UNIVERSITE JOSEPH KI-ZERBO

(UJKZ)

****

INSTITUT BURKINABE DES ARTS ET METIERS

(IBAM)

MEMOIRE POUR L'OBTENTION DU MASTER II

Option : **Ingénierie des Systèmes d'information en Entreprise (ISIE)**

THEME :

Système de détection de plagiat basé sur des algorithmes d'Intelligence Artificielle (IA)

Réalisé par : Wendinmi Rodrigue OUEDRAOGO

Directeur de mémoire :

Docteur Yaya TRAORE

Enseignant-chercheur en informatique à l'IBAM

Maître de stage :

Madame Zoulfaou DANKOURMA

Présidente de GOBELAB

Année universitaire 2020 – 2021

RÉSUMÉ

Authentifier l'originalité des travaux scientifiques, détecter les cas de plagiat sont des activités qui sont d'une extrême importance dans le monde universitaire et scientifique. Toutefois, le phénomène du plagiat prend beaucoup d'ampleur ces dernières années et cela va de pair avec la croissance d'Internet. Ainsi, il convient de créer continuellement des outils performants pour aider à la détection de cette forme de fraude. C'est pour cela, que dans notre travail, nous proposons un système de détection de plagiat basé sur des algorithmes d'Intelligence Artificielle (IA).

Dans une première partie, nous faisons ressortir une étude des algorithmes et travaux existants en lien avec la détection du plagiat et l'IA. Puis dans une seconde partie nous proposons une méthode de détection de plagiat utilisant des méthodes de TAL (Traitement automatique du langage naturel) avec notamment la variante SBERT (Sentence-BERT) de l'algorithme BERT (Bidirectional Encoder Representations from Transformers). Et enfin nous proposons une plateforme logicielle exploitant notre méthode.

Mots clefs : *Intelligence Artificielle, Traitement automatique du langage naturel, Plagiat, Similarité.*

ABSTRACT

Authenticating the originality of scientific works and detecting plagiarism are activities that are of extreme importance in the academic and scientific world. However, the phenomenon of plagiarism has been growing in recent years and this goes hand in hand with the growth of the Internet. Thus, it is necessary to continuously create efficient tools to help detect this form of fraud. That is why, in our work, we propose a plagiarism detection system based on Artificial Intelligence (AI) algorithms.

In a first part, we highlight a study of existing algorithms and works related to the detection of plagiarism and AI. Then, in a second part, we propose a plagiarism detection method using NLP (Automatic Natural Language Processing) methods with the SBERT (Sentence-BERT) variant of the BERT (Bidirectional Encoder Representations from Transformers) algorithm. Finally, we propose a software platform using our method.

Keywords: Artificial Intelligence, Natural Language Processing, Plagiarism, Similarity.

DEDICACE

Au Bon Dieu

Et à toute

ma famille

REMERCIEMENTS

Afin de mener à bien notre travail nous avions pu compter sur l'inestimable soutien et suivi de personnes formidables et spéciales. Nous leur traduisons, par ces écrits, notre sincère gratitude.

Nos remerciements vont particulièrement à l'endroit de :

- Dr Yaya TRAORE, notre Directeur de mémoire, qui nous a assistés, conseillés et guidés tout au long de notre travail.
- Mme Zoulfaou DANKOURMA, notre maître de stage, qui également, n'a ménagé aucun effort pour nous accompagner dans notre travail.
- Monsieur le Directeur de l'IBAM, le corps professoral et tout le personnel de l'IBAM pour leur accompagnement.
- Nos associés et collègues de l'entreprise ATIS (Africa Technologies & Innovation Space), qui nous ont toujours soutenus durant nos travaux.
- Tous nos amis, connaissances, camarades et toute autre personne dont le soutien fut présent directement ou indirectement.
- Notre famille, qui animé d'un soutien grandissime depuis toujours, nous soutient constamment dans nos études comme ce fut le cas durant ce travail.

SOMMAIRE

RÉSUMÉ i

ABSTRACT ii

DEDICACE iii

REMERCIEMENTS iv

SOMMAIRE v

LISTE DES SIGLES ET ABREVIATIONS vii

LISTE DES FIGURES GRAPHIQUES viii

LISTE DES TABLEAUX ix

INTRODUCTION GENERALE 1

Chapitre 1: ÉTAT DE L'ART 3

I. Concepts généraux 3

II. Outils existants 7

III. Algorithmes utilisés en NLP 8

IV. Travaux dans le domaine 13

V. Choix algorithmique 13

Chapitre 2: APPROCHE DE DÉTECTION DE PLAGIAT 15

I. Approche proposée 15

II. Analyse et conception de la plateforme 17

Chapitre 3: IMPLÉMENTATION DE LA PLATEFORME 25

I. Outils utilisés 25

II. Tests et résultats 27

III. Quelques maquettes de la plateforme 28

CONCLUSION 30

BIBLIOGRAPHIE 31

TABLE DES MATIERES 35

LISTE DES SIGLES ET ABREVIATIONS

SIGLE / ABREVIATION	SIGNIFICATION
IBAM	L'Institut Burkinabè des Arts et Métiers
CU	Cas d'Utilisation
ORM	Mapping Objet-Relationnel (en anglais Object-Relational Mapping)
BD	Base de données
UML	Unified Modeling language
IHM	Interactions Homme-Machines
CREAD	Create, Read, Update, Delete
HTTP	HyperText Transfert Protocol
MySQL	My Structural Query Language
PHP	Hypertext Processor
SQL	Structure Query Language
SGBD	Système de Gestion de Base de Données
RCCM	Registre du Commerce et du Crédit Mobilier
BERT	Bidirectional Encoder Representations from Transformers
SBERT	Sentence-BERT
IA	Intelligence Artificielle
TAL	Traitement Automatique du Langage Naturel
NLP	Natural Language Processing (ou TAL)
DL	Deep Learning (Apprentissage profond)
ML	Machine Learning (Apprentissage automatique)

LISTE DES FIGURES GRAPHIQUES

Figure 1: Image illustrative du test de Turing 6

Figure 2:Diagramme du processus de détection de plagiat dans la base de données locale 16

Figure 3:Diagramme de cas d'utilisation 19

Figure 4:Diagramme de classes 22

Figure 5:Diagramme de déploiement 23

Figure 6:Données de tests 27

Figure 7: Fonction principale d'analyse de similarité avec SBERT 27

Figure 8: Résultat de l'analyse avec les données de test 28

Figure 9:Page d'accueil de l'espace de travail 28

Figure 10:Page de création d'un nouveau dossier 28

Figure 11:Page d'accueil de dossier et de visualisation de résultats 29

LISTE DES TABLEAUX

Tableau 1: Dictionnaire de données 22

INTRODUCTION GENERALE

« Rendre la machine au moins aussi intelligente que l'homme », tel semble être le but de l'Homme dans ses recherches en Intelligence Artificielle (IA). Ainsi, plusieurs champs d'action ont connu le jour au sein des champs de recherches en IA. Nous avons entre autres le Machine Learning (ML), le Deep Learning (DL), Le Traitement Automatique du Langage (TAL ou NLP), etc. Chacun de ces domaines a connu de grands progrès. En l'occurrence le TAL qui a connu ces dernières années des avancées considérables. Ce qui démontre en partie de sa grande importance dans les progrès de nos sociétés. En effet, les champs d'application du TAL sont très variés. Ceci concerne notamment la traduction automatique, la reconnaissance vocale, l'analyse des sentiments, la détection de similarité, etc. Dans notre étude, nous aborderons le champ d'action du TAL qu'est l'étude des similarités textuelles. L'analyse des similarités en IA, est un domaine qui a connu de belles avancées. Beaucoup de méthodes algorithmiques sont proposées afin de permettre de faire de l'analyse de similarité entre des jeux de données et pouvant servir par exemple pour la détection de plagiat tel que nous le ferons dans nos travaux.

- **Contexte**

Le magazine l'Express intitulait l'un de ses articles en 2015 « Le plagiat, un "phénomène exponentiel" dans les universités françaises »[1], pour traiter en effet d'un sujet qui prend de l'ampleur dans les universités française, mais qui démontre la réalité dans toutes les universités du monde avec bien sûr des degrés différents. Le plagiat est un problème combattu depuis des années par les acteurs universitaires. Aussi, dans nos universités, beaucoup d'ouvrages ne sont pas en ligne, mais dans des bases de données locales. De ce fait, tout système de détection de plagiat devrait être adapté à des bases de données locales en plus des ressources déjà disponibles sur le web. En effet, pour le cas des rapports de soutenance de niveau licence ou master par exemple, qui ne sont généralement pas publiés en ligne, il y a souvent des cas de copier-coller qui passent inaperçus aux responsables des universités. Toutefois, de nos jours, il existe beaucoup d'outils qui permettent de détecter les similitudes dans les productions universitaires et ainsi de conclure souvent à des plagiats. De plus, ces outils sont pour la plupart propriétaires et/ou limités du point de vue de la détection poussé de plagiat.

[1] https://www.lexpress.fr/education/le-plagiat-un-phenomene-exponentiel-dans-les-universites-francaises_2096229.htmlhttps://www.lexpress.fr/education/le-plagiat-un-phenomene-exponentiel-dans-les-universites-francaises_2096229.html

❖ Problématique

Nous constatons donc que les outils de détection de plagiat devraient adaptations face à d'un environnement donné et évolutif. De ce qui découle de notre analyse précédente, il est donc question pour nous de proposer une méthode algorithmique utilisant des techniques avancées d'IA pour permettre la détection de plagiat et adaptable à un environnement donné. Toutefois, avec les avancés considérables en NLP, nous pourrons donc trouver des voies et moyens pour résoudre cette problématique.

❖ Objectifs

Nos objectifs à l'issue de notre travail sont de faire un état de l'art sur les méthodes algorithmiques en IA qui permettent la détection de plagiat, proposer une méthode de réalisation et enfin réaliser une plateforme de détection de plagiat.

❖ Organisations du document

Afin de présenter notre travail, notre document sera structuré avec 3 chapitres principaux. Premièrement, nous montrerons un chapitre qui fait une étude documentaire sur les méthodes algorithmiques en IA qui permettent la détection de plagiat. Deuxièmement, nous présenterons notre méthode de réalisation. Puis dernièrement, nous montrerons nos travaux d'implémentation de notre méthode avant de conclure.

