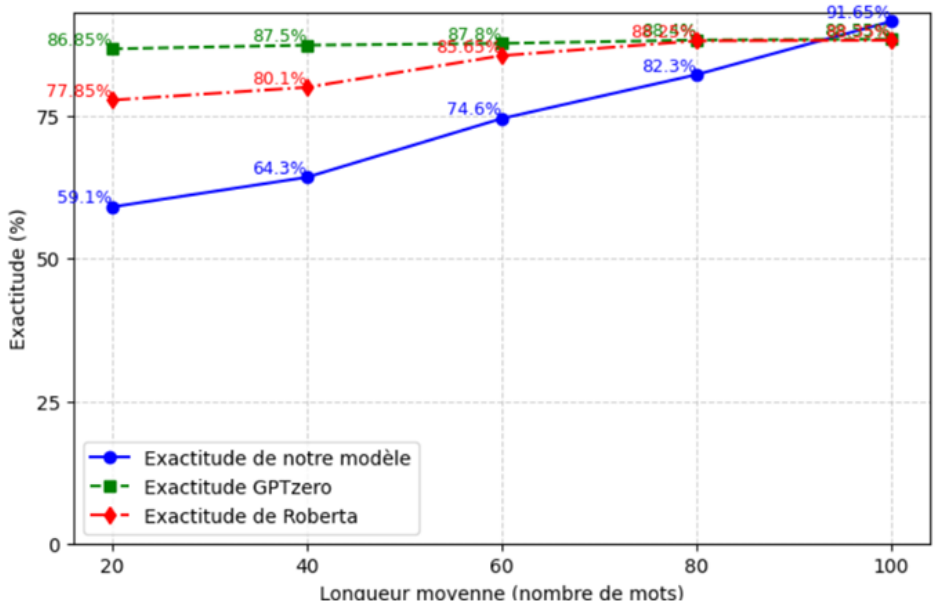

FIGURE 4.6 – Évolution des exactitudes en fonction du pourcentage des phrases dans le texte

ristiques issues de la fréquence des POS contre les caractéristiques issues de la prise en compte des *patterns* séquentiels. Nous avons enfin évalué la robustesse de notre modèle.

Chapitre 5

Conclusion

De récentes avancées dans le domaine du NLP montrent que les LLM peuvent générer des textes semblables à ceux des humains pour un certain nombre de tâches. Cette génération de textes à l'aide des LLM est souvent sujette à une utilisation abusive, notamment pour le plagiat, du *spamming*, de la désinformation, l'hameçonnage...etc. Il devient donc important de développer des détecteurs de textes LLM efficaces pour réduire l'exploitation abusive de la génération des textes par les LLM. Des travaux récents proposent une variété de détecteurs de texte IA utilisant les caractéristiques linguistiques, des caractéristiques neuronales ou une combinaison des deux types de caractéristiques.
Dans le cadre de ce mémoire, notre question de recherche principale est de savoir si on peut exploiter les caractéristiques grammaticales pour distinguer les textes humains des textes générés par les machines. Nous avons fait le tour des résultats pertinents à date sur ce sujet et avons constaté que les caractéristiques grammaticales seules n'ont pas trop été utilisées et quand elles le sont, ce sont des comptages et des caractéristiques parmi d'autres (TF-IDF, coefficient de Zipf, la fluidité, ...). Peut-on faire mieux ? Nous avons formulé l'hypothèse suivante :

*Il serait bénéfique de prendre en compte la disposition (au sens de l'ordre) des tokens qui sont des POS (*part-of-speech*) au sens grammatical (classes syntaxiques des mots, espaces, ponctuations, tout autre symbole) dans un texte, pour savoir s'il est généré par une IA ou écrit par un humain.*

Pour valider notre hypothèse, nous avons conçu une représentation des textes en tant que séquences d'étiquettes grammaticales (*Part-of-Speech tag*), par des vecteurs de nombres

