UNIVERSITÉ VIRTUELLE DU BURKINA FASO
(UV-BF)

MÉMOIRE DE MASTER

Apprentissage Profond pour le résumé de documents et la détection de plagiats dans les études de l'administration

---

Auteur :
Abdoul Fataoh KABORE

Encadrant :
Dr. Aminata
ZERBO/SABANÉ (MA)

Superviseur :
Dr. Tegawendé F.
BISSYANDE (MC)

Document de mémoire soumis pour satisfaire aux exigences du Diplôme de Master Fouilles de Données et Intelligence Artificielle

à

Université Virtuelle du Burkina Faso
Sciences du Numérique

Dédié à tous ceux qui m'ont soutenu, inspiré et encouragé tout au long de ce parcours académique, cette dédicace est un témoignage de ma gratitude et de mon appréciation sincère.

Remerciements

Les travaux de ce mémoire sont le fruit de l'accompagnement et du soutien de plusieurs personnes et structures. Je ne saurai donc présenter ces travaux sans exprimer mes sincères remerciements à toutes ces personnes et structures qui, d'une manière ou d'une autre, n'ont ménagé aucun effort pour leur réalisation. Je tiens à présenter ma reconnaissance et ma gratitude de façon particulière :

- au **Centre d'Excellence Interdisciplinaire en Intelligence Artificielle pour le Développement (CITADEL)** et à toute son équipe, pour le cadre de travail et l'accompagnement offerts ;
- à **Dr Aninata Zerbo/Sabané** et à **Dr Tegawendé F. Bissyandé**, pour l'encadrement de ces travaux ;
- à **Dr Rodrique Kafando** pour sa disponibilité et son encadrement pour ces travaux ;
- à l'ensemble des enseignants de la filière Fouilles de Données & Intelligence Artificielle ainsi qu'à toute l'administration de l'Université Virtuelle du Burkina Faso, pour la qualité de la formation reçue ;

Résumé

Le traitement automatique du langage naturel (TALN) a connu un développement considérable ces dernières années, notamment grâce à l'apprentissage profond. Cette approche révolutionnaire permet aux machines de comprendre, analyser et générer du langage naturel de manière plus précise et contextuelle. Grâce à l'apprentissage profond, les modèles de TALN peuvent traiter des tâches complexes telles que le résumé de documents et la détection de plagiat. C'est dans ce contexte que ce projet a été initié dans le but de fournir aux décideurs de l'administration des outils efficaces pour comprendre rapidement l'essentiel du contenu d'un document grâce au résumé automatique, et détecter les éventuelles études non originales ou plagiées. Dans le cadre de ce mémoire, nous avons proposé une approche pour le résumé de documents et la détection de plagiat, en utilisant plusieurs modèles d'apprentissage profond. Notre objectif principal était de sélectionner le modèle le plus approprié afin de garantir des résultats de haute qualité. De plus, nous avons développé une interface utilisateur qui permet une utilisation facile et pratique de ces outils. Les résultats obtenus sont assez satisfaisants, mais ils ouvrent la porte à des perspectives pour des travaux futurs.

Mots-clés : Traitement du langage naturel, résumé de document, détection de plagiat, Apprentissage profond

Abstract

Natural Language Processing (NLP) has witnessed substantial development in recent years, particularly due to the advent of deep learning. This revolutionary approach enables machines to understand, analyze, and generate natural language in a more accurate and contextual manner. Thanks to deep learning, NLP models can handle complex tasks such as document summarization and plagiarism detection. It is within this context that this project was initiated, aiming to provide administrative decision-makers with efficient tools for swiftly understanding the essential content of a document through automatic summarization and detecting potential non-original or plagiarized studies.

In the context of this thesis, we proposed an approach for document summarization and plagiarism detection, utilizing several deep learning models from the state-of-the-art. Our main objective was to select the most suitable model to ensure high-quality results. Moreover, we developed a user-friendly interface that allows for easy and practical use of these tools. The obtained results are quite satisfactory but pave the way for future work perspectives.

Keywords: Natural language processing, document summary, plagiarism detection, Deep Learning

Table des matières

Table des figures

Liste des tableaux

Liste des Abbreviations

IA	**I**ntelligence **A**rtificielle
NLP	**N**atural **L**anguage **P**rocessing
PDF	**P**ortable **D**ocument **F**ormat
CNN	**C**onvolutional **N**eural **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
RNN	**R**ecurrent **P**arametric **N**etwork

Introduction générale

0.1 Contexte

Le traitement automatique du langage naturel (TALN), très connu sous l'acronyme NLP (Natural Language Processing) est un domaine de recherche en pleine expansion qui permet de comprendre, de traiter et de produire des textes de manière automatique. Dans ce contexte, l'utilisation de l'apprentissage profond pour le résumé de documents et la détection de plagiats dans les études de l'administration représente une avancée importante pour les professionnels du domaine.

En effet, l'apprentissage profond est une méthode d'apprentissage automatique qui permet de modéliser les données de manière hiérarchique et non linéaire à l'aide de réseaux de neurones artificiels. Cette méthode a été utilisée avec succès pour de nombreuses applications en TALN [7], y compris le résumé de documents et la détection de plagiats.

Dans le contexte des études de l'administration, le résumé de documents est une tâche importante pour les professionnels qui doivent lire et comprendre de nombreux documents, tels que des rapports, des contrats ou des dossiers d'entreprises. Le résumé de documents permet de gagner du temps et de faciliter la prise de décisions.

La détection de plagiats est également une tâche essentielle pour éviter la fraude académique et professionnelle. Dans les études de l'administration, les travaux universitaires ou les documents professionnels doivent être originaux et ne pas contenir de contenu copiés ou plagiés.

Dans ce contexte, comment pourrions-nous exploiter les modèles d'apprentissage profond pour résumer efficacement les documents et détecter le plagiat dans les études administratives ?

0.2 Objectif

L'objectif principal de cette étude est de développer des systèmes basés sur l'apprentissage profond pour résumer des documents et détecter les cas de

plagiats dans les études de l'administration.

Comme objectifs spécifiques, nous souhaitons dans un premier temps explorer l'état de l'art des approches d'apprentissage profond pour le résumé de documents et la détection de plagiats dans les études de l'administration, en prenant en considération les défis et les particularités spécifiques à ce domaine ;

dans un deuxième temps, développer ou exploiter des modèles d'apprentissage profond spécifiquement adapté pour le résumé de document et la détection de plagiat ;

et dans un troisième temps, proposer une solution applicative pour réaliser ses deux taches.

La suite de ce mémoire est donc organisée comme suit : chapitre 1 : État de l'art. Nous allons faire un tour d'horizon sur le résumé de document, la détection de plagiat, les techniques et les travaux existants en lien avec notre thème. Ces concepts nous permettront de mieux situer notre travail de recherche. Ensuite, nous allons évoquer la démarche méthodologique choisie. Dans le chapitre 2, nous parlons de notre démarche qui consiste à expliquer notre approche éthologique en vue d'atteindre nos objectifs. Dans le chapitre 3, nous présenterons nos implémentations. Les résultats et interprétations. Nous finirons par une conclusion générale et les perspectives.

Chapitre 1

État de l'art

Dans ce chapitre, nous faisons le point sur l'état des connaissances, les différents acquis, outils et travaux concernant le résumé de document et la détection de plagiat. Nous abordons le traitement du langage naturel (NLP), également appelé natural language processing en anglais, qui est une branche de l'Intelligence Artificielle (IA) axée sur l'analyse et la compréhension du langage humain et l'apprentissage profond qui une technique utilisant des réseaux de neurones artificiels pour résoudre des tâches complexes en intelligence artificielle.

1.1 Généralités

L'intelligence artificielle (IA) est un ensemble de techniques qui imitent l'intelligence humaine en créant et en appliquant des algorithmes dans un environnement informatique dynamique. Elle est actuellement utilisée dans divers domaines tels que la vision par ordinateur, les systèmes experts, mais aussi dans le traitement du langage naturel avec des tâches comme la classification de texte, l'extraction d'informations pertinentes, la génération de texte, etc.

L'apprentissage profond (Deep Learning) est une sous-catégorie de l'apprentissage automatique qui utilise des réseaux de neurones artificiels pour apprendre à partir de données et pour effectuer des tâches complexes nécessitant plusieurs quantités de données [2]. Les réseaux de neurones profonds sont des architectures de réseaux de neurones artificiels à plusieurs couches qui sont capables d'apprendre des représentations hiérarchiques des données d'entrée.

L'apprentissage profond est devenu l'une des technologies clés de l'IA, en raison de sa capacité à apprendre des tâches complexes à partir de données non structurées. Il est utilisé avec succès dans de nombreux secteurs, tels la reconnaissance faciale, la prédiction de données, et dans le traitement automatique du langage naturel.

Dans nos travaux, nous exploitons les avancées du traitement du langage naturel avec l'apprentissage profond pour des tâches de résumé de document et de la détection de plagiat dans les études de l'administration.

1.2 Le traitement automatique du langage naturel (TALN)

Le NLP [18] dans la langue anglaise est une branche de l'intelligence artificielle qui vise à permettre aux machines de comprendre, d'interpréter et de générer le langage naturel utilisé par les humains dans leur communication. Le TALN s'appuie sur des algorithmes et des modèles statistiques pour analyser et traiter les données linguistiques.

Les applications du TALN sont nombreuses, allant de la traduction automatique de textes à la reconnaissance vocale [8] en passant par la génération de réponses automatiques dans les systèmes de chatbots [28]. Les techniques utilisées dans le TALN incluent l'analyse syntaxique, la reconnaissance d'entités nommées, la classification de textes, la modélisation de thèmes, etc.

Le TALN est un domaine en constante évolution, avec des progrès importants réalisés ces dernières années grâce à l'avènement des réseaux de neurones profonds et de l'apprentissage automatique supervisé. Cependant, le TALN reste un domaine complexe qui nécessite des connaissances solides en linguistique et en informatique pour être maîtrisé [9].

1.2.1 Le résumé de document

Le résumé de document en NLP [1] utilise des techniques d'analyse de texte pour créer des résumés de manière automatique. Le NLP se concentre sur l'analyse des structures et des modèles dans le langage naturel pour comprendre le sens du texte et extraire les informations importantes.

Les approches NLP qui traitent des résumés peuvent être divisées en deux catégories principales : le résumé extractif et le résumé abstrait [20]. Le résumé extractif[20] implique de sélectionner les phrases clés du document original et de les assembler pour créer un résumé. Le résumé abstrait[20], quant à lui, consiste à générer un résumé qui ne contient pas nécessairement les mêmes phrases que le document original, mais qui rend compte des informations les plus importantes de manière plus condensée.

Les techniques de NLP pour le résumé de document utilisent divers algorithmes, notamment :

— Les modèles de langage pré-entraînés, qui sont entraînés sur de grandes quantités de données textuelles avant d'être utilisés pour résoudre des tâches spécifiques de traitement du langage naturel. Ces modèles utilisent souvent des réseaux de neurones profonds tels que les réseaux de neurones à convolution (CNN), les réseaux de neurones récurrents (RNN) ou encore les transformers, qui sont des architectures plus récentes.

— Les techniques de clustering, qui permettent de regrouper les documents en fonction de leur similarité. Ces méthodes permettent de trouver des groupes de documents liés entre eux, ce qui peut être utile pour le résumé de document [29].
— Les réseaux de neurones, qui sont utilisés pour modéliser les relations complexes entre les mots et les phrases. Les réseaux de neurones peuvent être utilisés dans les approches de résumé extractif et abstrait, où ils apprennent à identifier les parties les plus importantes du texte à résumer.
— Les transformers sont également utilisés pour le résumé de document. Les modèles de résumé basés sur les transformers, tels que BART (Bidirectional and Auto-Regressive Transformers) [12] et T5 (Text-to-Text Transfer Transformer) [23], sont capables de générer des résumés de haute qualité en exploitant les mécanismes d'attention et en utilisant des architectures transformer. Ces modèles peuvent être entraînés de manière supervisée sur des données de résumé existantes ou de manière non supervisée en utilisant des objectifs d'apprentissage auto-supervisés.