CHAPITRE 1: ÉTAT DE L'ART

I. Concepts généraux

1) Le plagiat

a) Définition

Selon l'Université Angers, « C'est le fait de s'approprier les idées ou les mots de quelqu'un d'autre en les faisant passer pour les siens. Il peut être volontaire ou involontaire. » [21] . En effet, nous pouvons dire que le plagiat, c'est de la triche intellectuelle, c'est de la piraterie d'idées, c'est s'approprier le travail intellectuel durement abattu par quelqu'un. C'est une manière souvent subtile qui consiste à s'approprier la production intellectuelle d'une autre personne ou entité.

b) Les types de plagiats

Il existe plusieurs types de plagiat. Toutefois, nous allons citer 5 grands types de plagiats [18] :

- **Le plagiat direct :** le plagiat direct consiste à recopier le travail d'un auteur dans son travail sans en faire mention de ce dernier. Faisans ainsi croire qu'on en est l'auteur.
- **L'auto-plagiat :** l'auto plagiat est une forme de plagiat qui consiste à recopier, dans un nouvel ouvrage, partiellement et intégralement un travail que l'on a soit même déjà réaliser auparavant.
- **Le plagiat de citation :** cela consiste à paraphraser sans citer la source. Tout auteur de paraphrase doit être mentionné.
- **Le plagiat « copier-coller » :** lorsque vous formez un nouveau texte à partir de multiples paraphrases, cela constitue un plagiat « copier-coller ».
- **L'achat de travail intellectuel :** l'achat de production intellectuel pour faire sienne est également un plagiat, car ce travail n'est pas la production intellectuelle de l'auteur.

Ainsi, il existe plusieurs types de plagiats, qui sont certainement faits pour plusieurs raisons.

c) Les causes

Il existe plusieurs raisons qui peuvent pousser à s'approprier un travail intellectuel, dont nous en sommes pas l'auteur. Nous pouvons ainsi citer :

- L'ignorance : certaines personnes ignorent tout simple les bonnes pratiques en matière de rédaction documentaire scientifique ou pas ;
- La paresse ou l'incompétence face au travail à réaliser ;
- Un travail mal organisé et/ou mal planifié. C'est souvent le cas de travaux de dernière minute ;

Toutefois, quelle que soit la cause, les conséquences peuvent souvent être dures.

d) Les conséquences

Une personne dont le travail est jugé plagié peut être passible de plusieurs sanctions.

- L'avertissement, l'ajournement, la suspension temporelle ou totale dans le cas d'un travail universitaire.
- Une réputation salie : en effet s'approprier le travail de quelqu'un n'honore pas le fautif et peut nuire gravement à sa réputation.

e) Prévention

Il existe plusieurs moyens de se prémunir du plagiat. Nous pouvons citer le fait de :

- Citez systématiquement toutes les sources que nous utilisons
- Savoir bien organiser et bien planifier tout travail intellectuel que nous entamons
- Bien se documenter en ce qui concerne les bonnes pratiques en matière d'utilisation de travaux existants
- Pensez toujours à l'éthique
- Toujours lire les licences d'utilisation qui accompagne les œuvres intellectuelles.

f) Protéger son œuvre

Il existe plusieurs moyens de se protéger contre le plagiat de ses travaux. Il vous faut en somme vous documenter sur les licences d'utilisations et les lois relatives à la protection de la propriété intellectuelle.

Néanmoins pour les travaux académiques et de recherches, avant toute publication s'assurer d'avoir imposé une licence à votre œuvre. Des licences couramment utilisées sont celles de l'organisation Creative Commons[2]. Vous pouvez en utiliser sur vos œuvres.

2) L'Intelligence Artificielle (IA)

Pour John McCarthy [1] , il s'agit de la science et de l'ingénierie de la fabrication de machines intelligentes, en particulier de programmes informatiques. En effet, les recherches en IA visent à créer des algorithmes qui vont simuler l'intelligence humaine. Ce qui fait donc de grands challenges à relever d'autant plus que l'intelligence humaine est une notion elle-même difficile à définir et à appréhender dans son ensemble. Néanmoins le peu de découverte que nous savons sur le fonctionnement et la structure du cerveau humain nous a permis de faire d'énormes progrès même si le champ de recherche semble infini.

Nous pouvons dire que l'IA est la branche de l'informatique qui vise à donner de l'intelligence à la machine. Étant donné le champ élargi des recherches, le secteur de l'IA se divise en plusieurs branches. On peut donc citer entre autres les domaines tels que le raisonnement automatisé, l'apprentissage automatique ou Machine Learning (ML), le traitement automatique des langues (TAL) ou Natural Language processing (NLP), la perception, la robotique, les systèmes autonomes, etc.

3) Natural Language processing (NLP)

La compréhension du langage humain par la machine fait l'objet de beaucoup de travaux depuis les années 50 [16]. Tout d'abord, avec le test de Turing (Figure 1) proposé par Alan Turing dans les années 1950, un humain devrait converser avec des utilisateurs anonymes et devrait faire la distinction entre les utilisateurs humains et ceux machines. Ne pas arriver à faire de différence devrait prouver, selon la théorie d'Alan Turing, que la machine a atteint un certain niveau d'intelligence. Ainsi, depuis lors, le test de Turing est devenu comme un standard pour les tests d'IA. Et lorsqu'on parle de test de Turing, on parle déjà de la compréhension du langage humain par la machine.

Le NLP est une branche de l'informatique qui vise à permettre à la machine de comprendre les données relatives au langage humain (langage écrit, audio, etc.). Ceci permet ainsi de nos jours plusieurs possibilités à savoir la complétion automatique, la traduction automatique, la

[2] https://creativecommons.org/licenses/?lang=fr

génération de texte, le résumé automatique, l'analyse des sentiments, l'analyse des similarités textuelles, etc.

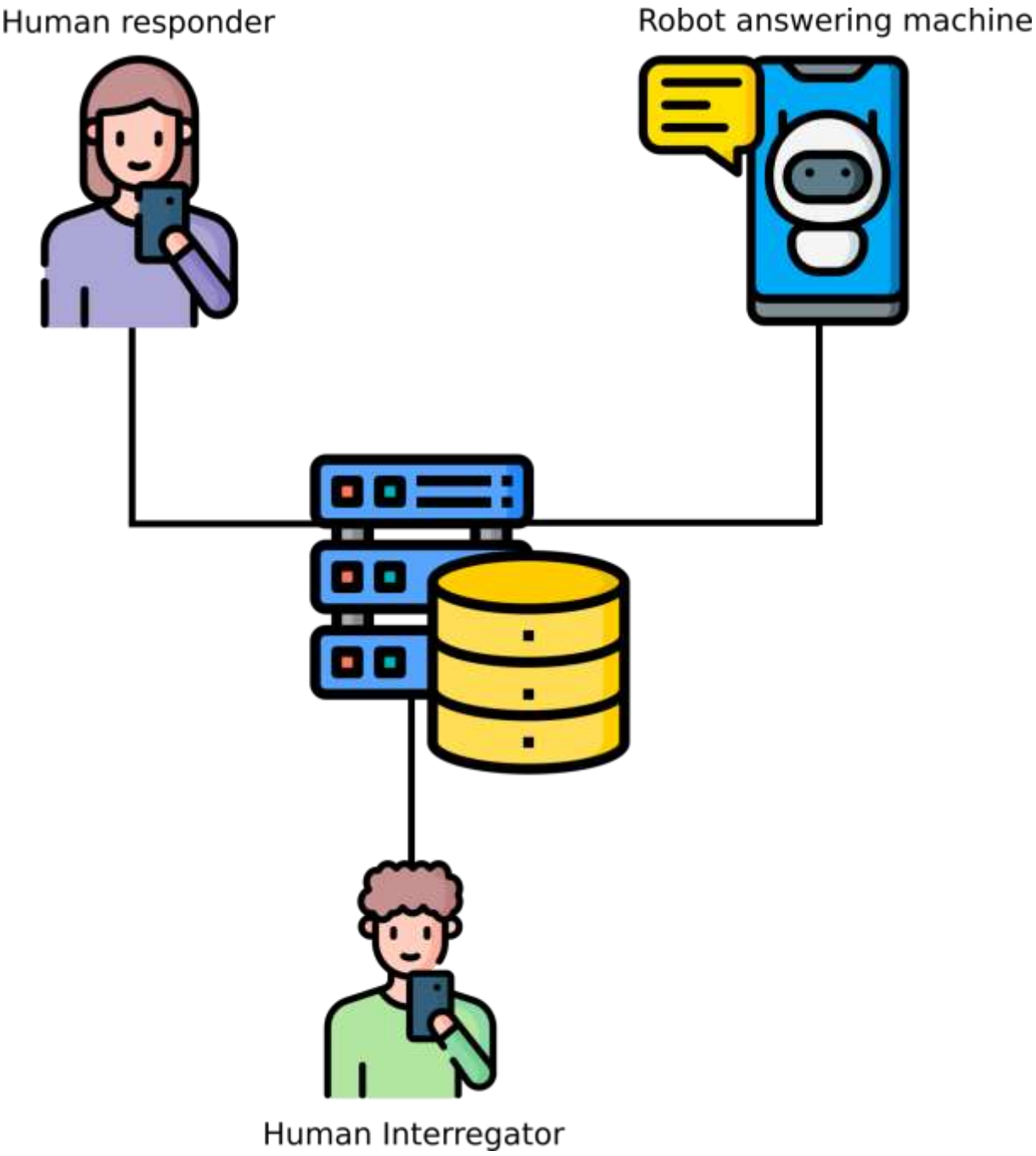

Figure 1: Image illustrative du test de Turing

II. Outils existants

1) Compilatio

Compilatio est un logiciel propriétaire de détection de similarité qui existe depuis 2005.[3] Proposée par l'entreprise française Compilatio, elle est utilisée grandement par beaucoup d'entreprises françaises[4] et est présente dans plus de 35 pays[5] dans le monde.

Compilatio permet d'analyser les similarités dans un document textuel (en format .txt, .odt, .pdf, .doc, .docx, .rtf, .xls, .xlsx, .ppt ou .pptx.), puis il analyse les similarités de ce document par rapport à ceux de sa base de données et produit un rapport d'analyse à la fin.[6]

2) Scribbr Plagiarism Checker

Le logiciel Scribrr Plagiarism Checker est un logiciel de détection de plagiat de la plateforme Scribbr.com.[7] C'est un logiciel efficace qui est toutefois payant. Il permet entre autres la détection des similarités de façon efficace, car il est propulsé par Turnitin, un autre logiciel de vérification de plagiat confirmé [24] . Également les soumissions sont comparées à une très grande base de données [24] .