(chaque étiquette de POS étant codé par un nombre unique), et développé un modèle qui utilise les CNN (que nous avons nommé modèle CNN+POS) pour identifier des *patterns* dans ces séquences d'étiquettes grammaticales à travers les vecteurs représentants les séquences de POS afin de s'en servir pour faire de la classification. Pour évaluer notre système, nous avons procédé de la manière suivante :

1. l'expérimentation 1 qui a consisté à tester les performances de notre modèle dans l'absolu. Nous avons obtenu une moyenne (sur 20 entrainements) d'exactitude 87,19% sur l'ensemble de test avec un écart-type de 2,0% et notre meilleur modèle était 91,1%. Même si ces résultats ne sont les meilleurs, ils restent assez compétitifs au regard des résultats obtenus lors de la conférence SemEval. Peut-on exploiter les *features* issus de notre approche ?
2. l'expérimentation 2 avait pour but d'évaluer la complémentarité des *features* issus de notre modèle. Nous avons combiné ces *patterns* séquentiels caractéristiques (issues des couches de convolution dans notre approche) aux vecteurs de représentation neuronale pour chaque texte et avons utilisé un réseau de neurones entièrement connecté pour former un nouveau modèle de classification (modèle CNN+CLS). Nous avons obtenu une moyenne (sur 20 entrainements) d'exactitude de 92,87% sur l'ensemble de test avec un écart-type de 1,03%. Notre meilleur modèle dans cette expérience nous a donné une exactitude de 95,4%. Ceci démontre que l'ajout des caractéristiques issues de notre approche aux caractéristiques neuronales permet d'avoir un modèle plus stable et plus performant.
3. l'expérimentation 4 avait pour objectif de comparer les caractéristiques issues de notre modèle et les caractéristiques par comptage des POS. Comme dans l'architecture du modèle CNN+CLS, elle consistait à combiner les *patterns* séquentiels caractéristiques aux vecteurs caractéristiques obtenus par comptage des POS dans chaque texte et avec un réseau de neurones entièrement connecté, d'entrainer un autre modèle de classification (modèle CLS+comptage). Nous avons obtenu une moyenne (sur 20 entrainements) d'exactitude de 87,44% et un écart-type de 5,59%. La figure 4.4 et le tableau 4.2 nous permettent de confirmer que la combinaison de notre approche (considérer les *patterns* séquentiels) avec les représentations neuronales de chaque texte permet d'obtenir de meilleures performances et une meilleure stabilité qu'avec la combinaison des caractéristiques par comptage et des représentations neuronales. Ceci nous démontre que les carac-

téristiques issues de notre approche sont plus pertinentes que les caractéristiques par comptage des POS.

4. les expérimentations 6 et 7 nous ont permis de tester respectivement la robustesse de notre approche par rapport aux attaques et la robustesse par rapport à la longueur des textes. Sur un échantillon de 2000 textes, on constate que notre modèle est plus performant que Roberta et GPTzero lorsqu'on considère 100% des phrases de chaque texte (textes plus longs) et moins performant pour les textes plus courts (voir figure 4.5 et figure 4.6). Tout comme GPTzero et Roberta, notre modèle diminue le nombre de faux positifs lorsqu'il est soumis aux attaques(ceci va à l'encontre des conclusions de l'article sur RAID et du leaderboard accessible en ligne, où on voit des performances assez médiocres de plusieurs détecteurs). Il est cependant moins performant que GPTzero et Roberta face aux attaques. Il faut noter que

Les résultats des expériences 1,2,4 nous permettent de valider l'hypothèse de recherche puisque notre modèle donne des performances compétitives dans l'absolu (proche de 90%) et la combinaison des caractéristiques issues de notre approche avec caractéristiques neuronales permet d'avoir de meilleures performances, mieux que la combinaison avec les caractéristiques issues du comptage des POS.