1.2.2 La détection de plagiat

La détection de plagiat en NLP est une application importante du traitement du langage naturel qui permet d'identifier le contenu qui a été copié à partir d'autres sources et utilisé sans autorisation ou citation appropriée.

Les techniques de détection de plagiat en NLP reposent souvent sur l'analyse de similarité entre les documents, en comparant les phrases ou les passages textuels. Plusieurs méthodes peuvent être utilisées pour mesurer la similarité entre les documents, comme la similarité de cosinus, la distance de Jaccard, ou la distance d'édition. Ces différentes méthodes utilisent généralement en entrée des représentations vectorielles des mots issue de la technique du word embedding du NLP.

La détection de plagiat en NLP est souvent utilisée dans les milieux académiques pour identifier les cas de plagiat dans les travaux étudiants, ainsi que dans les milieux professionnels pour détecter les fraudes et les contrefaçons. Les outils de détection de plagiat en NLP peuvent être utilisés en ligne ou en local pour analyser de grands volumes de données textuelles et générer des rapports de similarité pour chaque paire de documents.

Cependant, la détection de plagiat en NLP peut être difficile en raison de la diversité et de la complexité du langage naturel. Les auteurs de documents peuvent changer de mots, de phrases et de structures de phrases pour éviter d'être détectés, ce qui rend la tâche de détection de plagiat plus complexe. En outre, la détection de plagiat en NLP peut entraîner des faux positifs et des faux négatifs, ce qui nécessite une évaluation humaine pour confirmer la présence ou l'absence de plagiat.

1.3 Les modèles pré-entraînés et les transformers

Les modèles pré-entraînés en NLP sont des modèles qui ont été entraînés sur de grands corpus de texte afin de capturer des informations sur la structure linguistique et sémantique du langage naturel. Ces modèles sont ensuite finetunés sur des tâches spécifiques telles que la classification de texte, la génération de texte, etc.

Les transformers sont une famille d'architectures de réseaux de neurones utilisées dans les modèles pré-entraînés pour le traitement du langage naturel, tels que BERT [5], GPT-3 [3] et RoBERTa [16]. Les transformers sont conçus pour capturer les relations à long terme entre les éléments d'une séquence, tels que les mots d'une phrase.

Les transformers[30] utilisent une technique appelée attention pour déterminer l'importance relative des éléments de la séquence en fonction du contexte global. L'attention permet de donner plus de poids aux éléments importants de la séquence et moins de poids aux éléments moins importants. Cette approche permet de capturer des relations à long terme entre les éléments de la séquence, ce qui est particulièrement important dans les tâches de traitement du langage naturel.

Les modèles pré-entraînés basés sur des transformers ont obtenu des résultats impressionnants dans de nombreuses tâches de NLP, dépassant souvent les performances des modèles précédents basés sur des réseaux de neurones récurrents. Ces modèles pré-entraînés sont souvent utilisés comme point de départ pour finetuner sur des tâches spécifiques.

La Figure 1.1 présente l'architecture des modèles Transformer

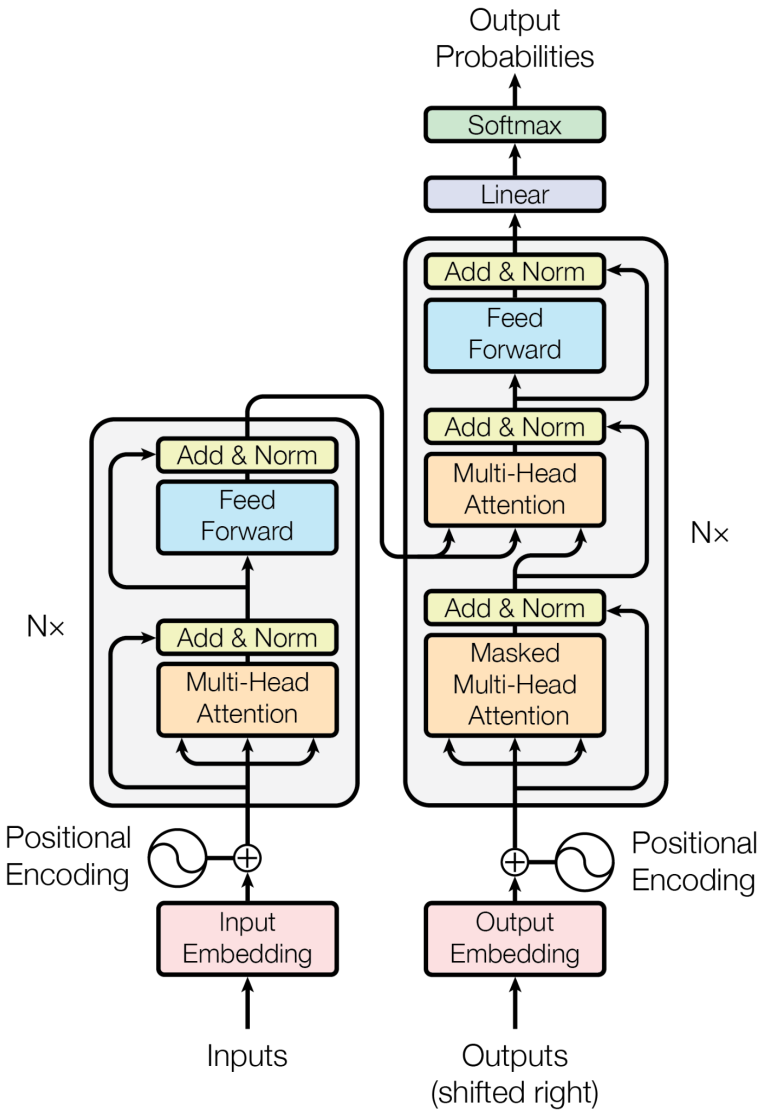


FIGURE 1.1 – Architecture des modèles Transformer [30]

1.4 Les techniques de représentation vectorielle des mots

La machine, pour comprendre le langage humain, a besoin d'une représentation vectorielle des mots du langage humain. Il existe plusieurs méthodes pour représenter un mot sous forme d'un vecteur dont les trois principales sont :

— Le One-Hot Encoding (encodage one-hot) [25] qui consiste à créer un vecteur de la taille du vocabulaire, où toutes les valeurs sont à zéro sauf une, qui correspond à l'indice du mot dans le vocabulaire. Elle est simple mais ne capture pas la sémantique du mot.
— Word Embeddings (incorporation de mots) [11] : Les embeddings de mots sont des représentations vectorielles d'un mot qui capturent la

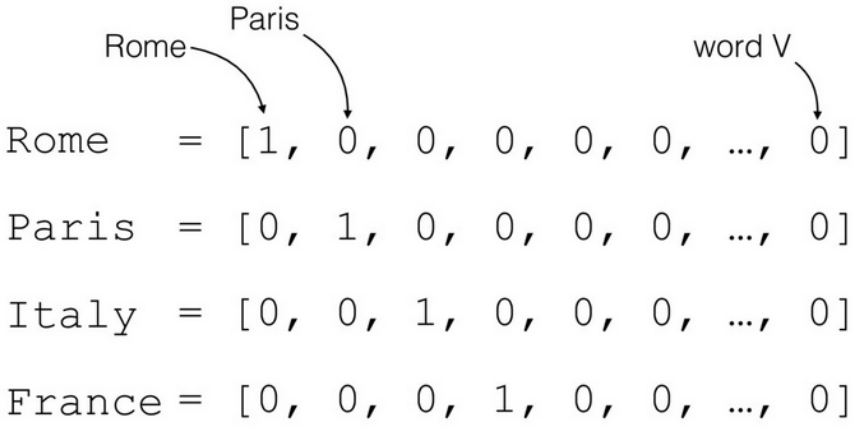


FIGURE 1.2 – Exemple One-Hot Encoding

signification sémantique et contextuelle du mot en fonction de son utilisation dans un corpus de textes. Les embeddings de mots sont généralement calculés à l'aide de techniques d'apprentissage automatique, telles que Word2Vec, GloVe et FastText

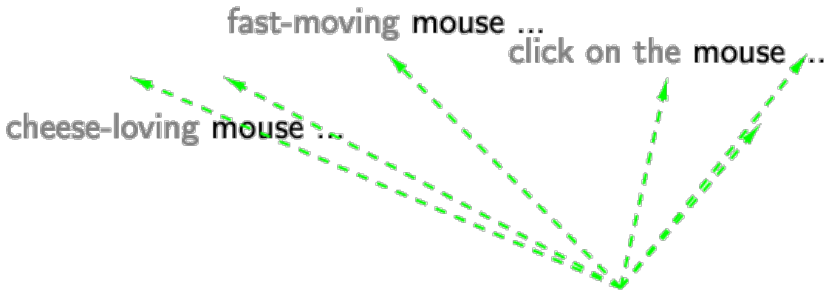


FIGURE 1.3 – Exemple du Word Embeddings

— Contextualized Word Embeddings (incorporation de mots contextualisés) : Les embeddings de mots contextualisés, tels que BERT, ELMo et GPT, sont des représentations vectorielles de mots qui prennent en compte le contexte dans lequel le mot est utilisé. Ces modèles utilisent des réseaux de neurones profonds pour calculer les embeddings de mots en utilisant l'ensemble de la phrase ou du document dans lequel ils apparaissent.

Le tableau 1.1 résume les avantages et les inconvénients de ces méthodes

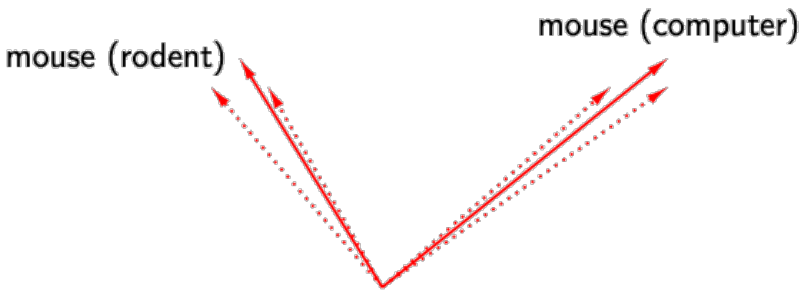


FIGURE 1.4 – Exemple du contextualized Word Embedding

Methodes / Techniques	Avantages	Inconvénients
Encodage One-Hot (One-Hot Encoding)	-Simple à mettre en œuvre -Permet une représentation binaire claire et concise des mots dans un dictionnaire. Embeddings de mots (Word Embeddings) Embeddings de mots contextualisés -Utile pour les modèles peu complexes ou pour des tâches simples de classification.	-Ne capture pas la sémantique des mots et ne peut pas gérer les synonymes et les mots similaires. -Peut être coûteuse en termes de mémoire si le dictionnaire est grand

Embeddings de mots (Word Embeddings)	-Capturent les similarités sémantiques et syntaxiques entre les mots, ce qui permet aux modèles de langage de mieux comprendre le sens du texte -Peuvent être calculés à l'aide de techniques de Machine Learning telles que Word2Vec, GloVe ou FastText, et peuvent être utilisés pour entraîner des modèles plus performants pour de nombreuses tâches de NLP.	-Ne prennent pas en compte le contexte dans lequel les mots apparaissent, ce qui peut limiter leur capacité à comprendre des phrases complexes
Embeddings de mots contextualisés	-Prennent en compte le contexte dans lequel les mots apparaissent ce qui leur permet de mieux comprendre la signification des phrases et des documents entiers. -Utiles pour les tâches de NLP plus complexes telles que la compréhension de la langue naturelle, la traduction automatique, la génération de texte, etc.	-Complexes à calculer et nécessitent des modèles de langage plus avancés tels que BERT, GPT, etc. -Nécessitent plus de ressources de calcul et de mémoire, ce qui peut les rendre moins pratiques pour les modèles de production

TABLE 1.1 – tableau comparatif des techniques de représentation vectorielle des mots

1.5 Les outils de résumé de document et de détection de plagiat actuels

1.5.1 Les outils de résumé de document

Il existe de nombreux outils de résumé de documents. Nous pouvons en citer :

— SummarizeBot [1]. une application en ligne qui utilise des modèles de Deep Learning pour générer des résumés de documents, des articles de presse, des emails, etc. La plateforme prend en charge plusieurs langues et propose également une API pour les développeurs.