3) Quetext

Quetext [8] est également un détecteur de plagiat assez efficace qui propose des offres gratuites et payantes. C'est un logiciel très efficace et avec un design permet de lire facilement les résultats. Toutefois, il n'est pas adapté pour la détection de plagiat dans les articles scientifiques.[9]

[3] https://www.compilatio.net/societe

[4] https://www.scribbr.fr/le-plagiat/compilatio/

[5] https://www.compilatio.net/societe

[6] https://www.scribbr.fr/le-plagiat/compilatio/

[7] https://www.scribbr.com/plagiarism-checker/

[8] https://www.quetext.com/

[9] https://digitiz.fr/blog/logiciel-anti-plagiat/

4) CopyScape

CopyScapey [10] quant à lui, est spécialisé dans la détection de similarité entre deux pages d'URL ou entre deux textes. Ainsi donc les fonctionnalités pouvant servir sont toutefois limitées en ce qui concerne l'aide dans les travaux académiques et scientifiques.

III. Algorithmes utilisés en NLP

Pour détecter les similarités, en NLP, il existe plusieurs modèles algorithmiques. Il existe en effets certains algorithmes qui effectuent de la comparaison syntaxique. C'est-à-dire qu'il compare directement le degré ressemblance des chaînes de caractères dans un document. Et il existe des algorithmes qui comparent non la ressemblance des chaînes de caractères uniquement, mais étendent la comparaison jusqu'à la ressemblance dans leur sens même s'il n'y pas de ressemblances syntaxiques [8] .Les algorithmes qui effectuent les similarités jusque dans leur sémantique sont les plus avancés de nos jours. Toutefois, pour pouvoir faire ces comparaisons, les algorithmes modernes utilisent les algorithmes de vectorisation qui consistent à créer des représentations numériques, sous forme de vecteurs des mots d'un document. La vectorisation est une étape très importante dans les algorithmes d'analyse de similarité basé sur l'IA. Ainsi, il est important de faire un point sur les algorithmes de NLP pour l'analyse des similarité et parcourir des travaux existants.

1) Algorithmes de vectorisation

a) Bag of words (BoW)

Bag of Words ou en français « Sac de mots » est un des algorithmes de vectorisation les plus basiques. La technique consiste principalement à créer un vocabulaire des mots utilisés dans le corpus. Ensuite, un vecteur de dimension ou de taille n (n étant le nombre de mots du vocabulaire) est créé pour chaque document du corpus [25] . On obtient ainsi une représentation numérique de notre texte. Cette technique, qui est relativement facile à mettre en œuvre, possède plusieurs avantages et inconvénients.

❖ **Avantages :**

- Facile à comprendre
- Facile à implémenter

[10] https://www.copyscape.com/compare.php

❖ **Inconvénients :**

- Ne permet pas de tenir compte de l'ordre des mots dans la vectorisation.
- Taille des vecteurs trop grands.

b) Term Frequency Inverse Document Frequency (TF-IDF)

Le TF-IDF est également une technique efficace qui se présente comme une continuité du BoW avec des améliorations majeures. En effet avec TF-IDF, l'ordre des mots est maintenant pris en compte et ainsi que l'importance des mots dans le document.

Pour effectuer du TF-IDF, l'algorithme a deux parties principales, la partie TF (Term Frequency) et la partie IDF (Inverse Document Frequency). La première partie TF calculant la fréquence des mots dans chaque document. La deuxième partie, le IDF (Inverse Document Frequency), permettant de prendre en compte l'importance du mot dans le corpus en calculant la IDF qui est définie comme le logarithme du rapport du nombre de documents dans un corpus par le nombre de documents dans le corpus où apparaît le mot [26] .

❖ **Avantages :**

- Reprends les avantages de BoW
- Prend en compte l'importance des mots dans un corpus

❖ **Inconvénients :**

- Ne tiens pas compte de la sémantique
- Taille de vocabulaire peut être grande et le calcul au niveau machine peut être coûteux en ressources mémoire

c) Word embedding Avec Word2Vec

Le Word embedding est une approche améliorée des précédents algorithmes. Il permet de tenir compte, en plus de la comparaison syntaxique, de la sémantique des mots. Pour cela, pour chaque mot, est créé, un vecteur de contexte dense (vecteur de dimension réduite). Chaque vecteur ayant des valeurs proches représente des mots similaires. Nous pouvons, en effet, dire que deux mots sont similaires s'ils ont des contextes similaire, grâce à la théorie de John Rupert Firth qui dit: "You shall know a word by the company it keeps" [11]. Il existe un modèle beaucoup

[11] https://www.sciencedirect.com/science/article/pii/S0028393214002942

utilisé de word embedding, qui utilise l'apprentissage automatique pour produire des vecteurs de mots, qui s'appelle Word2Vec.

Word2Vec est un modèle de Word embedding qui a été créé par Google en 2013 [14] .

❖ **Avantages :**

- Prend en compte la sémantique des mots
- Permet de faire des opérations arithmétiques sur les mots grâce aux vecteurs de contexte

❖ **Inconvénients :**

- Chaque mot à un seul vecteur de contexte et varie en fonction de son contexte

d) Transformer avec BERT & SBERT

Présenter dans le célèbre document « Attention Is All You Need » [4] , les transformers ont permis beaucoup d'avancer dans le NLP. Un transformer est une nouvelle approche d'apprentissage automatique qui utilise les mécanismes d'attention. Il permet de faire de l'apprentissage automatique de type « séquence à séquence » [5] .

Ainsi l'équipe de Google à proposer en 2018, un modèle nommé BERT (Bidirectional Encoder Representations from Transformers) se basant sur les transformers pour faire des taches NLP. Parmi lesquelles la traduction automatique, le calcule de similarité, etc. [5] . Toutefois il existe une version modifiée de BERT, nommée SBERT, qui produit d'excellents résultats.

Sentence-BERT (SBERT) est une version de BERT amélioré. SBERT utilise des réseaux de neurones siamois et réduit significativement le coût de calcule que produit BERT par défaut [30] . « SBERT [...] réduit l'effort pour trouver la paire la plus similaire de 65 heures avec BERT à environ 5 secondes avec SBERT, tout en maintenant la précision de BERT. » [30] . En effet, SBERT permet de rendre le traitement plus rapide.

❖ **Avantages :**

- Permet de comprendre le sens des mots dépendamment de leur contexte
- SBERT rend le calcule rapide et efficace

❖ **Inconvénients :**

- Peut-être coûteux pour entraîner les modèles personnalisés

2) Algorithmes de calcule de similarité

a) Similarité Jaccard

La Similarité Jaccard ou distance de Jaccard est une méthode de calcul de similarité entre deux ensembles d'éléments. La similarité Jaccard permet de voir à quel point deux ensembles de données partages des éléments communs. Soit deux ensembles de données A et B. La Similarité Jaccard entre A et B, que nous nommons Jaccard(A,B), consiste à faire le rapport, de la cardinalité de l'intersection des deux ensembles par la cardinalité de leur union. Le résultat donne le degré de similitude entre des deux ensembles sur une échelle de 0 à 1, 0 si les ensembles ne partagent aucune donnée commune et 1 s'ils sont identiques.[12] La similarité Jaccard est donc définie par la formule suivante :

$$Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Equation 1: Similarité Jaccard

- **Avantages :**
  - Facile à comprendre
  - Facile à mettre en œuvre
- **Inconvénients :**
  - Peut donner une similarité proche entre des données textuelles, pourtant différentes car il peut y arriver que deux textes différents contiennent des ensembles de données textuels proches.
  - Pas applicable sur les vecteurs de mots

b) Distance Euclidienne

« Les distances euclidiennes sont représentées par la mesure des distances par une règle à partir d'une vue d'oiseau. » [11] .

La distance euclidienne entre deux vecteurs $A = (a_1, a_2 \ldots, a_n)$ et $B = (b_1, b_2 \ldots, b_n)$, que nous nommons d(A,B), est définie par la formule générale :

[12] https://www.learndatasci.com/glossary/jaccard-similarity/

$$d(A,B) = \sqrt{\sum_{k=0}^{n}(a_k - b_k)^2}$$

Equation 2: Distance euclidienne

❖ **Avantages :**

- Permet le calcule de distance vectorielle
- Adapté pour regrouper les données qui ont des centres d'intérêts proches

❖ **Inconvénients :**

- Pas adapté pour la classification de données similaires vis à vis de certaines caractéristiques précises.

c) Similarité Cosinus

La similarité cosinus permet quant à lui de calculer la similitude de vecteurs vis à vis de leur orientation. En effet, deux vecteurs qui ont une distance angulaire faible convergent vers des cibles proches. Ainsi le calcul de similitude basé le cosinus permet dans notre cas de trouver les documents qui convergent vers le même sens [11] .

La similarité cosinus entre deux vecteurs $A = (a_1, a_2 \ldots, a_n)$ et $B = (b_1, b_2 \ldots, b_n)$, que nous nommons cos(θ), est définie par le rapport entre le produit scalaire des deux vecteurs par le produit de leurs normes :

$$cos(\square) = \frac{A.B}{\|A\|.\|B\|}$$

Equation 3:Similarité Cosinus

❖ **Avantages :**

- Permet le calcule de distance vectorielle
- Adapté pour la classification de données similaires vis à vis de certaines caractéristiques précises. Idéal pour la détection de similarité textuelle.

❖ **Inconvénients :**

- Pas adapté pour regrouper les données qui ont des centres d'intérêts proche

IV. Travaux dans le domaine

- Plagiarism Detection Using Transformers de Zoumana Keita[32] .

Dans la même optique que nos recherches, le travail de Zoumana Keita intitulé « Plagiarism Detection Using Transformers de Zoumana Keita »[32] , décrit une méthode algorithmique de détection de plagiat. Le problème qu'il aborde est plus précisément celui de trouver un ou plusieurs documents similaires à un nouveau document soumis. Dans son document, il aborde deux points spécifiques de la détection de plagiat dans les documents : le plagiat de paraphrase de contenu et le plagiat de traduction de contenu.

Pour cela, il fait le choix des modèles de ML basés sur les transfomers pour la résolution de cette problématique. Tout d'abord, il utilise BERT pour la vectorisation et ensuite MarianMT comme modèle de ML pour la traduction automatique en anglais si le document est écrit en allemand, français, japonais, grec ou russe. Enfin, pour le calcul de similarité entre les vecteurs, il utilise la similarité cosinus.

Pour son test, il utilisait le jeu de données cord-19. Les résultats de l'étude sont assez intéressants avec des tests sur trois documents écrits en anglais français et allemand.

- AI-POWERED PLAGIARISM DETECTION: LEVERAGING FORENSIC LINGUISTICS AND NATURAL LANGUAGE PROCESSING [31] .

Les travaux dans cet article visent également à proposer un système algorithmique basé sur l'IA pour la détection de plagiat [31] . Les objectifs des auteurs, Nwohiri et al., étaient de proposer un logiciel qui utilise l'IA pour détecter les plagiats à partir du web ou dans une bibliothèque local de documents, identifier le plagiat de codes source [31] .