5.1 Nos contributions

On a découvert qu'on peut distinguer des textes humains et générés avec une exactitude proche de 90% avec les seules annotations grammaticales (la séquence de POS), c'est-à-dire sans savoir quels mots spécifiques ont été utilisés. De plus, même si d'autres détecteurs ont des performances plus élevées que celles de notre modèle, on a démontré que les caractéristiques obtenues par notre approche sont complémentaires à d'autres caractéristiques comme le vecteur CLS, au sens où elles peuvent être intégrées à des détecteurs combinant plusieurs types de caractéristiques et améliorer leurs performances. Un article de recherche présentant ces travaux est en préparation pour soumission à la conférence ECAI 2025.

5.2 Recommandations et limitations

Notre approche produit des résultats compétitifs pour les textes plus longs, mais moins efficace pour les textes plus courts (91,65% pour les textes plus longs contre 59,1% pour les textes courts d'environ 100 jetons, voir expérimentation 7). Il faut noter aussi que les textes utilisés lors de l'entrainement de notre modèle étaient d'au plus 1500 *tokens*.

Notre approche peut être utilisée comme complément à d'autres méthodes. Cette méthode pourrait avoir un avantage significatif en ce sens qu'on pourrait identifier les séquences de POS caractéristiques des textes générés par l'IA. L'analyse grammaticale est spécifique à une langue et nous ne savons pas ce qui se passerait avec notre approche dans d'autres langues.

5.3 Travaux futurs

Nous pourrons l'explorer dans le futur et aussi explorer l'aspect interprétabilité de notre méthode. On pourrait aussi exploiter les arbres de parsage qui prennent en compte les fonctions grammaticales des *tokens* dans un texte pour entrainer un autre modèle de classification qui fournirait aussi des prédictions interprétables.

Annexe A

Performances des modèles en fonction de la longueur des textes

	Performances du modèle		**Performances de GPTzero**		**Performances de Roberta**	
Données dont les textes sont de longueur :	Exactitude	Précision	Exactitude	Précision	Exactitude	Précision
20% des phrases du texte initial	59,10 %	60,3%	86,85%	79,6%	77,85%	98,9%
40% des phrases du texte initial	64,30%	57,4%	87,5%	80,25%	80,1%	99,4%
60% des phrases du texte initial	74,60%	63,6%	87,8%	80,64%	85,65%	99,8%
80% des phrases du texte initial	82,30%	72,9%	88,40%	81,40%	88,25%	99,8%
100% des phrases du texte initial	91,65%	91,1 %	88,55%	81,72%	88,35%	99,9%

TABLEAU A.1 – Performances en fonction de la longueur des textes

Annexe B

Performances des modèles en fonction des attaques

	Performances du modèle		**Performances de GPTzero**		**Performances de Roberta**	
Attaques	Exactitude	Précision	Exactitude	Précision	Exactitude	Précision
Homoglyphe	93,4%	93,4%	99,9%	99,9%	62,4%	62,4%
Mélange aléatoire de chiffres	91,2%	91,2%	99,9%	99,9%	99,9%	99,9%
Suppression d'articles	89,8%	89,8%	99,8%	99,8%	99,9%	99,9%
Ajout de paragraphes	69,3%	69,3%	99,8%	99,8%	99,8%	99,8%
Insertion de fautes d'orthographe courantes	93,1%	93,1%	99,9%	99,9%	99,8%	99,8%
Majuscule-minuscule	91,2%	91,2%	99,9%	99,9%	99,8%	99,8%
Ajout des espaces entre les caractères	18,0%	18,0%	99,9%	99,9%	87,6%	87,6%
Espace de longueur nulle	99,3%	99,3%	99,8%	99,8%	99,8%	99,8%
Synonyme	93,7%	93,7%	99,9%	99,9%	99,9%	99,9%
Orthographe alternative	91,1%	91,1%	99,9%	99,9%	99,9%	99,9%
Texte initial	91,1 %	91,1 %	99,9%	99,9%	99,9%	99,9%

TABLEAU B.1 – Performances sur chaque échantillon de 1000 textes issu de l'attaque de 1000 textes générés par une machine. L'exactitude sur l'échantillon initial de 1000 textes générés est de 91,10%