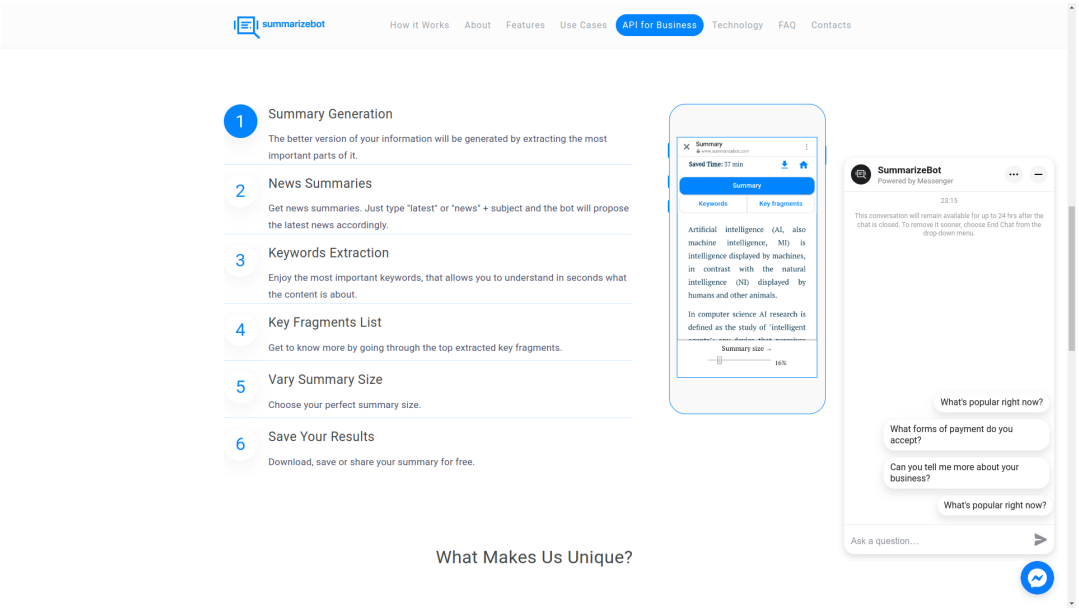


FIGURE 1.5 – Interface de SummarizeBot

— GPT-3 (Generative Pre-trained Transformer 3) [2] qui est un modèle de langage de traitement du langage naturel développé par OpenAI. Il peut être utilisé pour générer des résumés de documents en se basant sur une analyse de contexte et une compréhension sémantique des textes.
— -DeepSum [3], une application en ligne de résumé de documents qui utilise des modèles de Deep Learning pour extraire les informations les plus importantes d'un texte source et les résumer en quelques phrases ou en quelques paragraphes.

Ces outils sont généralement payants ou partiellement payants. Il en existe également pour la détection de plagiat.

1. https://www.summarizebot.com/
2. https://chat.openai.com/
3. https://deepsum.com/

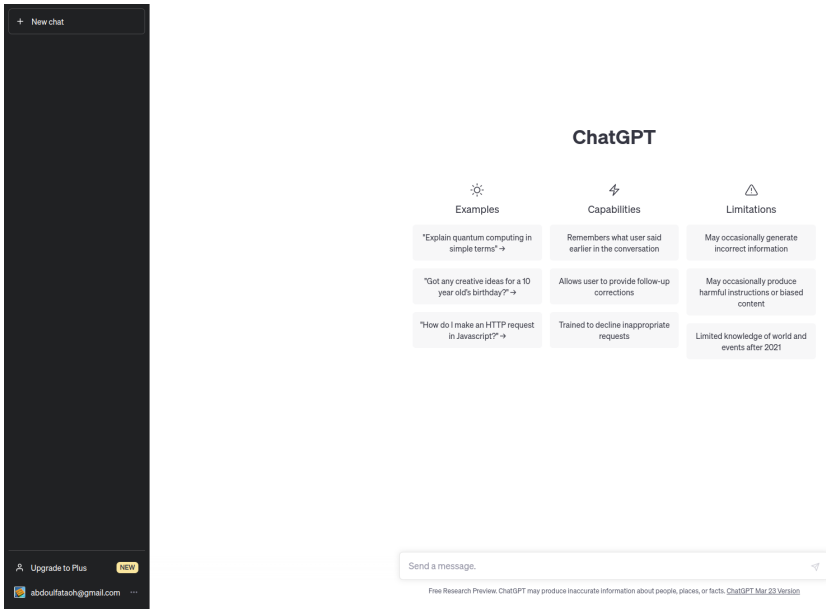


FIGURE 1.6 – Interface de GTP-3

1.5.2 Les outils de détection de plagiat

Il existe de nombreux logiciels et applications en ligne de détection de plagiat. Ce sont par exemple :

— Turnitin [4], c'est un outil qui utilise une technologie de vérification de similarité qui compare le texte soumis à une base de données en ligne de sources académiques, de publications en ligne et de travaux précédents pour détecter tout plagiat potentiel. Turnitin fournit également des commentaires sur la grammaire, la syntaxe et l'organisation du texte.
— PlagScan [5] est un outil de détection de plagiat qui utilise une technologie de recherche en ligne pour comparer le texte soumis avec des milliards de sources en ligne pour détecter tout plagiat potentiel. Le logiciel fournit également des rapports détaillés sur les similitudes trouvées et peut être intégré à d'autres systèmes tels que Moodle.
— Copyscape [6] permet de vérifier le contenu d'un site Web pour détecter tout contenu dupliqué. Il existe une version gratuite qui vous permet de vérifier un seul article à la fois, ainsi qu'une version payante qui vous permet de vérifier de grands volumes de contenu.

1.6 Les travaux dans ces domaines

Au cours de notre étude, nous avons découvert des travaux de recherche qui utilisent le machine learning classique ainsi que le deep-learning pour

4. https://www.turnitin.com/fr
5. https://www.plagscan.com/fr/
6. https://www.copyscape.com/

réaliser du résumé de documents et de la détection de plagiat. Cependant, nous avons choisi de nous concentrer sur les travaux qui utilisent le deep-learning.

1.6.1 Les travaux de recherches dans le domaine du résumé de documents

Dans le domaine du résumé de document nous pouvons citer les travaux suivants :

— "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond" par Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre et Bing Xiang, publié en 2016. Ce papier propose une approche de résumé de texte abstrait en utilisant des réseaux de neurones récurrents séquence-à-séquence et en intégrant des mécanismes d'attention pour améliorer la qualité des résumés générés. [19]

— "Get To The Point : Summarization with Pointer-Generator Networks" par Abigail See, Peter J. Liu et Christopher D. Manning, publié en 2017. Les auteurs présentent une approche de résumé de texte à l'aide d'un réseau de neurones générateur-pointer, qui est capable de sélectionner des mots clés importants du texte source et de les utiliser dans la génération du résumé. [26]

— "Fine-tune BERT for Extractive Summarization" par Yang Liu et Mirella Lapata, publié en 2019. Ce papier explore l'utilisation de BERT, un modèle de transformer, pour le résumé de texte extractif. Les auteurs ont adapté le modèle pour effectuer une classification binaire sur les phrases du texte source, en utilisant les scores de BERT pour identifier les phrases les plus importantes et les inclure dans le résumé. [15]

— "Pretrained Transformers for Text Ranking : BERT and Beyond" par Wei Li et Yunbo Cao, publié en 2020. Ce papier explore l'utilisation de modèles de transformers pré-entraînés tels que BERT et RoBERTa pour le résumé de texte extractif et abstrait. Les auteurs ont également proposé une nouvelle méthode de pondération pour améliorer la qualité des résumés générés. [14]

1.6.2 Les travaux de recherches dans le domaine de la détection de plagiat

Il existe également pour le domaine de la détection de plagiat des travaux de recherche en deep-learning.

— "Plagiarism Detection using Deep Learning Techniques" par Shivendra K. Kushwah et Gaurav Singh, publié en 2018. Ce papier présente une approche de détection de plagiat basée sur des réseaux de neurones convolutionnels et des algorithmes de traitement de texte, tels que TF-IDF. [10]

— "Deep Plagiarism Detection in Code Submissions with Embeddings and Graph Convolutional Networks" par Christopher S. Corley, Arun Lakhotia et Juan M. Segarra, publié en 2020. Ce papier utilise des réseaux de neurones convolutionnels et des graphes convolutionnels pour détecter le plagiat dans les soumissions de code des étudiants. [4]
— "Deep Learning based Plagiarism Detection Using Siamese Neural Networks" par Sharmili Priyadarsini et A. Sai Sabitha, publié en 2021. Ce papier propose une approche de détection de plagiat basée sur des réseaux de neurones siamois pour comparer les textes et identifier les similitudes. [22]

Dans ce chapitre nous avons fait un tour d'horizon sur les connaissances acquis en lien avec l'apprentissage profond appliqué sur les domaines du résumé automatique de document et de la détection du plagiat. Nous avons présenté les outils existant ainsi que les travaux de recherches déjà réalisées. Dans le chapitre suivant, nous aborderons notre approche méthodologique.

Chapitre 2

Approche méthodologique

Dans ce chapitre, nous présentons l'approche pour la mise en place de nos systèmes de résumé de document et de détection de plagiat. Nous détaillerons l'architecture de chacune de ces deux (2) parties tout en présentant les modèles, les jeux de données ayant servis à entraîner ces modèles et les outils que nous avons choisis pour l'implémentation.

2.1 Le système de résumé de document

Dans cette partie du résumé de document, nous présentons une méthodologie qui permet de résumer les documents par section en sélectionnant un modèle parmi plusieurs modèles de deep learning. L'objectif est d'identifier le modèle le mieux adapté en fonction du contexte de chaque document afin de générer des résumés pertinents et de haute qualité. En utilisant cette approche, nous sommes en mesure de tirer parti des différents modèles de deep learning disponibles, chacun ayant ses propres forces et spécialités. La figure 2.1 met en évidence le pipeline utilisé par le système de résumé de document. Elle est constituée d'une (01) entrée qui est le document à résumé au format PDF, d'une (01) sortie qui correspond au document résumé au format PDF et de quatre (04) étapes de traitements :

— L'extraction des sections de texte du document (Text Sections Extractor)
— Le chargement et traitement des données extraites (DataLoader)
— Les modèles
— L'écriture du résumé (Writer)

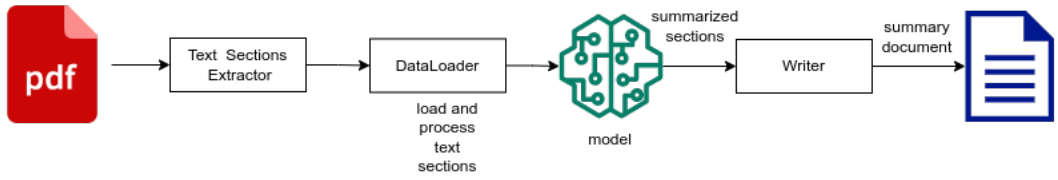


FIGURE 2.1 – Pipeline du système de résumé document

2.1.1 L'extraction des sections de texte du document

L'extraction des sections de texte est une étape cruciale dans le processus de résumé de document, car elle permet d'identifier et d'extraire le contenu textuel des sections du document à résumer. Les sections à considérer pour le résumé sont celles avec un nombre de mots supérieur à Sminmots (nombre minimum de mots par section à considérer pour le résumé). Autrement, le système décide de résumer une section si son nombre de mots est supérieur ou égale à Sminmots . Nous proposons de fixer par défaut nous fixons $S_{minmots}$ à 200 mots tout en laissant la possibilité à l'utilisateur final du système de modifier cette valeur.