Globalement, la solution proposée permet d'ajouter un ou plusieurs nouveaux documents au logiciel qui après analyse donne un rapport de plagiat et prodigue des suggestions de corrections [31] .

V. Choix algorithmique

Pour notre système de plateforme de gestion de détection de Plagiat, nous utiliserons SBERT pour la vectorisation et la similarité cosinus pour le calcul de similarité entre vecteurs.

Nous utilisons SBERT, car c’est un algorithme basé sur BERT très avancé qui tient compte de la sémantique.

Quant à l’utilisation de la similarité cosinus, nous avons fait ce choix car c’est le plus adapté en ce qui concerne l’analyse de similarité textuelle.

CHAPITRE 2: APPROCHE DE DÉTECTION DE PLAGIAT

Dans cette section, nous présentons notre approche (**Erreur ! Source du renvoi introuvable.**). P rincipalement, le fonctionnement de notre algorithme est de permettre la détection du plagiat en utilisant des algorithmes d'IA.

La recherche de similitude textuelle est le cœur de notre travail. Pour ce faire, de manière générale, un nouveau document (N) est soumis au système. Le système effectue un test de similarité entre ce document et ceux présents dans la base de données. Un seuil de plagiat est paramétrable dans le logiciel afin qu'un document dont le taux de similarité, avec tous les documents de la base, est inférieur ou égale au seuil autorisé soit accepté dans le système.

I. Approche proposée

Pour chaque comparaison entre le document entrant, *N* et un document dans la base de données, nous effectuons un ensemble d'étapes pour obtenir taux de similarité.

Soit *D* un tableau contenant la liste des documents de notre base de données et *D(i)* le document à l'indice i de la liste avec i représentant un entier naturel. Soit la fonction *taille(x)* le nombre d'éléments dans un tableau de liste x. Nous pouvons présenter ainsi notre approche dans la figure suivante :

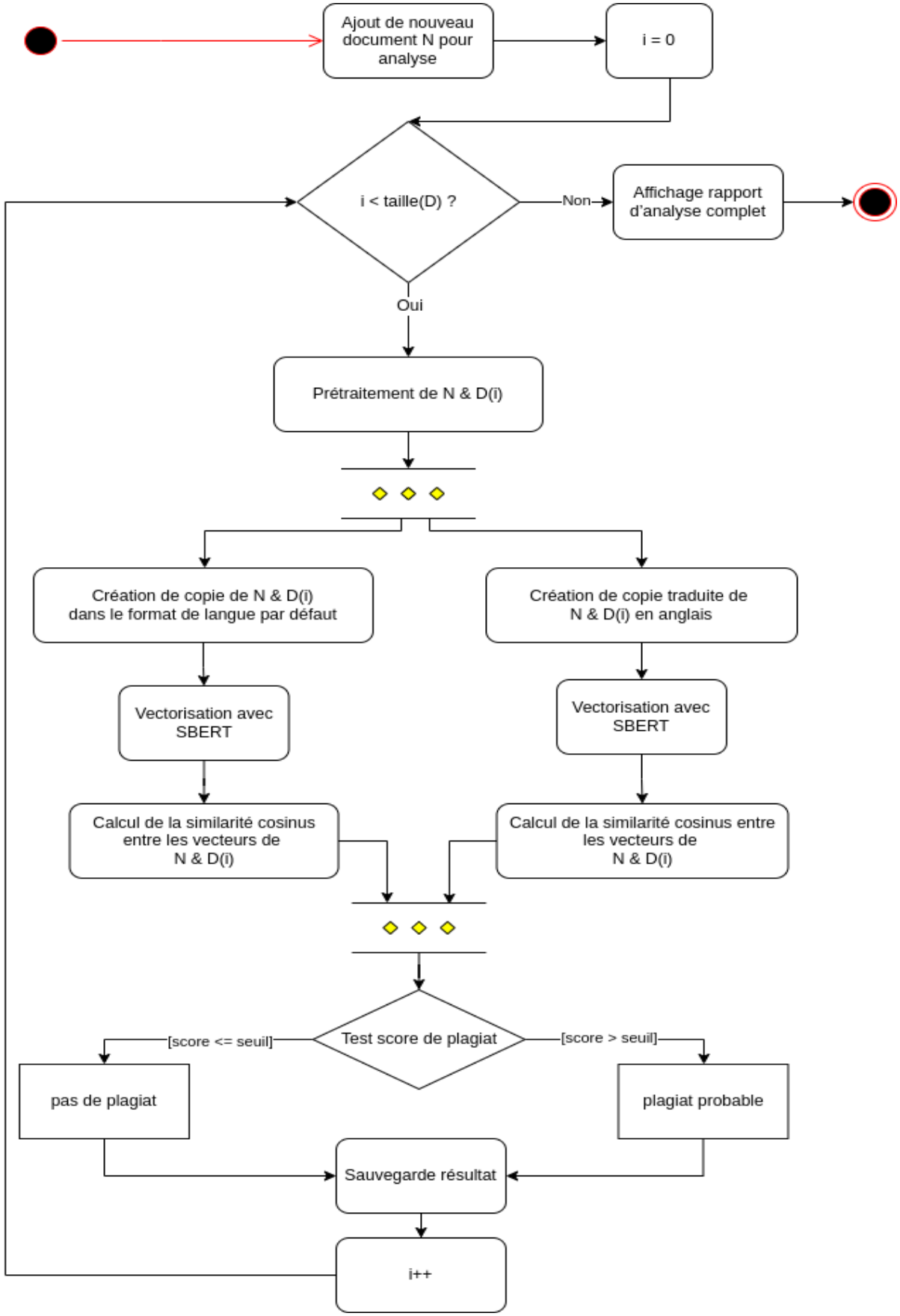

Figure 2:Diagramme du processus de détection de plagiat dans la base de données locale

Dans notre approche (Figure 2), lorsqu'un nouveau document est soumis pour analyse, celui-ci est comparé avec tous les autres documents de la base de données pris individuellement.

- Le prétraitement : le prétraitement est une étape nécessaire avant l'utilisation des données brutes. En effet il est important qu'un ensemble d'actions soit appliqué à ces données afin que les données à utilisées soient propres, débarrassées de toute impureté et soit dans un format convenable à une bonne exploitation. Dans notre cas, nous effectuons une étape de prétraitement pour chaque document avant application des algorithmes de comparaison.
- Deux versions probables du même document sont enregistrées: en effet, deux versions du même document sont stockées si la langue source du document n'est pas l'anglais. La première étant le document pré-traité dans la langue d'origine et dans la deuxième version étant une version traduite de la langue d'origine si toutefois la langue d'origine n'est pas déjà l'anglais. La raison de cette traduction étant inspirée, des travaux de Zoumana Keita [32] , permettra en effet de faciliter la comparaison entre les documents n'ayant pas les mêmes langues d'écriture. De plus, la vectorisation avec BERT nécessite d'utiliser des modèles d'apprentissage automatique entraînés alors que ces derniers ne sont pas forcément disponibles pour toutes les langues. Alors que l'anglais en regorge énormément.
- La vectorisation grâce à SBERT : après avoir obtenu les deux versions de nos documents, nous allons, effectuer la vectorisation grâce à SBERT.
- Le calcule de similarité : nos vecteurs ayant été obtenus, nous effectuerons un calcul de similarité cosinus sur chaque vecteur de même catégorie, c'est-à-dire résultant de la vectorisation entre deux documents de même langue d'écriture.
- Le résultat : après avoir obtenu le résultat du calcul de similarité cosinus, nous allons comparer ce résultat à notre seuil de plagiat autorisé, paramétrable dans le logiciel, et si le résultat de score de similarité est supérieur à ce seuil, nous signalerons un potentiel plagiat et dans le cas contraire non.

Puis nous répétons la boucle tant que tous les documents de la base de données ne sont pas comparé avec le nouveau document avant de renvoyer le résulta global de fin.

II. Analyse et conception de la plateforme

L'analyse et la conception sont des phases importantes en développement logiciel. Ainsi, dans cette partie, nous concentrons nos efforts à cette phase. Pour cela, nous utiliserons le langage de modélisation UML (Unified Modeling Language) dans notre démarche.

1) Cas d'utilisations

Le diagramme de cas d'utilisation permet de mettre en exergue les acteurs d'un système et les différentes actions possibles de ces acteurs dans le système.

a) Résumé des fonctionnalités de la plateforme

La liste des fonctionnalités principales de notre plateforme est la suivante :

- Authentification
- Gestion des utilisateurs
- Gestion de son espace de travail
- Création et gestion de dossier d'analyse
- Ajout de nouveaux documents dans un dossier
- Analyse de plagiat pour un document
- Visualisation de résultats d'analyse
- Gestion de la bibliothèque de références

b) Diagramme de cas d'utilisations

Notre diagramme de cas d'utilisation est le suivant :