Annexe C

Modèle 1

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 1496, 256)	1,536
dropout (Dropout)	(None, 1496, 256)	0
conv1d_1 (Conv1D)	(None, 1492, 256)	327,936
max_pooling1d (MaxPooling1D)	(None, 298, 256)	0
conv1d_2 (Conv1D)	(None, 294, 256)	327,936
conv1d_3 (Conv1D)	(None, 290, 256)	327,936
max_pooling1d_1 (MaxPooling1D)	(None, 145, 256)	0
flatten (Flatten)	(None, 37120)	0
dropout_1 (Dropout)	(None, 37120)	0
dense (Dense)	(None, 256)	9,502,976
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257

FIGURE C.1 – Résumé de notre modèle CNN+POS.

Annexe D

Modèle 2

Model: "functional_12"

Layer (type)	Output Shape	Param #	Connected to
seq_input (InputLayer)	(None, 1500, 1)	0	-
conv1d_4 (Conv1D)	(None, 1496, 256)	1,536	seq_input[0][0]
dropout_3 (Dropout)	(None, 1496, 256)	0	conv1d_4[0][0]
conv1d_5 (Conv1D)	(None, 1492, 256)	327,936	dropout_3[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 298, 256)	0	conv1d_5[0][0]
conv1d_6 (Conv1D)	(None, 294, 256)	327,936	max_pooling1d_2[0][0]
max_pooling1d_3 (MaxPooling1D)	(None, 58, 256)	0	conv1d_6[0][0]
conv1d_7 (Conv1D)	(None, 54, 256)	327,936	max_pooling1d_3[0][0]
max_pooling1d_4 (MaxPooling1D)	(None, 10, 256)	0	conv1d_7[0][0]
flatten_1 (Flatten)	(None, 2560)	0	max_pooling1d_4[0][0]
extra_input (InputLayer)	(None, 768)	0	-
concatenate (Concatenate)	(None, 3328)	0	flatten_1[0][0], extra_input[0][0]
dropout_4 (Dropout)	(None, 3328)	0	concatenate[0][0]
dense_2 (Dense)	(None, 512)	1,704,448	dropout_4[0][0]
dropout_5 (Dropout)	(None, 512)	0	dense_2[0][0]
dense_3 (Dense)	(None, 256)	131,328	dropout_5[0][0]
dropout_6 (Dropout)	(None, 256)	0	dense_3[0][0]
dense_4 (Dense)	(None, 128)	32,896	dropout_6[0][0]
dropout_7 (Dropout)	(None, 128)	0	dense_4[0][0]
dense_5 (Dense)	(None, 64)	8,256	dropout_7[0][0]
dense_6 (Dense)	(None, 1)	65	dense_5[0][0]

FIGURE D.1 – Résumé du deuxième modèle CNN+CLS

Bibliographie

[1] Adelani, D. I., Mai, H., Fang, F., Nguyen, H. H., Yamagishi, J., and Echizen, I. Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. In *Advanced Information Networking and Applications : Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA-2020)* (2020), Springer, pp. 1341–1354.

[2] Aghajanyan, A., Shrivastava, A., Gupta, A., Goyal, N., Zettlemoyer, L., and Gupta, S. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv :2008.03156* (2020).

[3] Baevski, A., Hsu, W.-N., Conneau, A., and Auli, M. Unsupervised speech recognition. *Advances in Neural Information Processing Systems 34* (2021), 27826–27839.

[4] Bakhtin, A., Gross, S., Ott, M., Deng, Y., Ranzato, M., and Szlam, A. Real or fake ? learning to discriminate machine from human generated text. corr abs/1906.03351 (2019). *arXiv preprint arXiv :1906.03351* (2019).

[5] Baki, S., Verma, R., Mukherjee, A., and Gnawali, O. Scaling and effectiveness of email masquerade attacks : Exploiting natural language generation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (2017), pp. 469–482.

[6] Bengio, Y. Neural net language models. *Scholarpedia 3*, 1 (2008), 3881.