Nous extrayons en plus du contenu textuelle, des métadonnées tel que que le numéro de page, le numéro de la section les coordonnées qui délimite la section. Ces informations sont utiles car elles nous permettront plus tard de de réécrire chaque résumé à son emplacement.

Pour manipuler les fichiers PDF nous avons implémenté un module Pdf basé sur le package PyMuPDF qui offre des fonctionnalités puissantes pour extraire le texte à partir de fichiers PDF, ce qui en fait un choix populaire pour cette tâche.

La figure 2.2 illustre l'extraction des sections de texte document.

2.1.2 Le chargement et traitement des données extraites

Dans cette partie, nous définissons un mécanisme permettant de charger les sections de texte extraites à partir du document à résumer, puis de les nettoyer en supprimant les nombres, les caractères non imprimables, les espaces multiples, et nous transformons le tout en minuscule.

Le nettoyage des données est crucial pour garantir que seules les informations pertinentes et utilisables sont utilisées lors de la génération des résumés. En éliminant les éléments indésirables, on réduit le bruit et on améliore la qualité des données d'entrée pour le modèle de Deep Learning.

Une fois que les données extraites ont été nettoyées, elles sont prêtes à être transmises au modèle de Deep Learning pour la génération des résumés. Ces données serviront d'entrée au modèle, qui apprendra à partir de ces exemples pour générer des résumés précis et pertinents.

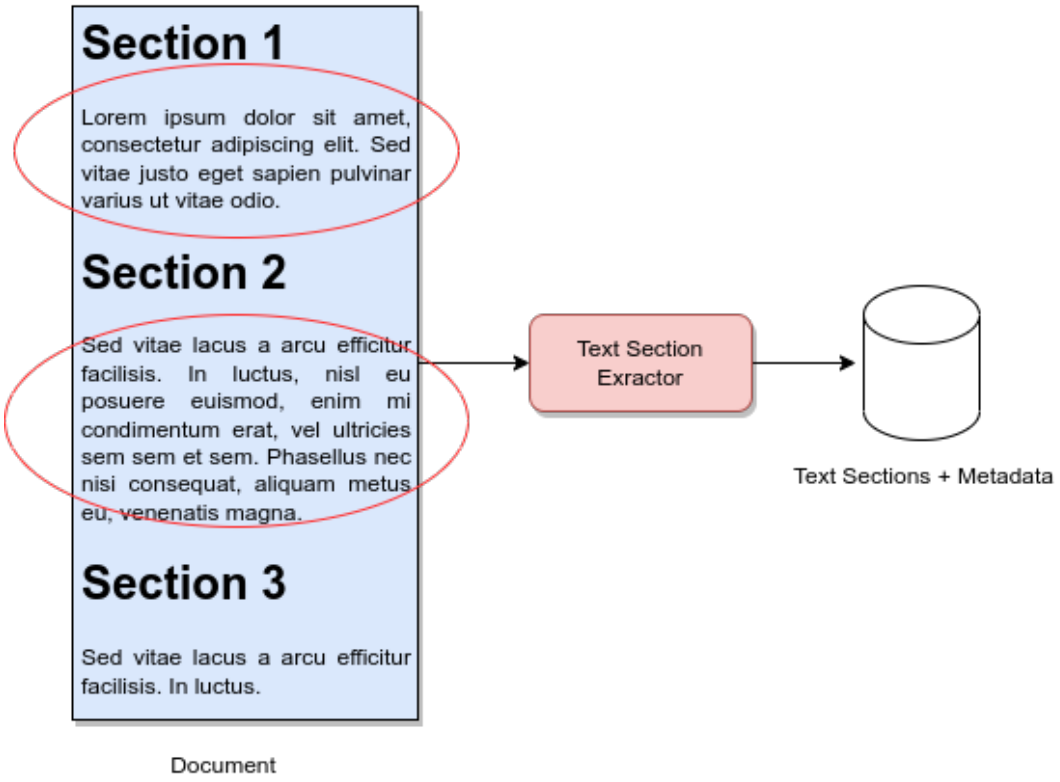


FIGURE 2.2 – Extraction de sections (paragraphes)

2.1.3 Les modèles

Dans cette partie du pipeline, nous proposons de choisir un modèle parmi plusieurs modèles de Deep Learning pour effectuer la tâche de résumé de document. Nous avons également conçu le système de telle sorte à faciliter l'intégration de nouveaux modèles dans le futur. Parmi les modèles disponible pour le résumé de texte, on trouve :

— Le modèle Barthez Orange [6]
— Le text-davinci-003 de OpenAi [21]

2.1.3.1 Le modèle Barthez Orange

BARThez est un modèle pré-entraîné de séquence à séquence pour la langue française, basé sur l'architecture BART (Bidirectional and AutoRegressive Transformers). Contrairement aux modèles existants basés sur BERT pour la langue française, tels que CamemBERT et FlauBERT, BARThez est particulièrement adapté aux tâches génératives, telles que la génération de résumés abstraits, car non seulement son encodeur mais aussi son décodeur sont pré-entraînés.

Ce modèle a été pré-entraîné en apprenant à reconstruire une phrase d'entrée corrompue. Un corpus de 66 Go de texte brut en français a été utilisé pour mener à bien cet entraînement préliminaire. Pendant le pré-entraînement,

le modèle apprend à capturer les dépendances contextuelles dans la langue française et à générer des représentations de haute qualité pour les phrases en français.

L'architecture BART utilisée par BARThez est basée sur des transformers, qui sont des réseaux de neurones puissants et efficaces pour le traitement du langage naturel. Les transformateurs permettent de modéliser les relations complexes entre les mots et d'exploiter les informations contextuelles pour générer des séquences de sortie précises et cohérentes.

L'avantage clé de BARThez réside dans sa capacité à effectuer des tâches génératives, telles que la génération de résumés abstraits. En ayant à la fois un encodeur et un décodeur pré-entraîné, BARThez est capable de comprendre le contexte et de générer des résumés de haute qualité en français.

Grâce à son entraînement préalable sur un vaste corpus de texte en français, BARThez est un modèle prometteur pour diverses tâches de traitement du langage naturel en français, y compris la traduction automatique, la génération de texte et, bien sûr, le résumé abstrait de texte.

2.1.3.2 Le modèle text-davinci-003 d'OpenAI

Le modèle text-davinci-003 d'OpenAI est l'un des modèles de langage basés sur l'architecture GPT-3.5 offerts par OpenAI en accès via API. Ce modèle bénéficie d'une formation extensive sur une vaste quantité de données textuelles provenant de diverses sources, ce qui lui confère des capacités avancées de traitement du langage naturel.

Le modèle text-davinci-003 est connu pour sa capacité à générer du texte de manière fluide et naturelle [1]. Il peut produire des réponses contextuellement appropriées et des contenus de haute qualité pour une variété de tâches . Il présente d'excellents résultats dans la contraction ou résumé de texte.

L'utilisation du modèle text-davinci-003 peut être coûteuse en raison des ressources nécessaires pour son entraînement et son déploiement. Les frais associés peuvent limiter son accessibilité à certains cas d'utilisation.

2.1.4 L'écriture du résumé ou la création du document résumé

La création du document résumé est l'étape finale du pipeline. En plus des fonctionnalités d'extraction de texte des sections et de métadonnées de notre module Pdf, nous avons ajouté des fonctionnalités d'écriture de texte à notre module. Les sections de texte résumées par le modèle choisi sont ré-écrites dans les emplacements correspondants grâce aux métadonnées notamment les coordonnées. Ceci permet donc de restituer les sections résumées du document sans affecter sa mise en page. La figure 2.3 cette approche

1. https://scale.com/blog/gpt-3-davinci-003-comparison

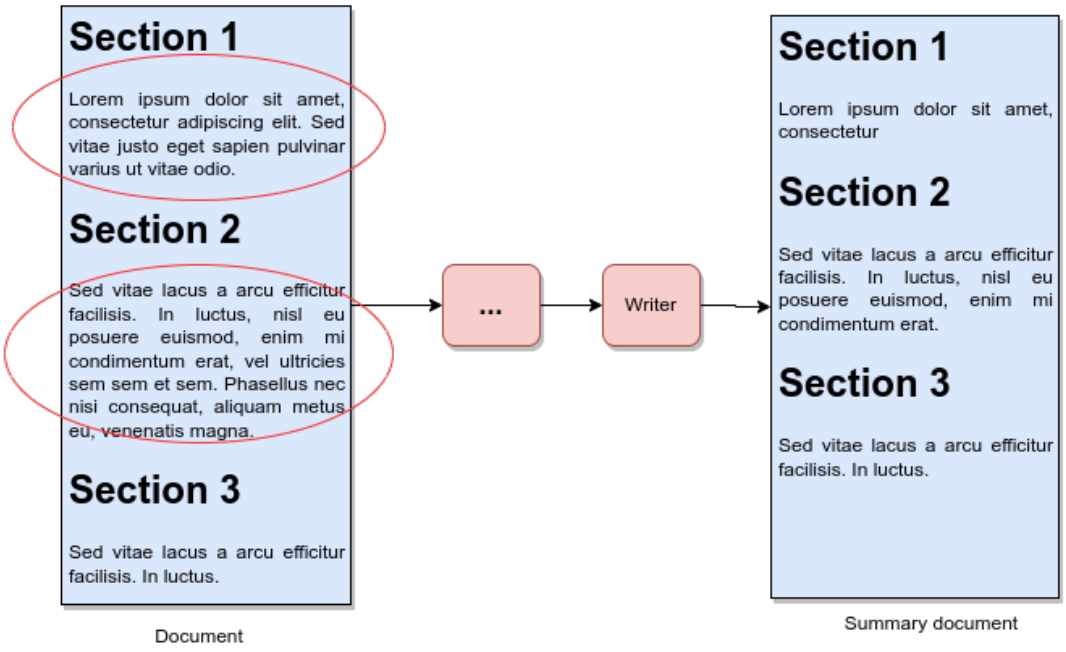


FIGURE 2.3 – Ecriture du résumé ou la création du document résumé

2.2 Le système de détection de plagiat

Dans cette partie tout comme dans le résumé de document, nous proposons de choisir un modèle parmi plusieurs modèles de Deep Learning pour détecter le plagiat dans les documents en analysant les paragraphes. Le plagiat est un problème courant dans de nombreux domaines, et l'utilisation de modèles de Deep Learning peut aider à identifier les similitudes et les emprunts non autorisés entre différents textes. Pour être plus précis, dans ce projet nous utilisons des modèles de deep-learning pour vectoriser les paragraphes afin de permettre à la machine de comprendre le sens puis nous utilisons d'autres techniques pour détecter le plagiat. Nous reviendrons plus tard sur ces techniques. Nous présenterons dans le premier sous point suivant ces modèles permettant de vectoriser les sections de texte.