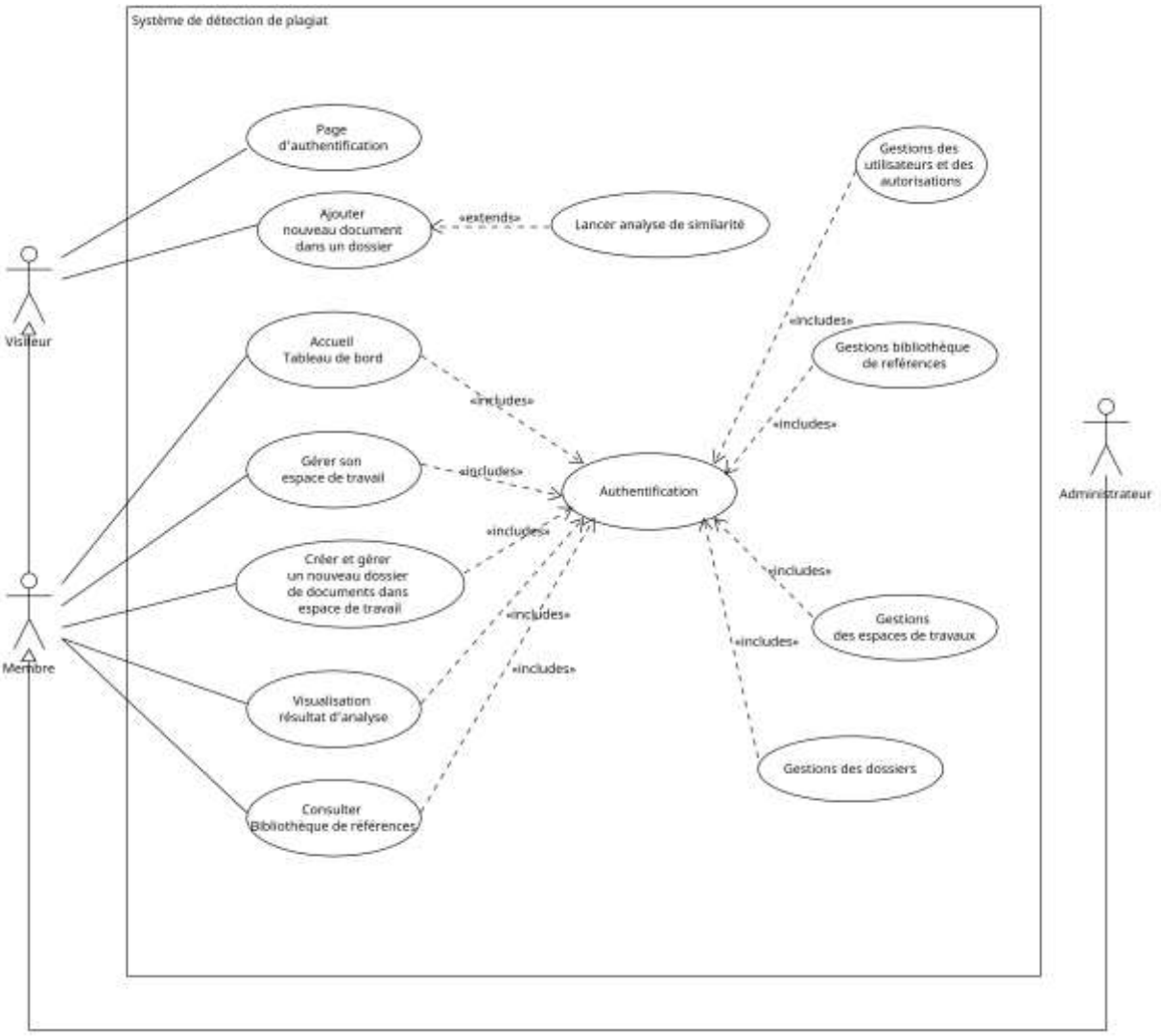

Figure 3:Diagramme de cas d'utilisation

Ce diagramme (Figure 3) nous montre que nous avons, trois profils différents à savoir le profil « visiteur », le profil « membre » et le profil « administrateur ». Un visiteur peut accéder à la page d'authentification ou ajouter un nouveau document à un dossier pour lequel il sera autorisé. Le membre, en plus de faire les actions du visiteur peut gérer son espace de travail, un dossier de son espace de travail, ajouter des documents dans ses dossiers et visualisé les résultats dans ses dossiers. Il peut également consulter la bibliothèque de références. Quant à l'utilisateur administrateur, il peut administrer toutes les données sur la plateforme.

2) Diagramme de classes

En génie logiciel, le diagramme de classes (**Erreur ! Source du renvoi introuvable.**) r eprésente la schématisation de l'ensemble des classes d'un système et des relations en elles.

a) Dictionnaire de données

Le dictionnaire de données est l'énumération et la description de l'ensemble des informations relatives aux données enregistrées dans notre base de données.

Attributs	Description	Type de données
Identifiants communs à certaines tables		
id	Identifiant	Int
slug	Identifiant unique sous forme de chaîne de caractères	String
created_at	Date de création	Date
updated_at	Date de mise à jour	Date
profile	Image de profile	String
cover	Image de couverture	String
name	Nom	String
description	Une description	String
Classe « User » (La Classe des utilisateurs)		
email	Adresse e-mail	String
phone	Numéro de téléphone	String
email_verified_at	Date de vérification de l'adresse e-mail	String
bio	Biographie	String
password	Mot de passe	String
Classe « Role » (Le Rôle d'un utilisateur comme par exemple Administrateur, Membre, etc.)		

Classe « Permission » (Représente une permission accordée à un utilisateur)		
Classe « Workspace » (Représente un espace de travail d'un utilisateur)		
Classe « Folder » (Représente un dossier dans un espace de travail)		
plagiarism_threshold_allowed	Seuil de plagiat autorisé dans un dossier	Int
allowed_public_access	Variable booléenne spécifiant si un dossier est à accès public ou pas. "Vraie" si l'accès est public.	String
allowed_users	Utilisateurs autorisés à déposer leur nouveau document dans un dossier	Json
comments	Commentaires	Json
submitter_email	Email du déposant	String
submitter_fullname	Nom complet du déposant	String
automatic_analysis	Variable booléenne spécifiant si l'analyse de plagiat est déclenchée automatiquement pour tout nouveau dossier entrant dans un dossier. "Vraie" si l'analyse est automatique.	String
Classe « AnalysisItem » (Représente un élément d'analyse dans un dossier. Concrètement, il contient les informations pour un document ajouté dans un dossier pour analyse.).		
analysis_result	Le résultat de l'analyse	Json
last_analysis_date	Dernière date d'analyse	Datetime

Classe « Document » (Contient les informations d'un document)		
url	Lien de localisation d'un document	String
Classe « Setting » (Les informations de paramétrage de notre plateforme)		
default_plagiarism_threshold_allowed	Seuil de plagiat autorisé dans un dossier par défaut dans le système	Int

Tableau 1: Dictionnaire de données

b) Diagramme de classes

Notre diagramme de classes est le suivant :

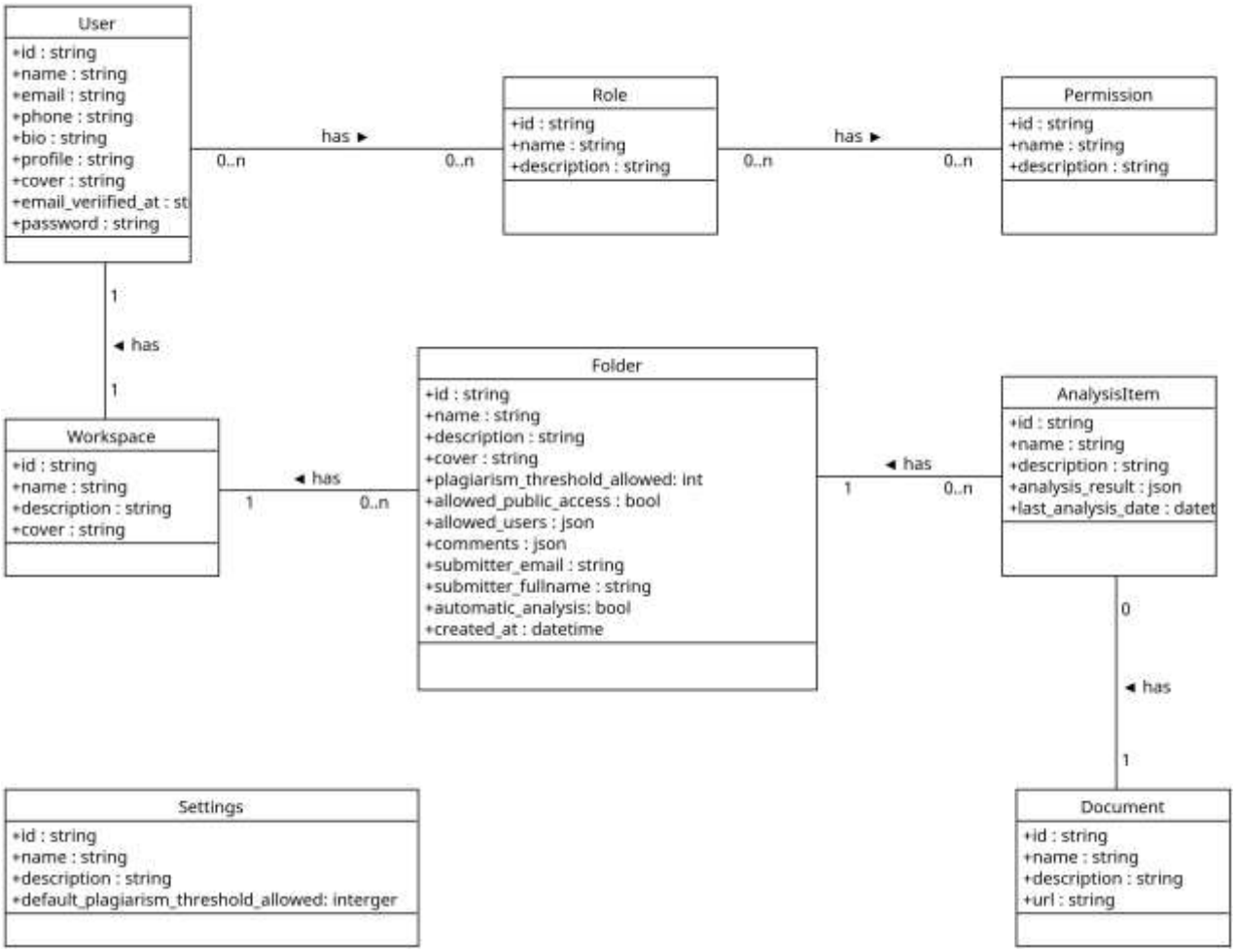

Figure 4:Diagramme de classes

Dans ce digramme (Figure 4) un utilisateur a 0 ou plusieurs rôles qui possèdes à leur tour 0 ou plusieurs permissions. Un utilisateur inscrit à un et seul espace de travail enregistré. Chaque espace de travail peut avoir plusieurs dossiers de plusieurs fichiers d'analyses. Chaque fichier d'analyse contient les informations d'un enregistrement de document physique. Nous avions également une entité pour enregistrer les informations de paramétrage de la plateforme.