[7] Berglund, M., Raiko, T., Honkala, M., Kärkkäinen, L., Vetek, A., and Karhunen, J. T. Bidirectional recurrent neural networks as generative models. *Advances in neural information processing systems 28* (2015).

[8] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language

models are few-shot learners. *Advances in neural information processing systems 33* (2020), 1877–1901.

[9] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. Attention-based models for speech recognition. *Advances in neural information processing systems 28* (2015).

[10] Clark, E., August, T., Serrano, S., Haduong, N., Gururangan, S., and Smith, N. A. All that's' human'is not gold : Evaluating human evaluation of generated text. *arXiv preprint arXiv :2107.00061* (2021).

[11] Clive, J., Cao, K., and Rei, M. Control prefixes for parameter-efficient text generation. *arXiv preprint arXiv :2110.08329* (2021).

[12] Crossley, S. A., Allen, D. B., and McNamara, D. S. Text readability and intuitive simplification : A comparison of readability formulas. *Reading in a foreign language 23*, 1 (2011), 84–101.

[13] Crothers, E., Japkowicz, N., Viktor, H., and Branco, P. Adversarial robustness of neural-statistical features in detection of generative transformers. In *2022 International Joint Conference on Neural Networks (IJCNN)* (2022), IEEE, pp. 1–8.

[14] Crothers, E., Japkowicz, N., and Viktor, H. L. Machine-generated text : A comprehensive survey of threat models and detection methods. *IEEE Access* (2023).

[15] Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models : A simple approach to controlled text generation. *arXiv preprint arXiv :1912.02164* (2019).

[16] Dehouche, N. Plagiarism in the age of massive generative pre-trained transformers (gpt-3). *Ethics Sci. Environ. Politics 21* (Mar. 2021).

[17] Dugan, L., Hwang, A., Trhlik, F., Ludan, J. M., Zhu, A., Xu, H., Ippolito, D., and Callison-Burch, C. Raid : A shared benchmark for robust evaluation of machine-generated text detectors. *arXiv preprint arXiv :2405.07940* (2024).

[18] Fariello, S., Fenza, G., Forte, F., Gallo, M., and Marotta, M. Distinguishing human from machine : A review of advances and challenges in ai-generated text detection. *Int. J. Interact. Multimed. Artif. Intell.* (2024).

[19] Feng, L., Jansche, M., Huenerfauth, M., and Elhadad, N. A comparison of features for automatic readability assessment. In *Coling 2010 : Posters* (2010), pp. 276–284.

[20] Feng, X., Liu, M., Liu, J., Qin, B., Sun, Y., and Liu, T. Topic-to-essay generation with neural networks. In *IJCAI* (2018), pp. 4078–4084.

[21] Flesch, R. A new readability yardstick. *Journal of applied psychology 32*, 3 (1948), 221.

[22] Fröhling, L., and Zubiaga, A. Feature-based detection of automated language models : tackling gpt-2, gpt-3 and grover. *PeerJ Computer Science 7* (2021), e443.

[23] Giaretta, A., and Dragoni, N. Community targeted phishing : A middle ground between massive and spear phishing through natural language generation. In *Proceedings of 6th International Conference in Software Engineering for Defence Applications : SEDA 2018 6* (2020), Springer, pp. 86–93.

[24] Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T., and Saenko, K. Youtube2text : Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proceedings of the IEEE international conference on computer vision* (2013), pp. 2712–2719.

[25] Guo, Z., Jiao, K., Yao, X., Wan, Y., Li, H., Xu, B., Zhang, L., Wang, Q., Zhang, Y., and Mao, Z. Ustc-bupt at semeval-2024 task 8 : Enhancing machine-generated text detection via domain adversarial neural networks and llm embeddings. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)* (2024), pp. 1511–1522.

[26] Hargrave, J. Scigen—an automatic cs paper generator. *MIT, Boston,MA, USA* (2005).

[27] Harkous, H., Groves, I., and Saffari, A. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. *arXiv preprint arXiv :2004.06577* (2020).