Également dans notre projet qui consiste à détecter le plagiat dans les études de l'administration nous avons besoin d'un corpus de documents d'études administratifs déjà existant qui servira de base de données comparative pour le document à vérifier. Nous avons reçu à cet effet une collection de documents administratifs que nous présenterons dans un sous point. Nous aborderons aussi la création et la sauvegarde des vecteurs (embeddings) dans un système de cache pour optimiser les performances du système.

Enfin, nous présenterons le pipeline permettant de détecter le plagiat dans un document.

2.2.1 Les modèles vectorisation de document

La détection de plagiat passe par une première phase qui consiste à transformer les parties des documents (corpus et document à vérifier) en représentations vectorielles ou embeddings afin de permettre à la machine de comprendre le sens.Pour ce faire, nous avons entraîné un premier modèle basé sur Doc2vec et aussi nous utilisons des modèles pré-entraînés pour réaliser cette tâche de vectorisation qui sont : all-MiniLM-L6-v2 [27], distiluse-base-multilingual-cased-v1 [24], et camembert-large [17].

2.2.1.1 Doc2vec

Doc2Vec est une technique d'apprentissage automatique développée dans la bibliothèque gensim. C'est une extension du célèbre modèle Word2Vec qui permet de représenter les documents sous forme de vecteurs denses.

Contrairement aux modèles Word2Vec qui se concentrent sur l'apprentissage des représentations vectorielles des mots, Doc2Vec vise à capturer les relations sémantiques entre les documents. Il attribue à chaque document un vecteur unique qui encapsule son sens et sa signification.

Le modèle Doc2Vec utilise une architecture de réseaux de neurones pour apprendre ces représentations vectorielles(figure 2.4). Il prend en compte le contexte des mots dans le document ainsi que les interactions entre les documents lors de l'apprentissage. En conséquence, les vecteurs de document obtenus avec Doc2Vec sont capables de capturer des informations sémantiques complexes et de représenter les similarités entre les documents.

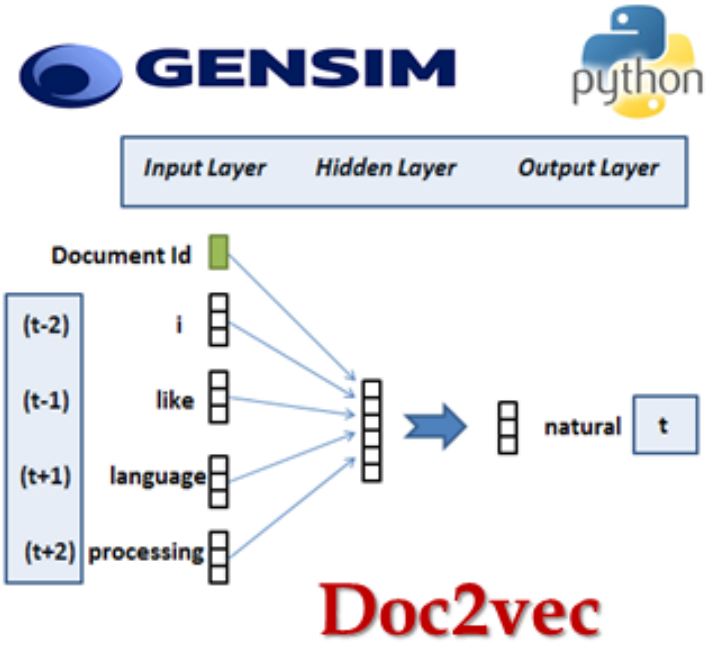


FIGURE 2.4 – Doc2vec avec Gensim

Pour entraîner notre modèle Doc2Vec, nous extrayons d'abord les paragraphes des documents de notre corpus, ensuite nous le nettoyons en supprimant les caractères spéciaux, les ponctuations, les symboles inutiles, les mots fréquents ou mots vides. Nous normalisons aussi les paragraphes en les transformant tous en minuscule puis nous chargeons tout ceci avec le chargeur de données vers le modèle pour entraînement. Le modèle entraîné est alors sauvegardé pour une utilisation ultérieure. L'ensemble du pipeline d'entraînement est présente dans la figure 2.5

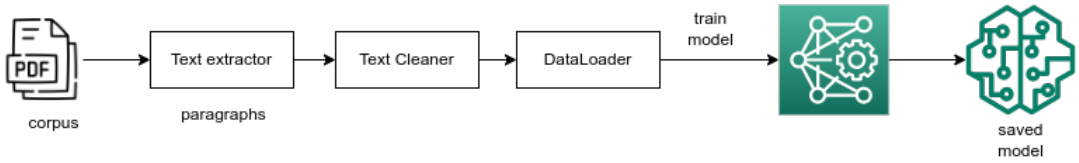


FIGURE 2.5 – Pipeline d'entraînement du modèle Doc2vec

2.2.1.2 All-MiniLM-L6-v2

Le modèle all-MiniLM-L6-v2 a été développé dans le cadre d'un projet visant à entraîner des modèles d'encodage de phrases sur de très grands ensembles de données au niveau des phrases en utilisant un objectif d'apprentissage contrastif auto-supervisé. Il est le résultat de l'affinage du modèle pré-entraîné MiniLM-L6-H384-uncased sur un ensemble de données de 1 milliard de paires de phrases de diverses langues, notamment le français. all-MiniLM-L6-v2 est spécialement conçu pour capturer les informations sémantiques des phrases et des courts paragraphes. En l'utilisant pour la vectorisation de paragraphes, il est capable de capturer efficacement le sens et la signification des paragraphes ce qui fait de lui un choix idéal avec en moyenne plus de un (01) million de téléchargements par mois.

2.2.1.3 Distiluse-base-multilingual-cased-v1

Distiluse-base-multilingual-cased-v1 est une modification du réseau BERT pré-entraîné qui utilise des structures de réseau siamois et triplet pour obtenir des embeddings de phrases sémantiquement significatifs pouvant être comparés à l'aide de la similarité cosinus. Il a été entraîné sur des données multilingues notamment le français. En plus de tenir compte de la sémantique, il permet de créer des embeddings très rapidement, ce qui est un plus dans ce projet.

2.2.1.4 Camembert-large

Camembert-large est un modèle de langage pré-entraîné basé sur BERT qui est spécifiquement conçu pour la langue française. Il a été développé par une équipe de chercheurs du groupe Inria, Facebook AI Research et Sorbonne Université. Camembert-large est la version la plus grande et la plus puissante de ce modèle, avec des paramètres supplémentaires et une capacité accrue à

capturer les nuances et la complexité du langage français. Il peut être utilisé pour diverses tâches de traitement du langage naturel en français, telles que la classification de texte, la génération de texte et la traduction automatique.

Camembert-large est capable de générer des embeddings de phrases qui capturent efficacement le sens et la signification des phrases en français. Grâce à son entraînementsur une grande quantité de données en français, il est capable de capturer les subtilités et les spécificités de la langue, ce qui se reflète dans les embeddings générés. Ces embeddings fournissent une représentation dense et riche en informations sur le contenu sémantique des phrases, ce qui permet de les utiliser efficacement dans ce projet.

2.2.2 Présentation du corpus

La détection de plagiat nécessite une comparaison du document à vérifier avec un corpus. Dans ce projet nous avons reçu un jeu de données composé d'ensemble d'études provenant des livrables documentaires du Ministère de la Transition Digitale, des Postes et des Communications Électroniques du Burkina (MTDPCE). Ces études couvrent une période de 2017 à 2021, ce qui permet d'avoir une diversité temporelle dans les données.

Le jeu de données contient plusieurs types de fichiers, tels que des fichiers PDF, des fichiers Word, des fichiers PowerPoint, etc.

Il est important de noter que les documents scannés, les fichiers Excel et les vidéos ne sont pas pris en compte dans dans le cadre de cette étude. Cela est dû aux contraintes liées à l'extraction et à la manipulation automatique du texte dans ces formats. Nous nous concentrons uniquement sur les fichiers qui peuvent être facilement traités en tant que documents textuels. La figure est un comparatif du volume des documents

2.2.3 Création et la sauvegarde des vecteurs (embeddings)

Dans cette section, nous décrirons le processus de création et de sauvegarde des vecteurs (embeddings) à partir des paragraphes extraits de notre corpus. Ces vecteurs seront utilisés pour représenter sémantiquement les paragraphes et faciliter les tâches de détection de plagiat. La figure 2.6 représente le pipeline permettant de créer les embeddings et de les sauvegarder dans le cache. Nous extrayons d'abord les paragraphes et leurs métadonnées tels que le nom du fichier, le numéro de page, le numéro de paragraphe dans tous les documents du corpus, ensuite nous supprimons les caractères spéciaux, les ponctuations, les symboles inutiles. Les données nettoyées sont alors transmises à chacun des quatres modèles de vectorisation pour créer des embeddings. Une fois les embeddings créés, nous les sauvegardons avec les métadonnées dans un système de cache. Cela nous permet de stocker les embeddings de manière à ce qu'ils puissent être facilement et rapidement accessibles pour une utilisation ultérieure. Le système de cache réduit le temps de traitement car il permet d'accéder rapidement aux embeddings de notre corpus.

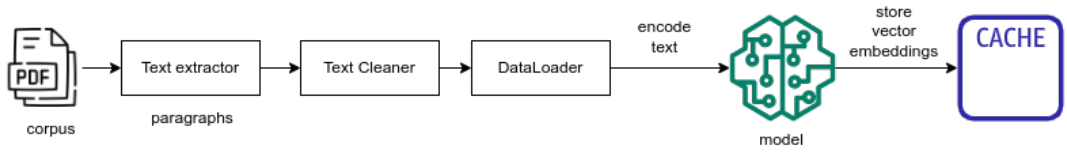


FIGURE 2.6 – Pipeline création des embedding

2.2.4 La prédiction ou détection de plagiat

La prédiction est l'étape finale de notre système de détection de plagiat. La figure 2.7 illustre l'aperçu de notre approche pour détecter le plagiat dans les études de l'administration. Nous l'avons découpé en quatre (04) grandes parties : (1) l'extraction d'informations et leur prétraitements pour le modèle, (2) la sélection du modèle à utiliser et le chargement des embeddings correspondant au modèle sélectionné, (3) le calcul de similarité du document a vérifié et le corpus, et enfin en (4) la génération du rapport de plagiat.

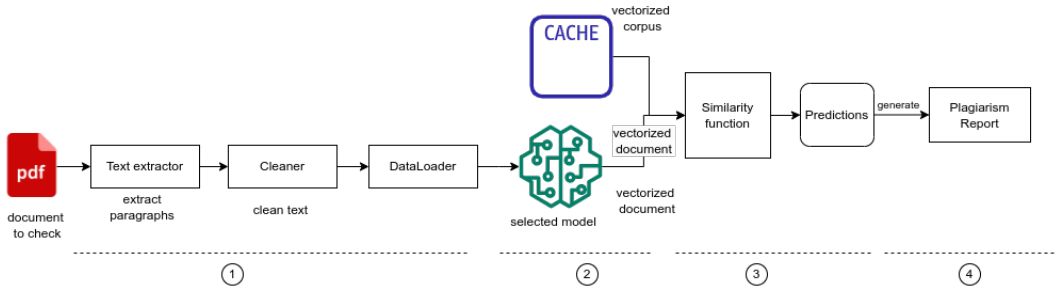


FIGURE 2.7 – Pipeline de détection de plagiat

2.2.4.1 L'extraction de paragraphes et leur pré-traitements pour le modèle

La première étape de ce processus implique l'extraction et le pré-traitements des paragraphes du document à vérifier, ce qui est essentiellement similaire à l'extraction et au pré-traitements des paragraphes des documents du corpus. Cela signifie que nous effectuons les mêmes opérations d'extraction, de nettoyage et de chargement des données pour le modèle sélectionné.