3) Diagramme de déploiement

Notre diagramme de déploiement est comme suit :

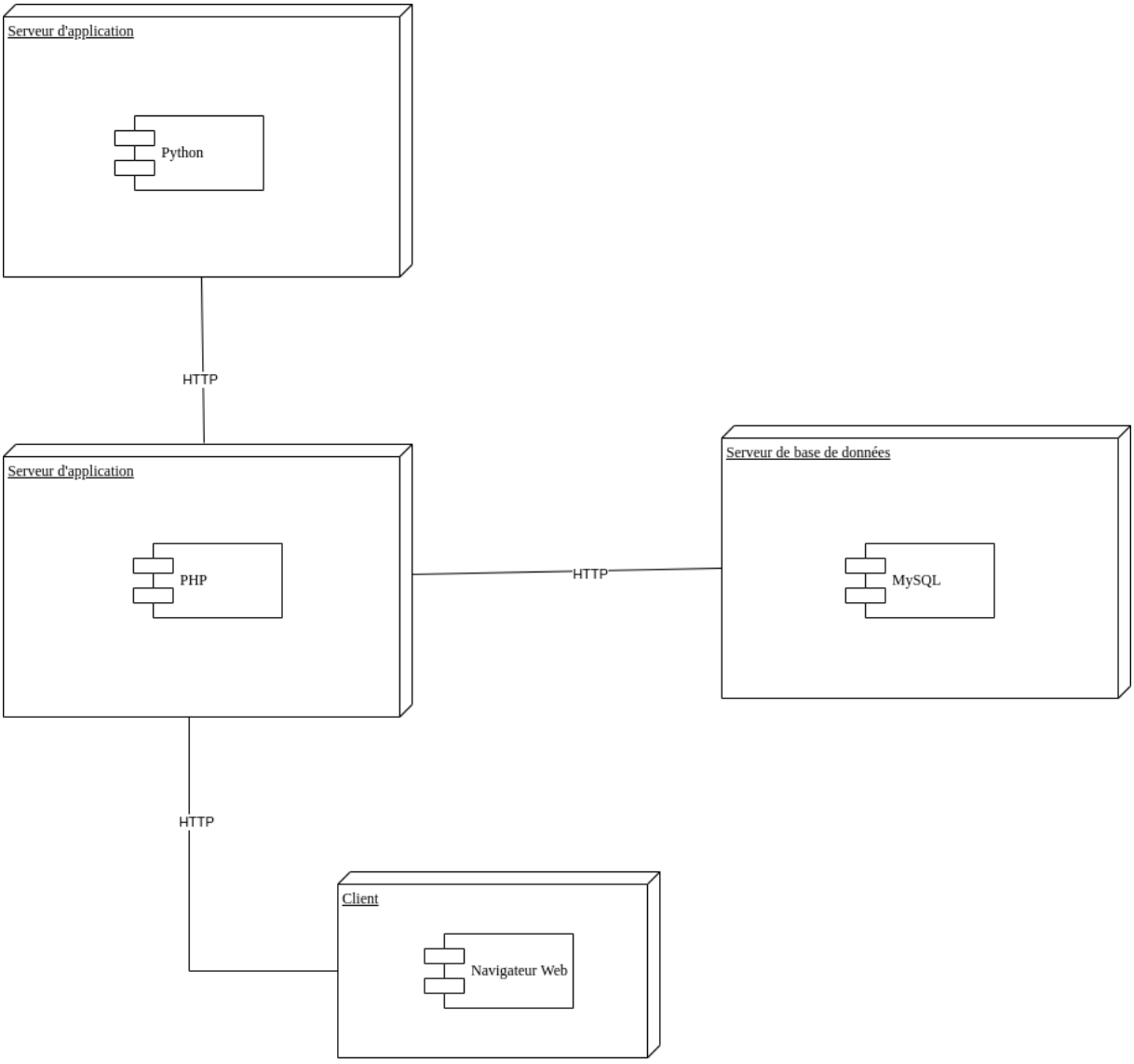

Figure 5:Diagramme de déploiement

Le déploiement (Figure 5) de notre plateforme suit une architecture n-tiers. D'abord le client qui fait une requête au serveur d'application contenant les codes PHP de notre plateforme notamment pour les pages web de l'application. Ensuite on l'application PHP qui peut éventuellement effectuer des opération (stockage, modification, suppression, récupération) sur la base de données afin de le fournir à l'utilisateur, ou dans le cas de lancement de l'analyse de plagiat accéder à un autre serveur pour exécuter le code python de notre détecteur de plagiat.

CHAPITRE 3: IMPLÉMENTATION DE LA PLATEFORME

I. Outils utilisés

Nous présentons ici l'ensemble des outils qui nous ont aidés, de notre travail de conception à la réalisation.

1) Langages de programmation utilisés

Les langages de programmation que nous avons utilisés sont :

- **Python 3 :** Python est un langage de programmation orienté objet, multiplateforme qui est beaucoup utilisé dans le monde de l'intelligence artificielle. Nous utilisons le langage Python pour nos algorithmes entrants dans le calcul de similarité.
- **PHP 8 :** PHP 8 est un langage de programmation impératif orienté objet qui permet d'exécuter du code coté serveur. Nous l'avions utilisé principalement pour la conception de certaines pages web dynamique pour notre plateforme.
- **JavaScript** : c'est un langage de script qui dans est initialement conçu pour apporter de l'interactivité aux pages web et donc s'exécutait côté client uniquement auparavant. Toutefois, il est utilisé de nos jours côté serveur également. Pour rendre notre plateforme interactive coté utilisateur, nous avons utilisé le langage de script JavaScript.
- **HTML 5 & CSS 3 :** Ce sont les langages de balisage standard pour la structuration des pages web. Nous les avions utilisés pour nos pages web.

2) Outils de conception

- **Visual Studio Code (VS Code) :** Cet éditeur de code développé par Microsoft a été notre outil d'édition de code dans le développement de notre plateforme.[13]
- **Git :** Git est outil de gestion de version décentralisé, libre et gratuit.[14] Il nous a aidés à gérer nos versions sur GitHub.
- **GitHub :** GitHub est une plateforme web qui permet de gérer les projets notamment informatiques et principalement la gestion de versions.[15]

[13] https://code.visualstudio.com/

[14] https://git-scm.com/

[15] https://github.com/

- **PhpMyAdmin :** phpMyAdmin nous a permis d'administrer notre base de données.
- **Draw.io ou Diagrams.net**: « diagrams.net est un logiciel de dessin graphique multiplateforme gratuit et open source développée en HTML5 et JavaScript.»[16].
- **ONLYOFFICE :** Logiciel bureautique gratuit, il nous a aidés dans notre rédaction documentaire.
- **Google Colab**[17] :Google Colab est environnement de travail intégré ; en ligne; qui permet d'exécuter du code python et de surcroît avec des ressources mémoires dédiées et personnalisables. Ce qui peut être très pratique pour des travaux d'IA. Nous l'avons utilisé pour écrire et tester une grande partie de nos codes python.

3) Plateformes et Frameworks de développement utilisés

- **Bootstrap 5 :** Framework CSS, Bootstrap 5 nous a été très utiles dans le design de notre interface utilisateur.
- **Laravel 8 :** Laravel est un Framework PHP très bien développé, Maintenu, qui possède une grande communauté. Il est également open-source. C'est notre Framework de base qui contient la majorité de nos codes plateformes.
- **SBERT Framework :** SBERT un Framework Python pour les incorporations de phrases, de textes et d'images à la pointe de la technologie. Il nous a aidés dans nos algorithmes de vectorisation.
- **MarianMT model** : un modèle de ML pré-entraîné qui nous a aidé dans la traduction automatique dans notre algorithme.
- **MySQL** : c'est un SGBD très populaire. Nous l'avons utilisé en tant que SGBD pour notre plateforme.
- **Apache HTTP Server** : Apache est un serveur d'application créé par Apache Software Foundation. Nous l'utilisons comme serveur d'application pour l'interprétation de nos pages web développé en PHP.

[16] https://diagrams.net

[17] https://colab.research.google.com

II. Tests et résultats

```python
#message original
original_message = """I stand here today humbled by the task before us, grateful for the trust you have bestowed,
mindful of the sacrifices borne by our ancestors. I thank President Bush for his service to our nation,
as well as the generosity and cooperation he has shown throughout this transition. The new movie is awesome.
The cat plays in the garden. The new movie is so great."""

#message plagié
rewritten_message ="""I am humbled by the task at hand, appreciative of the trust you have placed in me,
and conscious of the suffering endured by our forefathers as I stand here today.
I am grateful to President Bush for his service to our country,
as well as for his kindness and cooperation during this transition. The new movie is so great."""

#retourne la lists des phrases dans un text sous forme de list
def sentence_tokenizer(sentence: str) -> list:
    result = nltk.sent_tokenize(sentence)
    return result
```

Figure 6:Données de tests

En somme, les résultats de notre algorithme sont concluants. En effet, après avoir fourni des données (Figure 6) de test à notre algorithme nous obtenons, des résultats qui sont intéressants.

```python
#fonction d'analyse de la similarité qui retourne une list contenant les phrases qui ont un score de plagiat élevé
def analyze_similarity_between_sentences(document1_sentences: list, document2_sentences: list, threshold = 0.8) -> list:
    result = []

    sentences = document1_sentences + document2_sentences

    doc1_len = len(document1_sentences)

    paraphrases = util.paraphrase_mining(model, sentences, show_progress_bar=True)

    k = 0
    loop_stop = 1.0
    while loop_stop >= threshold:
        score, i, j = paraphrases[k]
        if (((i < doc1_len and j >= doc1_len) or (i >= doc1_len and j < doc1_len)) and len(sentences[i]) > 50 and len(sentences[j]) > 50):
            result.append([sentences[i], sentences[j], score])
        k = k + 1
        loop_stop = score
    #print(result)
    return result
```

Figure 7: Fonction principale d'analyse de similarité avec SBERT

La fonction principale (Figure 7) de notre algorithme prend en entrée les listes des phrases de nos deux documents et fournie les résultats de l'analyse sous forme de liste de listes. Chaque élément de la liste contenant dans l'ordre la phrase de premier document, la phrase du deuxième document et le score de similarité mais seulement si le score de similarité est supérieur à 80% par défaut.

```
[ ] #test de la similarité
    analyze_similarity_between_sentences(sentence_tokenizer(original_message), sentence_tokenizer(rewritten_message))

Batches: 100%  1/1 [00:00<00:00, 2.90it/s]
[['I thank President Bush for his service to our nation, \nas well as the generosity and cooperation he has shown throughout this transition.',
  'I am grateful to President Bush for his service to our country, \nas well as for his kindness and cooperation during this transition.',
  0.9402774572372437],
 ['I stand here today humbled by the task before us, grateful for the trust you have bestowed, \nmindful of the sacrifices borne by our ancestors.',
  'I am humbled by the task at hand, appreciative of the trust you have placed in me, \nand conscious of the suffering endured by our forefathers as I stand here today.',
  0.8086135998346718],
 ['I thank President Bush for his service to our nation, \nas well as the generosity and cooperation he has shown throughout this transition.',
  'I am humbled by the task at hand, appreciative of the trust you have placed in me, \nand conscious of the suffering endured by our forefathers as I stand here today.',
  0.8758381112098694]]
```

Figure 8: Résultat de l'analyse avec les données de test

Au regard des résultats (Figure 8) de ce test, nous pouvons dire que l'algorithme de base de notre moteur d'analyse de plagiat fournie des résultats concluants.