[28] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. *arXiv preprint arXiv :1904.09751* (2019).

[29] Hommel, B. E., Wollang, F.-J. M., Kotova, V., Zacher, H., and Schmukle, S. C. Transformer-based deep neural language modeling for construct-specific automatic item generation. *psychometrika 87*, 2 (2022), 749–772.

[30] Jakesch, M., Hancock, J. T., and Naaman, M. Human heuristics for ai-generated language are flawed. *Proceedings of the National Academy of Sciences 120*, 11 (2023), e2208839120.

[31] Jawahar, G., Abdul-Mageed, M., and Lakshmanan, L. V. Automatic detection of machine generated text : A critical survey. *arXiv preprint arXiv :2011.01314* (2020).

[32] Kalchbrenner, N., and Blunsom, P. Recurrent continuous translation models. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (2013), pp. 1700–1709.

[33] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. Ctrl : A conditional transformer language model for controllable generation. *arXiv preprint arXiv :1909.05858* (2019).

[34] Köbis, N., and Mossink, L. D. Artificial intelligence versus maya angelou : Experimental evidence that people cannot differentiate ai-generated from human-written poetry. *Computers in human behavior 114* (2021), 106553.

[35] Kreps, S., McCain, R. M., and Brundage, M. All the news that's fit to fabricate : Ai-generated text as a tool of media misinformation. *Journal of experimental political science 9*, 1 (2022), 104–117.

[36] Kurenkov, A. Lessons from the gpt-4chan controversy. *Gradient* (2022).

[37] Li, G., Zhu, L., Liu, P., and Yang, Y. Entangled transformer for image captioning. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 8928–8937.

[38] Li, H., Zhu, J., Ma, C., Zhang, J., and Zong, C. Multi-modal summarization for asynchronous collection of text, image, audio and video. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), pp. 1092–1102.

[39] Li, Z., Kiseleva, J., and De Rijke, M. Dialogue generation : From imitation learning to inverse reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence* (2019), vol. 33, pp. 6722–6729.

[40] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta : A robustly optimized bert pretraining approach. *arXiv preprint arXiv :1907.11692* (2019).

[41] Lokna, J., Balunovic, M., and Vechev, M. Human-in-the-loop detection of AI-generated text via grammatical patterns, 2024.

[42] Luhn, H. P. The automatic creation of literature abstracts. *IBM Journal of research and development 2*, 2 (1958), 159–165.

[43] Lyons, J. *Natural Language and Universal Grammar : Volume 1 : Essays in Linguistic Theory*, vol. 1. Cambridge University Press, 1991.

[44] Ma, Y., Liu, J., Yi, F., Cheng, Q., Huang, Y., Lu, W., and Liu, X. Ai vs. human–differentiation analysis of scientific content generation. *arXiv preprint arXiv :2301.10416* (2023).

[45] Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., et al. Language models are few-shot learners. *arXiv preprint arXiv :2005.14165* (2020).

[46] Marchitan, T.-g., Creanga, C., and Dinu, L. P. Team Unibuc - NLP at SemEval-2024 task 8 : Transformer and hybrid deep learning based models for machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)* (Mexico City, Mexico, June 2024), A. K. Ojha, A. S. Doğruöz, H. Tayyar Madabushi, G. Da San Martino, S. Rosenthal, and A. Rosá, Eds., Association for Computational Linguistics, pp. 403–411.

[47] Marcus, M., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english : The penn treebank. *Computational linguistics 19*, 2 (1993), 313–330.

[48] Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint arXiv :1708.02182* (2017).

[49] Mikolov, T., et al. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April 80*, 26 (2012).

[50] Nguyen-Son, H.-Q., Tieu, N.-D. T., Nguyen, H. H., Yamagishi, J., and Zen, I. E. Identifying computer-generated text using statistical analysis. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (2017), IEEE, pp. 1504–1511.

[51] Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq : A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv :1904.01038* (2019).