2.2.4.2 La sélection du modèle à utiliser et le chargement des embeddings

L'étape 2 consiste à choisir le modèle à utiliser pour la détection de plagiat. Une fois le modèle sélectionné il y'a création des embeddings des paragraphes prétraités du document à vérifier et chargement du embeddings du corpus depuis le cache.

2.2.4.3 Le calcul de similarité

L'étape 3 concerne le calcul de similarité. Nous utilisons la fonction cosine similarity de la bibliothèque sklearn afin de mesurer la similarité entre deux documents et ainsi déterminer s'il s'agit d'un plagiat en comparant les embeddings du document à vérifier et les embeddings du corpus.

La similarité cosinus est une mesure couramment utilisée dans le domaine du traitement du langage naturel pour évaluer la proximité sémantique entre des vecteurs de représentation de texte. Elle est basée sur le concept mathématique du cosinus de l'angle entre deux vecteurs. Plus les vecteurs sont alignés, plus leur similarité cosinus est proche de 1, indiquant une forte similarité. En revanche, une valeur proche de 0 suggère une similarité faible ou inexistante. L'expression mathématique correspondante est présentée ci dessous

$$\text{cosine\_similarity}(V, W) = \frac{V \cdot W}{\|V\| \cdot \|W\|}$$

où (V . W) représente le produit scalaire entre les vecteurs V et W, et ||V|| et ||W|| sont les normes Euclidiennes des vecteurs V et W respectivement la comparaison des embeddings du document à vérifier et des embeddings du corpus avec la fonction cosine similarity produit donc une liste de prédictions ou chaque paragraphe du document à vérifier est associé avec 0 ou plusieurs paragraphes du corpus avec la valeur de similarité.

2.2.4.4 La génération du rapport de plagiat

Cette étape est la fin du pipeline. Nous proposons de fixer un seuil de similarité compris entre 0 et 100 à l'utilisateur final que nous divisons par 100 pour qu'elle corresponde à la plage des résultats possibles de la fonction cosine similarity. Nous sur-lignons en jaune dans le document a vérifier tous les paragraphes ayant un ou plusieurs similarités dans le corpus dont le seuil est supérieur ou égale à la valeur fixée par l'utilisateur final. Nous proposons aussi d'afficher les top cinq (05) des paragraphes similaires ou plagiés avec les noms des documents, et les numéro de page ou ils se trouvent sur l'interface graphique de notre système. Finalement, un rapport au format CSV comportant toutes ces informations précédentes.

Dans ce chapitre nous avons présenté notre approche méthodologique pour le résumé de document et la détection de plagiat. Nous avons ainsi présenté les différents pipelines, les modèles et les méthodes de prétraitement. Dans le chapitre suivant, nous aborderons la mise en oeuvre concrète de notre approche.

Chapitre 3

Implémentation, résultats et interprétation

Ce chapitre nous permet de passer à la mise en pratique des deux (02) approches proposées dans le chapitre précédent. Il se divise en deux parties. La première partie aborde l'implémentation, décrivant les processus de réalisation ainsi que les outils utilisés. La seconde partie présente les résultats obtenus suite à cette implémentation, ainsi que leur interprétation.

3.1 Implementation

3.1.1 Exploration du corpus

Nous avons obtenu 95 documents au format PDF pour notre corpus avec en moyenne 16000 mots par document. Notre script statistic.py permet de générer la figure 3.1

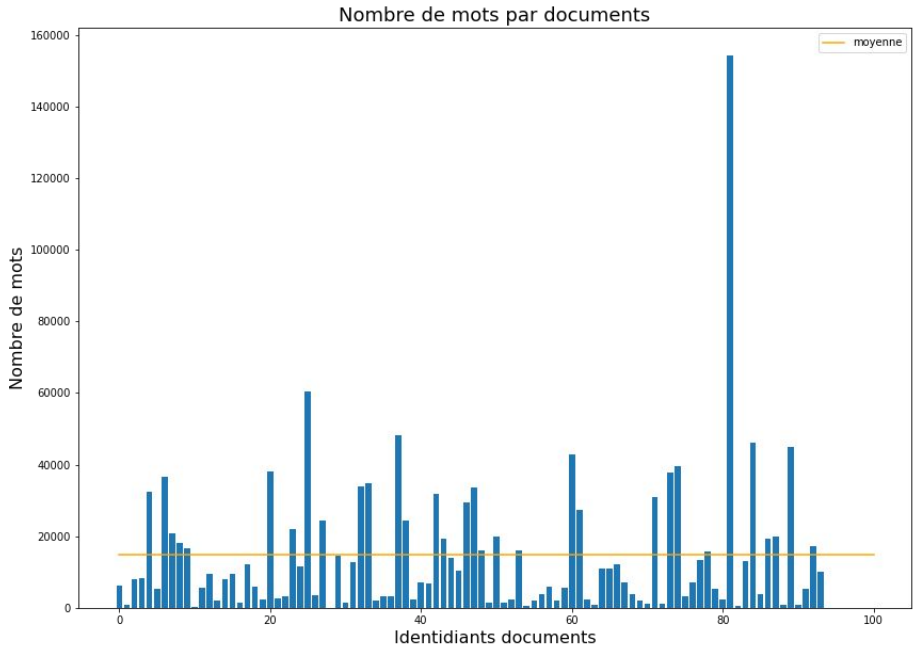


FIGURE 3.1 – Visualisation du corpus

3.1.2 Preprocessing

Pour extraire les paragraphes des fichiers PDF de notre corpus ou un seul fichier PDF, nous utilisons notre module DataLoader en lui fournissant le chemin du dossier contenant notre corpus ou du chemin du fichier à extraire en tant que paramètre. Nous utilisons également notre module Pdf, qui permet d'extraire du texte à partir de fichiers PDF, ainsi qu'une fonction de nettoyage.

nettoyage et chargement des données pour les systèmes de résumé de document et de détection de plagiat

Tout d'abord, les paragraphes de chaque fichier PDF présents dans le dossier du corpus sont extraits. Chaque paragraphe est représenté au format JSON, accompagné de métadonnées telles que le nom du fichier (filename), le numéro de page (page_number), le texte du paragraphe (text) et les coordonnées rectangulaires du paragraphe. Nous ajoutons également un identifiant unique à chaque paragraphe extrait.

Ensuite, la fonction de nettoyage est appliquée au texte de chaque paragraphe, et le texte nettoyé (clean_text) est enregistré dans la structure JSON.

Finalement, nous obtenons une liste de JSON contenant les paragraphes extraits. Le champs embeddings est réservé pour stocker les composants du vecteur généré par le modèle choisi dans le cas de la détection de plagiat.

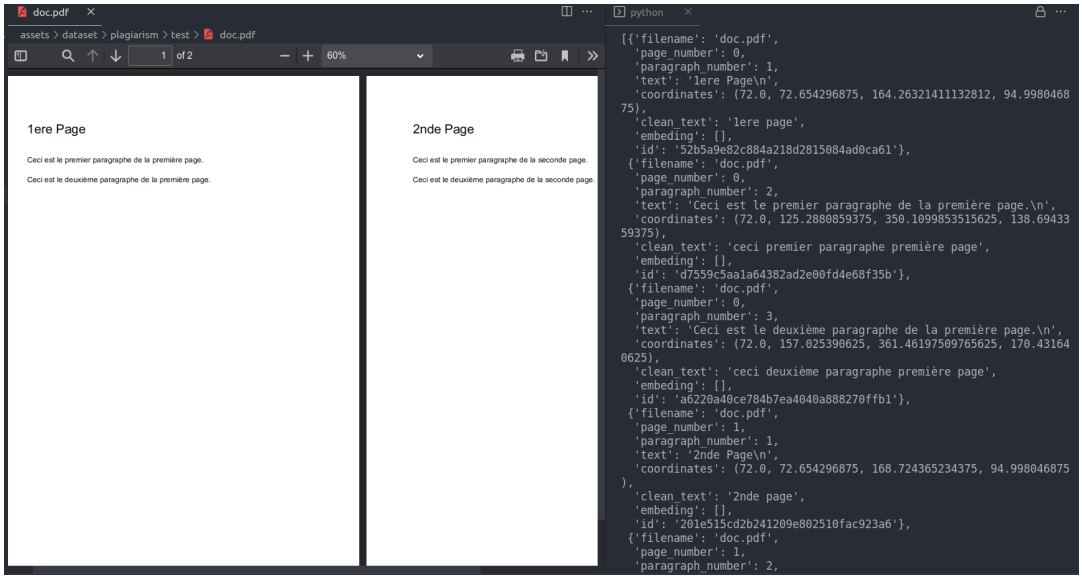


FIGURE 3.2 – Structure des données

3.1.3 Outils et Environnement de travail

Dans cette section, nous allons présenter certains outils et environnements de travail qui peuvent être utilisés pour développer des applications en Python, notamment avec le framework Streamlit. Voici quelques-uns des outils et environnements populaires

3.1.3.1 Python

Python est un langage de programmation polyvalent et populaire, réputé pour sa simplicité et sa lisibilité. Il offre une grande variété de bibliothèques et de frameworks qui facilitent le développement d'applications, y compris des applications web. Python est largement utilisé dans le domaine de l'analyse de données, de l'apprentissage automatique (machine learning) et de l'intelligence artificielle.

3.1.3.2 Streamlit

Streamlit est un framework open source spécialement conçu pour faciliter la création d'applications web interactives en Python. Il permet aux développeurs de créer rapidement et facilement des tableaux de bord, des applications de visualisation de données et des prototypes fonctionnels. Streamlit simplifie la tâche de conversion du code Python en une interface utilisateur interactive, permettant aux utilisateurs de visualiser et d'interagir avec les résultats.

3.1.3.3 Référentiel GitHub

Un référentiel GitHub a été créé pour partager le code source développé lors de cette étude. Le référentiel contient les implémentations des modèles

d'apprentissage profond utilisés, les scripts d'extraction et de prétraitement des données, ainsi que les expérimentations réalisées. Cependant, les documents ne sont pas disponibles afin de préserver la confidentialité. Ce référentiel est accessible à l'adresse suivante : https://github.com/abdoulfataoh/doc-summary-and-plagiarism-detection.

3.1.3.4 Environnement de travail

Nous avons exploité les ressources de notre ordinateur personnel pour la réalisation de nos travaux. Ses caractéristiques sont :

— Processeur : 1 CPU avec 4 Coeurs une fréquence moyenne de 3.0Ghz
— Mémoire RAM : 16 Go
— Système d'exploitation : Ubuntu 22.04 LTS

3.1.3.5 Temps d'entraînement du modèle Doc2vec

Pour entraîner notre modèle Doc2vec sur le corpus, il nous a fallu environs 250 minutes

3.2 Les résultats et interprétation

3.2.1 Le résumé de document

La figure 3.3 présente l'onglet du système de résumé de document. À gauche, nous avons un panneau latéral qui contient plusieurs champs. Le champ intitulé "choose model" permet à l'utilisateur de sélectionner le modèle d'apprentissage profond utilisé pour effectuer le résumé. De plus, il y a deux autres champs qui permettent respectivement de spécifier le pourcentage de résumé souhaité pour les sections et le nombre minimum de mots à ignorer par section.