III. Quelques maquettes de la plateforme

1) Page d'accueil de l'espace de travail

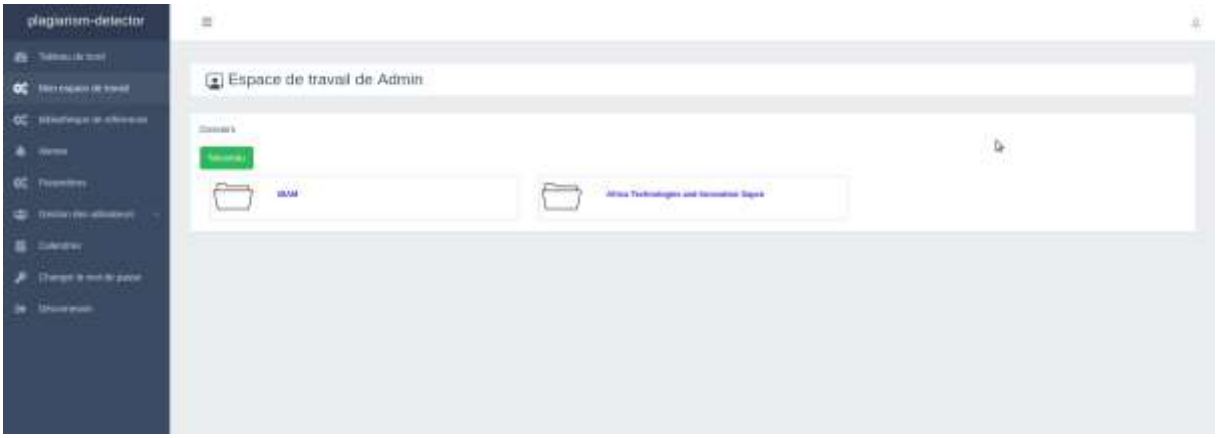

Figure 9:Page d'accueil de l'espace de travail

2) Page de création d'un nouveau dossier

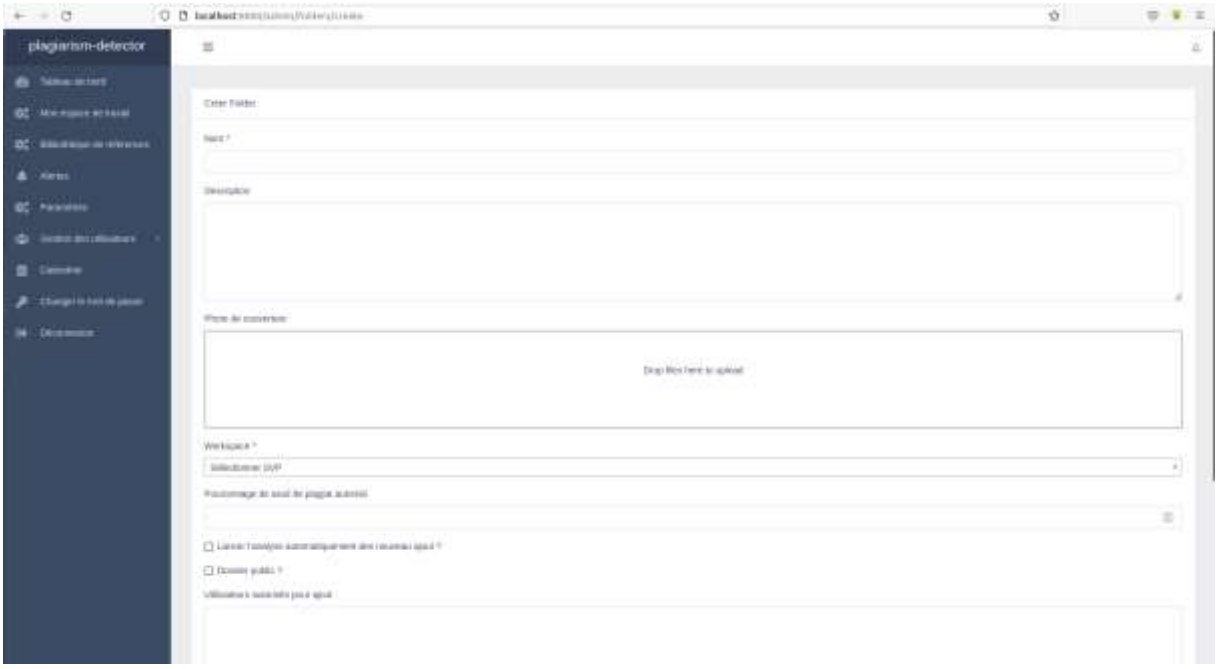

Figure 10:Page de création d'un nouveau dossier

3) ***Page de soumission d'un nouveau fichier pour analyse***

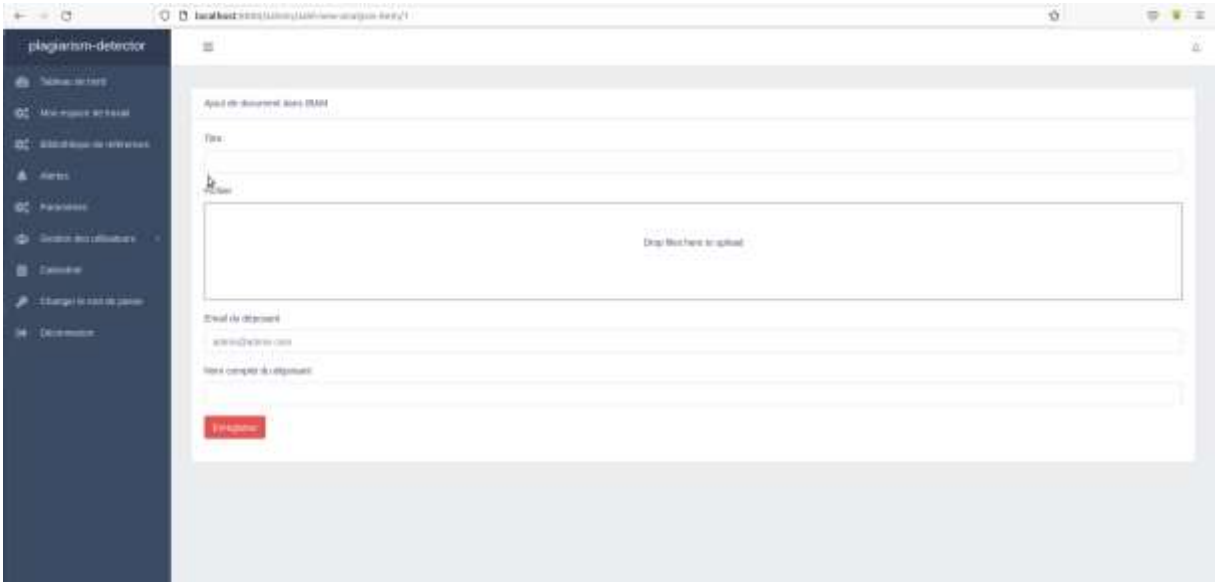

Figure : Page de soumission d'un nouveau fichier pour analyse

4) ***Page d'accueil de dossier et de visualisation de résultats***

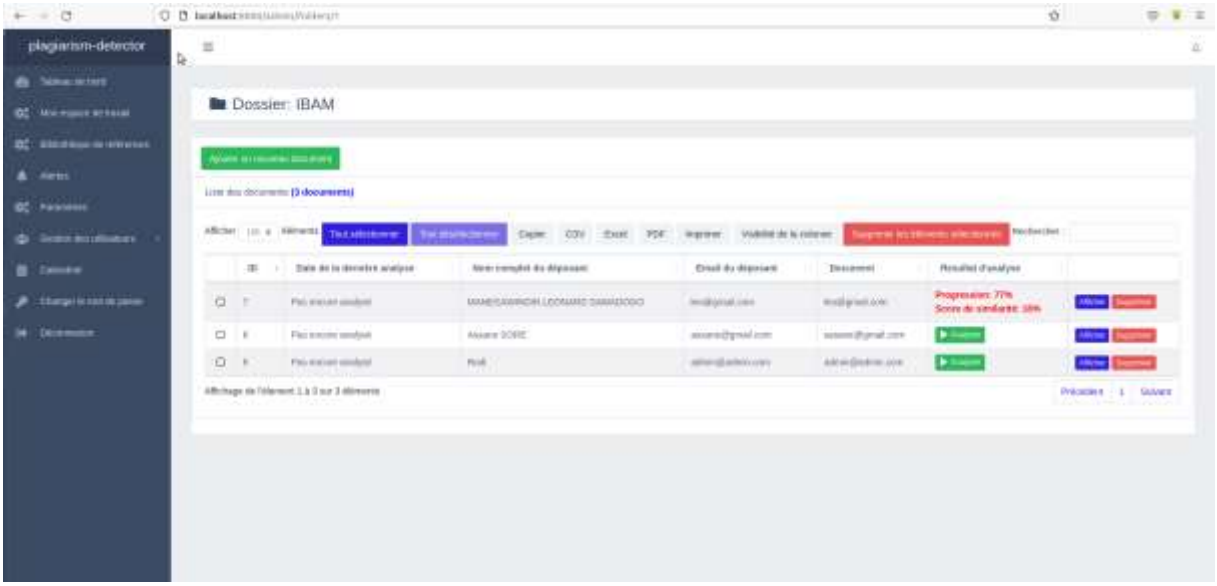

Figure 11:Page d'accueil de dossier et de visualisation de résultats

CONCLUSION

En somme, nous avons travaillé sur l'une des applications probables de l'IA et notamment en NLP. Le NLP a beaucoup d'avenir devant lui. La traduction automatique, l'analyse des sentiments, la recherche sémantique ou encore la similarité textuel sont parmi les nombreux champs d'action du NLP. Ce dernier point fut en effet le champ d'application de notre thème qui est « Système de détection de plagiat basé sur des algorithmes d'Intelligence Artificielle (IA) ».

Nous avons donc parcouru différents algorithmes entrant dans le cadre de la vectorisation et du calcul de distance entre ces vecteurs. Pour la vectorisation, nous en avons présenté plusieurs à savoir le Bag of Words, le TF-IDF, Le Word Embedding, BERT, et SBERT. Et concernant le calcule de similarité nous avons vu la méthode Jaccard (qui n'utilise pas en amont des vecteurs de mots), la distance euclidienne et la similarité cosinus. Après nos analyses donc nous avions choisie SBERT couplé avec le similarité cosinus pour la réalisation de notre système.

Les différentes étapes de notre travail nous ont permis d'apprendre davantage du monde de l'IA et notamment du NLP, de mettre en œuvre de façon concrète un système logiciel exploitant l'IA et donc de passer de la théorie à la pratique et aussi d'accroître notre passion envers ce domaine passionnant qu'est l'IA et particulièrement le NLP.