[52] Przybyła, P., Duran-Silva, N., and Egea-Gómez, S. I've seen things you machines wouldn't believe : Measuring content predictability to identify automatically-generated text. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2023). CEUR Workshop Proceedings, CEUR-WS, Jaén, Spain* (2023).

[53] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog 1*, 8 (2019), 9.

[54] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research 21*, 1 (2020), 5485–5551.

[55] Sarvazyan, A. M., González, J.-Á., and Franco-Salvador, M. Genaios at semeval-2024 task 8 : Detecting machine-generated text by mixing language model probabilistic features. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)* (2024), pp. 101–107.

[56] Sarvazyan, A. M., González, J. Á., Franco-Salvador, M., Rangel, F., Chulvi, B., and Rosso, P. Overview of autextification at iberlef 2023 : Detection and attribution of machine-generated text in multiple domains. *arXiv preprint arXiv :2309.11285* (2023).

[57] Schulman, J., Zoph, B., Kim, C., Hilton, J., Menick, J., Weng, J., Uribe, J. F. C., Fedus, L., Metz, L., Pokorny, M., et al. Chatgpt : Optimizing language models for dialogue. *OpenAI blog* (2022).

[58] See, A., Pappu, A., Saxena, R., Yerukola, A., and Manning, C. D. Do massively pretrained language models make better storytellers ? *arXiv preprint arXiv :1909.10705* (2019).

[59] Sheikha, F. A., and Inkpen, D. Generation of formal and informal sentences. In *Proceedings of the 13th European Workshop on Natural Language Generation* (2011), pp. 187–193.

[60] Shu, K., Wang, S., Lee, D., and Liu, H. Mining disinformation and fake news : Concepts, methods, and recent advancements. *Disinformation, misinformation, and fake news in social media : Emerging research challenges and opportunities* (2020), 1–19.

[61] Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv :1908.09203* (2019).

[62] Stiff, H., and Johansson, F. Detecting computer-generated disinformation. *International Journal of Data Science and Analytics 13*, 4 (2022), 363–383.

[63] Sun, S., Zhao, W., Manjunatha, V., Jain, R., Morariu, V., Dernoncourt, F., Srinivasan, B. V., and Iyyer, M. IGA : An intent-guided authoring assistant. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (Online and Punta Cana, Dominican Republic, Nov. 2021), Association for Computational Linguistics, pp. 5972–5985.

[64] Tang, R., Chuang, Y.-N., and Hu, X. The science of detecting llm-generated text. *Communications of the ACM 67*, 4 (2024), 50–59.

[65] Thomas, J., Hoang, G. B., and Mitchell, L. Simple models are all you need : Ensembling stylometric, part-of-speech, and information-theoretic models for the ALTA 2024 shared task. In *Proceedings of the 22nd Annual Workshop of the Australasian Language Technology Association* (Canberra, Australia, Dec. 2024), T. Baldwin, S. J. Rodríguez Méndez, and N. Kuo, Eds., Association for Computational Linguistics, pp. 207–212.

[66] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[67] Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C., and Wang, L. Git : A generative image-to-text transformer for vision and language. *arXiv preprint arXiv :2205.14100* (2022).

[68] Wang, Y., Mansurov, J., Ivanov, P., Su, J., Shelmanov, A., Tsvigun, A., Afzal, O. M., Mahmoud, T., Puccetti, G., Arnold, T., et al. Semeval-2024 task 8 : Multidomain, multimodel and multilingual machine-generated text detection. *arXiv preprint arXiv :2404.14183* (2024).

[69] Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., and Choi, Y. Defending against neural fake news. *Advances in neural information processing systems 32* (2019).

[70] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. Dialogpt : Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv :1911.00536* (2019).

[71] Zhang, Y., and Wallace, B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv :1510.03820* (2015).

[72] Zipf, G. K. *The psycho-biology of language : An introduction to dynamic philology.* Routledge, 2013.

[73] Zipf, G. K. *Human behavior and the principle of least effort : An introduction to human ecology.* Ravenio Books, 2016.