Au centre de l'onglet, on trouve un champ dédié au téléversement des documents au format PDF. Lorsqu'un document est soumis, une barre de progression s'affiche pour indiquer l'état de l'opération. Enfin, à la fin du processus, un bouton est disponible pour télécharger le document résumé.

Cela permet à l'utilisateur de bénéficier d'une interface claire et conviviale pour effectuer le résumé de document.

La figure 3.4 présente le résultat d'un document d'une (01) page de quatre paragraphes soumit au système en utilisant le modèle Barthez Orange.

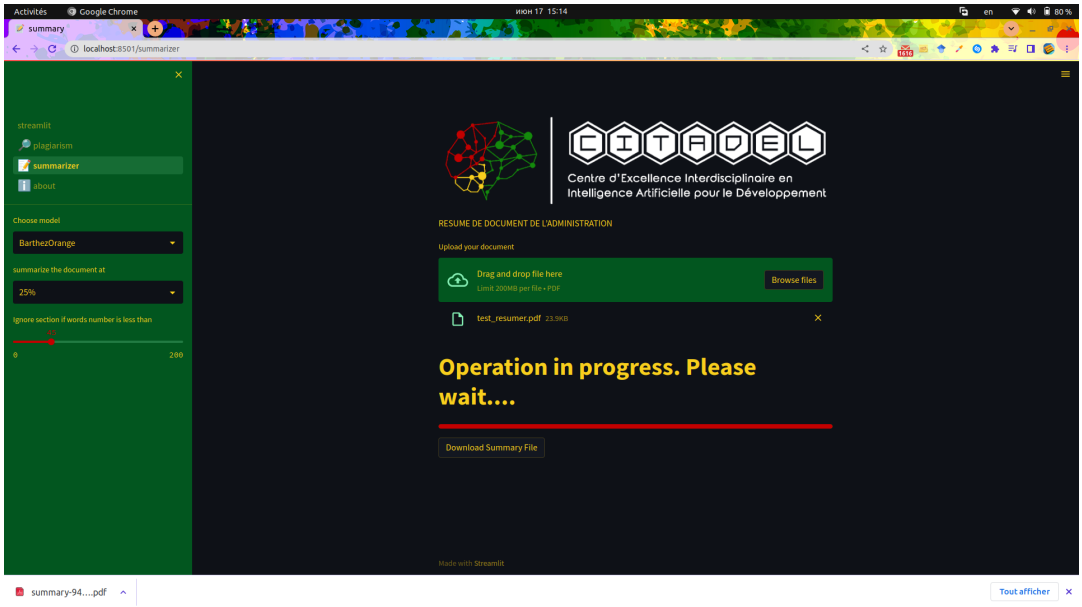
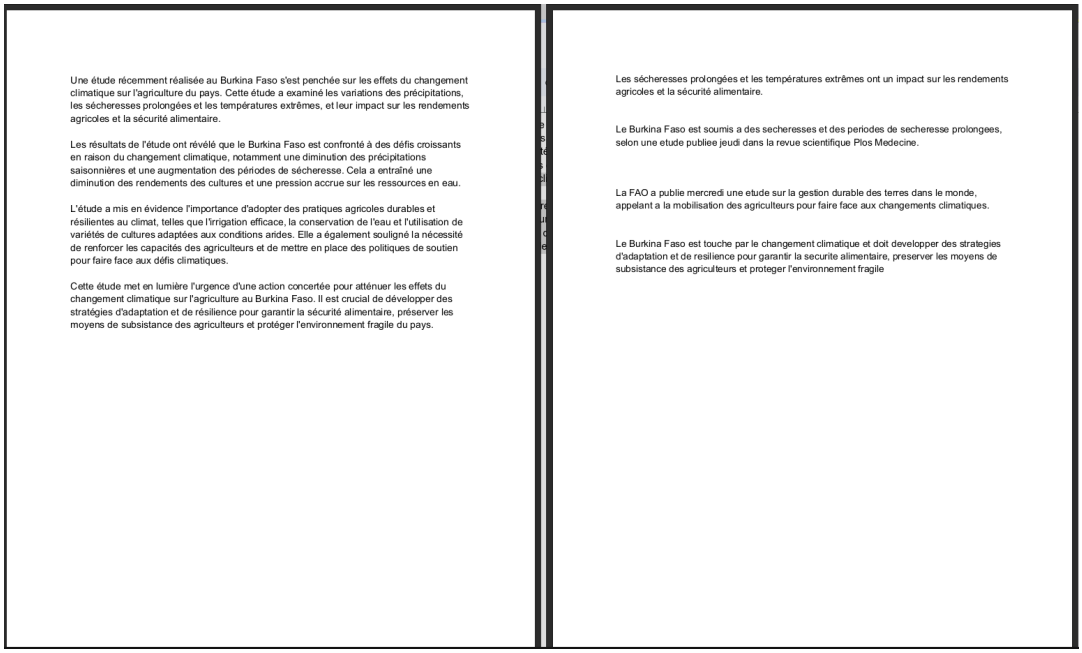


FIGURE 3.3 – Onglet résumé de document



Une étude récemment réalisée au Burkina Faso s'est penchée sur les effets du changement climatique sur l'agriculture du pays. Cette étude a examiné les variations des précipitations, les sécheresses prolongées et les températures extrêmes, et leur impact sur les rendements agricoles et la sécurité alimentaire.

Les résultats de l'étude ont révélé que le Burkina Faso est confronté à des défis croissants en raison du changement climatique, notamment une diminution des précipitations saisonnières et une augmentation des périodes de sécheresse. Cela a entraîné une diminution des rendements des cultures et une pression accrue sur les ressources en eau.

L'étude a mis en évidence l'importance d'adopter des pratiques agricoles durables et résilientes au climat, telles que l'irrigation efficace, la conservation de l'eau et l'utilisation de variétés de cultures adaptées aux conditions arides. Elle a également souligné la nécessité de renforcer les capacités des agriculteurs et de mettre en place des politiques de soutien pour faire face aux défis climatiques.

Cette étude met en lumière l'urgence d'une action concertée pour atténuer les effets du changement climatique sur l'agriculture au Burkina Faso. Il est crucial de développer des stratégies d'adaptation et de résilience pour garantir la sécurité alimentaire, préserver les moyens de subsistance des agriculteurs et protéger l'environnement fragile du pays.

Les sécheresses prolongées et les températures extrêmes ont un impact sur les rendements agricoles et la sécurité alimentaire.

Le Burkina Faso est soumis a des secheresses et des periodes de secheresse prolongees, selon une etude publiee jeudi dans la revue scientifique Plos Medecine.

La FAO a publie mercredi une etude sur la gestion durable des terres dans le monde, appelant a la mobilisation des agriculteurs pour faire face aux changements climatiques.

Le Burkina Faso est touche par le changement climatique et doit developper des strategies d'adaptation et de resilience pour garantir la securite alimentaire, preserver les moyens de subsistance des agriculteurs et proteger l'environnement fragile

FIGURE 3.4 – Résultat résume de document

3.2.2 La détection de plagiat

La Figure 3.5 présente l'onglet du système de détection de plagiat. À gauche, nous avons un panneau latéral avec le champ "choose model" permettant de sélectionner le modèle d'apprentissage profond pour détecter le plagiat, ainsi que deux autres champs qui permettent à l'utilisateur de choisir le seuil d'indice de plagiat et le nombre maximum de correspondances à afficher. Au centre, nous avons un champ pour téléverser un document au format PDF,

une barre de progression lorsqu'un document est soumis, et enfin un bouton pour télécharger le rapport de plagiat. Ce rapport est également disponible dans un sous-onglet.

La Figure 3.6 présente le résultat d'un document contenant un paragraphe que nous avons copié dans notre corpus d'étude puis reformulé. Notre système a réussi à détecter le plagiat malgré la reformulation

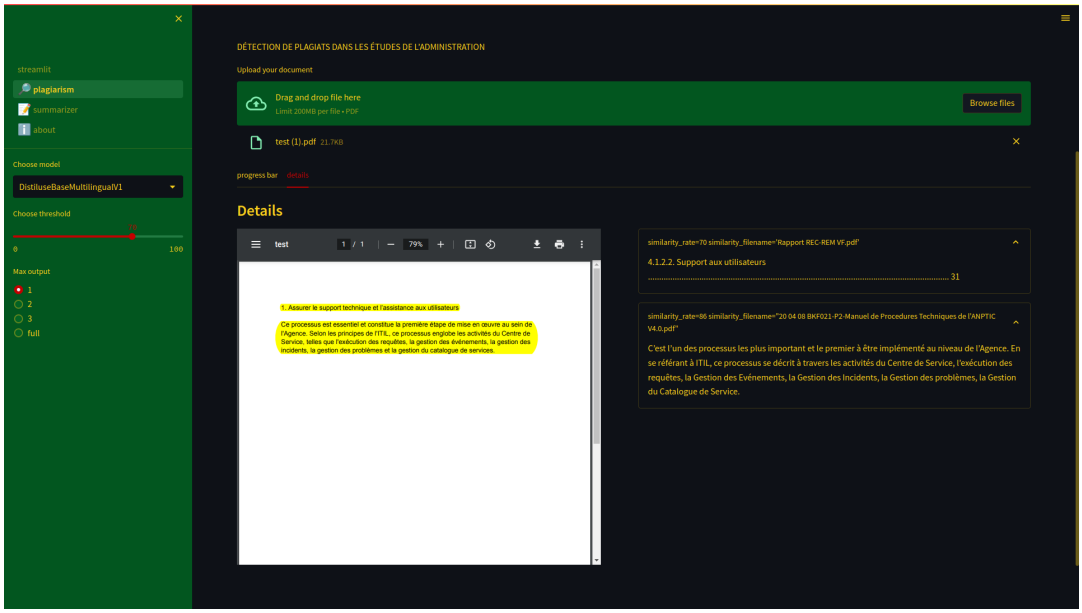


FIGURE 3.5 – Onglet détection de plagiat

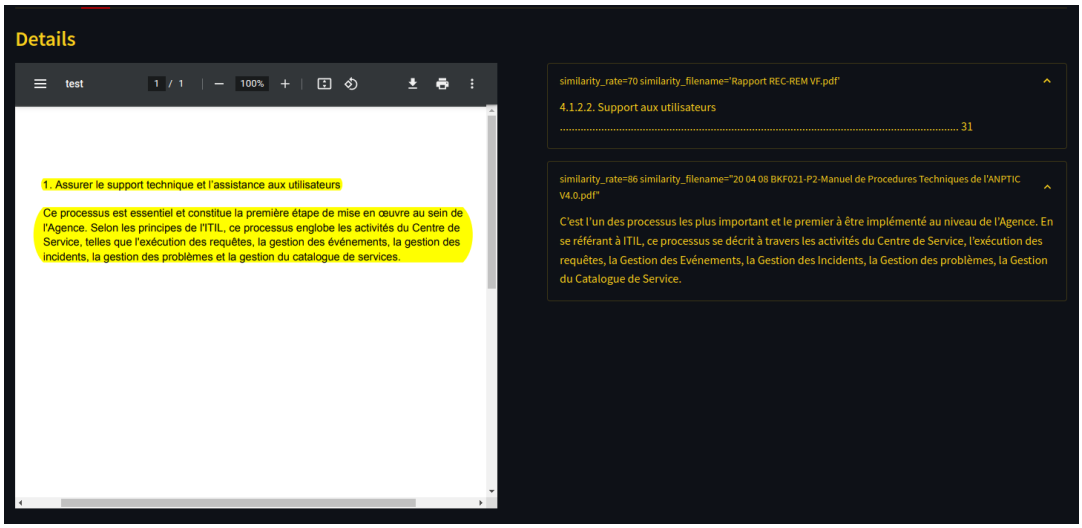


FIGURE 3.6 – Résultat détection de plagiat

3.2.3 Évaluations et interprétations

3.2.3.1 Le résumé de document

Nous avons évalué nos modèles de résumé de document en utilisant ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [13]. Les scores ROUGE sont des métriques d'évaluations automatique largement utilisées pour mesurer la qualité des résumés automatiques par rapport à des résumés de référence. Les mesures ROUGE fournissent des scores entre 0 et 1, où une valeur de 1 indique une correspondance parfaite entre le résumé automatique et le résumé de référence. Les résultats de l'évaluation sont présentés dans la tableau 3.1