Aussi, beaucoup de perspectives s'offrent à nous afin de parfaire notre solution au grand bénéfice du monde éducatif et scientifique. Ainsi donc, comme tout produit, celui a besoin d'amélioration continue. De ce fait, nous avons comme perspectives de mettre en production la première version, ajouter des fonctionnalités de détection des similarités dans les sources multimédias (images, audio, vidéo, etc.), détecter le plagiat dans les codes source informatiques, améliorer de façon continue des algorithmes d'IA et du logiciel, etc. Ces idées, qui sont potentiellement inintéressantes, feront partie de nos prochains challenges.

BIBLIOGRAPHIE

[1] J. McCarthy, « WHAT IS ARTIFICIAL INTELLIGENCE? », p. 15.

[2] « À la découverte du Transformer », Le Data Scientist, 18 octobre 2020. https://ledatascientist.com/a-la-decouverte-du-transformer/ (consulté le 17 novembre 2022).

[3] A. Nwohiri, O. Joda, et O. Ajayi, « AI-POWERED PLAGIARISM DETECTION: LEVERAGING FORENSIC LINGUISTICS AND NATURAL LANGUAGE PROCESSING », FUDMA JOURNAL OF SCIENCES, vol. 5, nº 3, Art. nº 3, nov. 2021, doi: 10.33003/fjs-2021-0503-700.

[4] A. Vaswani et al., « Attention Is All You Need ». arXiv, 5 décembre 2017. Consulté le: 16 novembre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/1706.03762

[5] « BERT : Le “Transformer model” qui s’entraîne et qui représente », Les Dieux Du Code. https://lesdieuxducode.com/blog/2019/4/bert--le-transformer-model-qui-sentraine-et-qui-represente (consulté le 17 novembre 2022).

[6] M. Pietikäinen et O. Silvén, Challenges of Artificial Intelligence -- From Machine Learning and Computer Vision to Emotional Intelligence. 2022.

[7] « Classement des États du monde par nombre de publications scientifiques », Atlasocio.com. https://atlasocio.com/classements/education/publications/classement-etats-par-nombre-publications-scientifiques-monde.php (consulté le 20 novembre 2022).

[8] Pierre, « Comparaison de similarités syntaxiques avec Jaccard, Dice, TF-IDF et BM25 en Python », Anakeyn, 3 septembre 2021. https://www.anakeyn.com/2021/09/03/comparaison-similarites-syntaxiques-jaccard-dice-tf-idf-bm25/ (consulté le 15 novembre 2022).

[9] « Compare Two Web Pages or Articles - Copyscape ». https://www.copyscape.com/compare.php (consulté le 10 novembre 2022).

[10] « Compilatio : solution de détection de similitudes », p. 35.

[11] P. Baeldung, « Distance euclidienne vs similarité cosinus | Baeldung sur l’informatique », 23 juin 2020. https://www.baeldung.com/cs/euclidean-distance-vs-cosine-similarity (consulté le 18 novembre 2022).

[12] « Écriture originale, simplifiée | Quetexte ». https://www.quetext.com/ (consulté le 10 novembre 2022).

[13] « Figure 2.1. The Turing test. (© 123RF) », ResearchGate. https://www.researchgate.net/figure/The-Turing-test-C-123RF_fig2_357618741 (consulté le 22 novembre 2022).

[14] « Google Code Archive - Long-term storage for Google Code Project Hosting. » https://code.google.com/archive/p/word2vec/ (consulté le 16 novembre 2022).

[15] L. BOURDOIS, « ILLUSTRATION DU BAG-OF-WORDS », Loïck BOURDOIS, 26 novembre 2019. https://lbourdois.github.io/blog/nlp/Bag-of-word/ (consulté le 15 novembre 2022).

[16] « Introduction au NLP | Natural Language Processing | Jedha Bootcamp ». https://www.jedha.co/blog/introduction-au-nlp-natural-language-processing (consulté le 22 novembre 2022).

[17] « Le plagiat, un “phénomène exponentiel” dans les universités françaises », LExpress.fr, 4 septembre 2019. https://www.lexpress.fr/education/le-plagiat-un-phenomene-exponentiel-dans-les-universites-francaises_2096229.html (consulté le 20 novembre 2022).

[18] J. Debret, « Les 5 types de plagiat », Scribbr, 11 avril 2018. https://www.scribbr.fr/le-plagiat/types-de-plagiat/ (consulté le 9 novembre 2022).

[19] « Natural Language Processing – Calcul de similarité entre deux textes | Smals Research ». https://www.smalsresearch.be/natural-language-processing-calcul-de-similarite-entre-deux-textes/ (consulté le 10 novembre 2022).

[20] « Plagiarism Checker | Viper Online », Viper Plagiarism Checker. https://www.scanmyessay.com/ (consulté le 10 novembre 2022).

[21] g.haumont#utilisateurs-ldap, « Plagiat : l’UA mise sur la pédagogie », 31 janvier 2019. https://www.univ-angers.fr/fr/vous-etes/etudiant-e/examens/plagiat.html (consulté le 9 novembre 2022).

[22] « Planetoscope - Statistiques : Publications d’articles scientifiques dans le monde ». https://www.planetoscope.com/entreprises/2026-publications-d-articles-scientifiques-dans-le-monde.html (consulté le 20 novembre 2022).

[23] Digitiz, « Quel logiciel anti-plagiat choisir pour éviter le contenu dupliqué », Digitiz, 27 avril 2021. https://digitiz.fr/blog/logiciel-anti-plagiat/ (consulté le 10 novembre 2022).

[24] « Scribbr Plagiarism Checker (Premium-only) », Scribbr. https://www.scribbr.com/plagiarism-checker/ (consulté le 10 novembre 2022).

[25] V. Jayaswal, « Text Vectorization: Bag of Words (BoW) », Medium, 28 septembre 2020. https://towardsdatascience.com/text-vectorization-bag-of-words-bow-441d1bfce897 (consulté le 15 novembre 2022).

[26] V. Jayaswal, « Text Vectorization: Term Frequency — Inverse Document Frequency (TFIDF) », Medium, 4 octobre 2020. https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d (consulté le 15 novembre 2022).

[27] R. Bachelet, « Voler des idées : Le plagiat », p. 37.

[28] « Word2vec : NLP & Word Embedding - DataScientest », Formation Data Science | DataScientest.com, 18 septembre 2020. https://datascientest.com/nlp-word-embedding-word2vec (consulté le 16 novembre 2022).

[29] Z. Sadeghi, J. L. McClelland, et P. Hoffman, « You shall know an object by the company it keeps: An investigation of semantic representations derived from object co-occurrence in visual scenes », Neuropsychologia, vol. 76, p. 52-61, sept. 2015, doi: 10.1016/j.neuropsychologia.2014.08.031.

[30] Reimers, Nils, et Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. arXiv:1908.10084, arXiv, 27 août 2019. *arXiv.org*, http://arxiv.org/abs/1908.10084.

[31] (PDF) AI-POWERED PLAGIARISM DETECTION: LEVERAGING FORENSIC LINGUISTICS AND NATURAL LANGUAGE PROCESSING. https://www.researchgate.net/publication/356189052_AI-POWERED_PLAGIARISM_DETECTION_LEVERAGING_FORENSIC_LINGUISTICS_AND_NATURAL_LANGUAGE_PROCESSING. Consulté le 13 décembre 2022.

[32] Plagiarism Detection Using Transformers | Pinecone. https://www.pinecone.io/learn/plagiarism-detection/. Consulté le 13 décembre 2022.2) AI-POWERED PLAGIARISM DETECTION: LEVERAGING FORENSIC LINGUISTICS AND NATURAL LANGUAGE PROCESSING.

[33] MANEGAWINDIN LEONARD SAWADOGO, « Détection et classification des intrusions en utilisant des techniques de machine learning ». MÉMOIRE DE MASTER : Ingénierie Logicielle et Système d'Information Informatisé : ILSII ; UNIVERSITÉ JOSEPH KI-ZERBO, Unité de Formation et de Recherche en Sciences Exactes et Appliquées (UFR-SEA).

[34] KABORE Abdoul Kader, « Optimisation et compression de modèles de réseaux de neurones artificiels ». MÉMOIRE DE MASTER : Ingénierie Logicielle et Système d'Information Informatisé : ILSII ; U NIVERSITÉ OUAGA 1 PR JOSEPH KI ZERBO, Unité de Formation et de Recherche en Sciences Exactes et Appliquées.

TABLE DES MATIERES

RÉSUMÉ i

ABSTRACT ii

DEDICACE iii

REMERCIEMENTS iv

SOMMAIRE v

LISTE DES SIGLES ET ABREVIATIONS vii

LISTE DES FIGURES GRAPHIQUES viii

LISTE DES TABLEAUX ix

INTRODUCTION GENERALE 1

Chapitre 1: ÉTAT DE L'ART 3

I. Concepts généraux 3

1) Le plagiat 3

a) Définition 3

b) Les types de plagiats 3

c) Les causes 3

d) Les conséquences 4

e) Prévention 4

f) Protéger son œuvre 4

2) L'Intelligence Artificielle (IA) 5

3) Natural Language processing (NLP) 5

II. Outils existants 7

1) Compilatio 7

2) Scribbr Plagiarism Checker 7

3) Quetext 7

4) CopyScape 8

III. Algorithmes utilisés en NLP 8
1) Algorithmes de vectorisation 8
a) Bag of words (BoW) 8
b) Term Frequency Inverse Document Frequency (TF-IDF) 9
c) Word embedding Avec Word2Vec 9
d) Transformer avec BERT & SBERT 10
2) Algorithmes de calcule de similarité 11
a) Similarité Jaccard 11
b) Distance Euclidienne 11
c) Similarité Cosinus 12
IV. Travaux dans le domaine 13
V. Choix algorithmique 13
Chapitre 2: APPROCHE DE DÉTECTION DE PLAGIAT 15
I. Approche proposée 15
II. Analyse et conception de la plateforme 17
1) Cas d'utilisations 18
a) Résumé des fonctionnalités de la plateforme 18
b) Diagramme de cas d'utilisations 18
2) Diagramme de classes 19
a) Dictionnaire de données 20
b) Diagramme de classes 22
3) Diagramme de déploiement 23
Chapitre 3: IMPLÉMENTATION DE LA PLATEFORME 25
I. Outils utilisés 25
1) Langages de programmation utilisés 25
2) Outils de conception 25

3) Plateformes et Frameworks de développement utilisés 26

II. Tests et résultats 27

III. Quelques maquettes de la plateforme 28

1) Page d'accueil de l'espace de travail 28

2) Page de création d'un nouveau dossier 28

3) Page de soumission d'un nouveau fichier pour analyse 29

4) Page d'accueil de dossier et de visualisation de résultats 29

CONCLUSION 30

BIBLIOGRAPHIE 31

TABLE DES MATIERES 35