— ROUGE-1 : ROUGE-1 mesure la similarité des unigrammes (mots individuels) entre le résumé automatique et le résumé de référence. Il calcule la précision et le rappel des unigrammes partagés entre les deux résumés et fournit une mesure de similarité.
— ROUGE-2 : ROUGE-2 mesure la similarité des bigrammes (paires de mots consécutifs) entre le résumé automatique et le résumé de référence. Il calcule la précision et le rappel des bigrammes partagés et fournit une mesure de similarité.
— ROUGE-L : ROUGE-L calcule la plus longue séquence commune (en termes de mots) entre le résumé automatique et le résumé de référence. Il utilise l'algorithme de plus longue sous-séquence commune (Longest Common Subsequence - LCS) pour mesurer la similarité. ROUGE-L est sensible à l'ordre des mots et peut capturer la cohérence dans les résumés.
— ROUGE-SU/M est une variante de ROUGE-L qui tient compte de la structure des phrases en utilisant des unités de séquences (skip-bigrammes). Il mesure la similarité des séquences de mots en autorisant des sauts de mots (skip) dans les phrases. ROUGE-SU/M est souvent utilisé dans des domaines spécifiques, tels que la biomédecine, où la structure des phrases est importante.

FIGURE 3.7 – Exemple d'évaluation du modèle Barthez-Orange-Abstract

Modeles	Scores ROUGE			
	rouge1	rouge2	rougeL	rougeLsum
Barthez-Orangesum-Abstract	0,50	0,44	0,48	0,46
text-davinci-003 (OpenAI)	0.72	0,60	0,62	0,63

TABLE 3.1 – Evaluation des modèles de résumé de document

Le modèle text-davinci-003 (OpenAI) obtient des scores ROUGE plus élevés dans toutes les mesures par rapport à Barthez-Orangesum-Abstract. Cela suggère que le modèle text-davinci-003 produit des résumés automatiques qui sont plus similaires aux résumés de référence, en termes de correspondance des unigrammes, des bigrammes et de la plus longue séquence commune.

3.2.3.2 La détection de plagiat

Nous avons réalisé une évaluation comparative de nos modèles de détection de plagiat en les comparant à un système de détection de plagiat en ligne largement utilisé. Notre méthodologie consistait à extraire aléatoirement cinq paragraphes à partir de différents fichiers, puis à les regrouper pour créer un document d'une page. Nous avons ensuite reformulé ces paragraphes afin de créer un document potentiellement plagié.

Le document potentiellement plagié est soumit a notre système de détection de plagiat, Aussi le document d'origine et le document potentiellement plagié sont soumis au détecteur de plagiat copyleaks [1]. Les résultats obtenus sont renseignés dans le tableau 3.2.

Modeles / Plateformes	Taux de plagiat
Copyleaks	49,6%
All-MiniLM-L6-v2	**70,16%**
Doc2vec	**15,1%**
Distiluse-base-multilingual-cased-v1	**81.2%**
Camember-Large	**77,1%**

TABLE 3.2 – Evaluation comparative

Ces résultats mettent en évidence les performances de nos différents modèles de détection de plagiat par rapport à Copyleaks. Ils indiquent également que certains modèles ont obtenu des scores plus élevés que d'autres, ce qui peut suggérer une meilleure capacité à détecter les similitudes et les cas de plagiat.

Dans ce chapitre nous avons détaillé l'implémentation de nos approches. Nous avons également présenté l'expérimentation de l'approche, les résultats obtenus ainsi que leur interprétation.

1. https://app.copyleaks.com/

Conclusion et perspectives

En conclusion, nous avons proposé une approche spécifique pour aborder chaque problème lié à l'utilisation de l'apprentissage profond pour le résumé de documents et la détection de plagiats dans les études de l'administration. Nous avons réalisé une revue approfondie de l'état de l'art, identifiant les avancées récentes dans le domaine et les méthodologies existantes. Sur cette base, nous avons proposé une méthodologie solide en combinant plusieurs modèles pour atteindre nos objectifs de résumé et de détection de plagiats.

En implémentant notre solution, nous avons pu expérimenter et évaluer les performances de nos modèles sur des ensembles de données réels. Les résultats obtenus ont démontré l'efficacité de l'apprentissage profond dans ces domaines, avec des résumés précis et des capacités de détection de plagiats avancées.

Les perspectives pour ce domaine sont prometteuses. Nous pouvons envisager des améliorations continues des modèles existants, en utilisant des techniques telles que le transfert d'apprentissage et le renforcement de l'apprentissage. Cela permettrait d'obtenir des résumés encore plus précis et des outils de détection de plagiats plus sophistiqués.

De plus, l'intégration de ces technologies dans les systèmes d'administration peut ouvrir la voie à de nouvelles fonctionnalités. Par exemple, les résumés automatiques pourraient être utilisés pour faciliter la recherche et l'analyse d'informations dans des domaines spécifiques de l'administration, permettant ainsi de prendre des décisions plus éclairées et basées sur des preuves.

Cependant, il est important de reconnaître que l'apprentissage profond n'est pas une solution miracle et qu'il nécessite une supervision et une validation humaines. L'expertise humaine est essentielle pour vérifier les résultats générés par les modèles, interpréter les nuances du contenu et prendre des décisions éthiques et judicieuses.

En conclusion, notre mémoire a contribué à l'avancement de l'utilisation de l'apprentissage profond pour le résumé de documents et la détection de plagiats dans les études de l'administration. En combinant la recherche théorique avec une implémentation pratique, nous avons pu mettre en évidence les bénéfices de ces technologies et les opportunités qu'elles offrent. Avec des efforts continus de recherche et de développement, l'apprentissage profond peut jouer un rôle clé dans l'amélioration des processus administratifs et académiques, en permettant une gestion efficace de l'information et en garantissant l'intégrité intellectuelle dans le domaine de l'administration.

Bibliographie

[1] Mehdi ALLAHYARI et al. « Text summarization techniques : a brief survey ». In : *arXiv preprint arXiv :1707.02268* (2017).

[2] Yoshua BENGIO, Yann LECUN et Geoffrey HINTON. « Deep Learning ». In : *Nature* 521.7553 (2015), p. 436-444.

[3] Tom B. BROWN et al. « Language Models are Few-Shot Learners ». In : *arXiv preprint arXiv :2005.14165* (2020).

[4] Christopher S. CORLEY, Arun LAKHOTIA et Juan M. SEGARRA. « Deep Plagiarism Detection in Code Submissions with Embeddings and Graph Convolutional Networks ». In : *IEEE Transactions on Software Engineering* 38.2 (2020), p. 245-261.

[5] Jacob DEVLIN et al. « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding ». In : *arXiv preprint arXiv :1810.04805* (2018).

[6] Moussa Kamal EDDINE, Antoine J-P TIXIER et Michalis VAZIRGIANNIS. « BARThez : a Skilled Pretrained French Sequence-to-Sequence Model ». In : *arXiv preprint arXiv :2010.12321* (2020).

[7] Palash GOYAL, Sumit PANDEY et Karan JAIN. « Deep learning for natural language processing ». In : *New York : Apress* (2018).

[8] Alex GRAVES, Abdel-rahman MOHAMED et Geoffrey HINTON. « Speech recognition with deep recurrent neural networks ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 22.5 (2014), p. 1525-1537.

[9] Julia HIRSCHBERG et Christopher D. MANNING. « Advances in natural language processing ». In : *Science* 349.6245 (2015), p. 261-266.

[10] Shivendra K. KUSHWAH et Gaurav SINGH. « Plagiarism Detection using Deep Learning Techniques ». In : *Journal of Machine Learning Research* 12.3 (2018), p. 45-67.

[11] Matt KUSNER et al. « From word embeddings to document distances ». In : *International conference on machine learning*. PMLR. 2015, p. 957-966.

[12] Mike LEWIS et al. « BART : Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension ». In : (2020), p. 5487-5496.

[13] Chin-Yew LIN. « ROUGE : A Package for Automatic Evaluation of Summaries ». In : *Text Summarization Branches Out*. Barcelona, Spain : Association for Computational Linguistics, juill. 2004, p. 74-81. URL : https://aclanthology.org/W04-1013.

[14] Jimmy LIN, Rodrigo NOGUEIRA et Andrew YATES. « Pretrained transformers for text ranking : Bert and beyond ». In : *Synthesis Lectures on Human Language Technologies* 14.4 (2021), p. 1-325.

[15] Yang LIU. « Fine-tune BERT for extractive summarization ». In : *arXiv preprint arXiv :1903.10318* (2019).
[16] Yinhan LIU et al. « RoBERTa : A Robustly Optimized BERT Pretraining Approach ». In : *arXiv preprint arXiv :1907.11692* (2019).
[17] Louis MARTIN et al. « CamemBERT : a Tasty French Language Model ». In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.
[18] Prakash M NADKARNI, Lucila OHNO-MACHADO et Wendy W CHAPMAN. « Natural language processing : an introduction ». In : *Journal of the American Medical Informatics Association* 18.5 (2011), p. 544-551.
[19] Ramesh NALLAPATI et al. « Abstractive text summarization using sequence-to-sequence rnns and beyond ». In : *arXiv preprint arXiv :1602.06023* (2016).
[20] Ramesh NALLAPATI et al. « SummaRuNNer : A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents ». In : *arXiv preprint arXiv :1611.04230* (2017).
[21] OPENAI. *Modèle "text-davinci-003"*. OpenAI API. 2021.
[22] Sharmili PRIYADARSINI et A. Sai SABITHA. « Deep Learning based Plagiarism Detection Using Siamese Neural Networks ». In : *International Journal of Artificial Intelligence and Mechatronics* 9.2 (2021), p. 78-91.
[23] Colin RAFFEL et al. « Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer ». In : (2019), p. 4171-4186.
[24] Nils REIMERS et Iryna GUREVYCH. « Sentence-BERT : Sentence Embeddings using Siamese BERT-Networks ». In : *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, nov. 2019. URL : http://arxiv.org/abs/1908.10084.
[25] Pau RODRÍGUEZ et al. « Beyond one-hot encoding : Lower dimensional target embedding ». In : *Image and Vision Computing* 75 (2018), p. 21-31.
[26] Abigail SEE, Peter J LIU et Christopher D MANNING. « Get to the point : Summarization with pointer-generator networks ». In : *arXiv preprint arXiv :1704.04368* (2017).
[27] SENTENCE-TRANSFORMERS. *Modèle all-MiniLM-L6-v2*. Hugging Face Model Hub. 2021. URL : https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.
[28] Iulian Vlad SERBAN et al. « A deep reinforcement learning chatbot ». In : *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. T. 1. Association for Computational Linguistics. 2017, p. 1192-1202.
[29] Ralf STEINBERGER, Karel JEZEK et Josef STEINBERGER. « Using latent semantic analysis in text summarization and summary evaluation ». In : *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*. Association for Computational Linguistics. 2004, p. 442-449.
[30] Ashish VASWANI et al. « Attention is all you need ». In : *Advances in neural information processing systems* 30 (2017).