Table des matières

Table des figures

Liste des tableaux

Les acronymes

R&D	*Recherche et Développement*
MOA	*Maîtrise d'Ouvrage*
MOE	*Maîtrise d'œuvre*
TMA	*Tierce Maintenance Applicative*
PMO	*Project Management Office*
CRISP-DM	*Cross Industry Standard Process for Data Mining*
NLP	*Natural Language Processing*
TALN	*Traitement Automatique du Langage Naturel*
OCR	*Optical Character Recognition*
CSV	*Comma Separated Values*
TF-IDF	*term Frequency-Inverse Document Frequency*
GloVe	*Global Vectors for Word Representation*
BERT	*Bidirectional Encoder Representation from Transformers*
SBERT	*Sentences Bidirectional Encoder Representation from Transformers*
MLM	*Modélisation du langage masqué*

Chapitre 1

Contexte General

1.1 Introduction

Dans ce chapitre nous allons présenter dans la première partie l'organisme d'accueil dans laquelle notre projet de fin d'étude a été effectué. Dans la deuxième partie nous allons élaborer la présentation du projet en parlant du cadre de projet et la problématique trouvée. Dans la troisième partie, nous allons établir une étude de l'existant et ses limites en présentant la solution proposée avec une étude concurrentielle. Dans une dernière partie, nous présenterons le domaine de l'intelligence artificielle et la méthodologie adaptée.

1.2 ENVIRONNEMENT DU PROJET

Dans cette section, nous allons présenter l'entreprise d'accueil ainsi que les différents secteurs d'activités.

1.2.1 Établissement d'accueil

Yellowsys a été fondée en 2017. Yellowsys est une cabinet de conseil spécialisé dans l'accompagnement des entreprises sur les domaines de la transformation digitale : Big Data, Business Intelligence. Fort de ses succès et d'une croissance continue depuis sa création, Yellowsys :

- compte aujourd'hui plus de 30 collaborateurs,
- a réalisé plus de 40 projets,
- réinvestit plus de 5% de son chiffre d'affaires annuel dans la R&D afin d'encourager ses collaborateurs à la création et l'innovation via YellowLABS.

YellowSys a pour mission d'aider les entreprises d'accélérer leur performance et leur développement en générant de nouveaux leviers de croissance, de compétitivité et de pérennité.

Fig. 1.1 : Logo YELLOWSYS

1.2.2 Domaine d'expertise

- ***CONSEIL*** **:** Yellowsys participe dans la transformation digitale des entreprises, elle propose une expertise unique qui associe le conseil, l'assistance à la MOA et le MOE, le pilotage des projets et essentiellement le PMO.

- ***TMA/SUPPORT*** : Yellowsys veille au bon fonctionnement des systèmes de gestion de base de données et de reporting tout en corrigeant les failles réclamées par les utilisateurs.

- ***DATA PROJECTS*** : Data projets est l'une des missions de Yellowsys. Ce domaine d'expertise permet aux clients la mise en place des systèmes décisionnels de façon agile et personnalisée. Elle développe des plateformes permettant la création du reporting.

1.3 Présentation du projet

Dans cette section, nous présentons le cadre du projet et la problématique.

1.3.1 Cadre du projet

Notre projet est réalisé dans le cadre de l'obtention du diplôme de la licence en informatique de gestion spécialité Business Intelligence pour l'année universitaire 2021/2022.

Ce projet consiste à " développer une solution intelligente de Matching de documents financiers afin d'identifier les écarts et permettre à des utilisateurs de procéder à des annotations".

1.3.2 Problématique

De nos jours, les auditeurs internes fournissent une ligne de défense contre les erreurs et les omissions évitables qui peuvent réduire la qualité, nuire à la réputation et à la fiabilité

d'une entreprise, manquer des opportunités ou causer des pertes financières directes. Ces activités sont menées au moyen d'évaluations fondées sur les risques et de communications avec les comités du conseil d'administration, tandis que les audits financiers se concentrent sur la découverte de problèmes potentiellement importants et parfois de corrections dans les registres des transactions.

En d'autres termes, les pertes constatées lors de l'audit financier se sont déjà produites, alors que les résultats des travaux d'audit interne montrent des lacunes en matière de conformité, de qualité, et d'autres. Pour identifier ces lacunes, les opérations d'audit interne nécessitent le suivi des résultats dans chaque domaine de risque auquel les programmes de l'organisation sont exposés. Ce suivi nécessite une contextualisation et des résultats rapides, et non seulement un rapport de synthèse annuel au comité d'audit.

1.4 Étude de l'existant, Critique de l'existant, solution proposée et critique de concurrents

Dans cette section, nous allons présenter les difficultés rencontrées avec le système existant ainsi que la solution proposée pour résoudre le problème et l'étude concurrentielle.

1.4.1 Étude de l'existant

Il est difficile de consulter les bilans financiers des entreprises et dégager les écarts d'une année à une autre dans le but d'analyser la performance. Pour atteindre ces objectifs, l'auditeur va réaliser une comparaison manuellement puis il va noter les écarts dans un autre rapport pour avoir les informations nécessaires.

Cependant, cela ne résout pas le problème puisqu'il y a un risque d' erreurs au niveau de saisie d'où la nécessité d'une solution pour faciliter l'analyse et déduire les écarts rapidement.

1.4.2 Critique de l'existant

Vu qu'un grand volume de documentation pour chaque année, les tâches d'audit interne au sein des grandes organisations sont ralenties. Ainsi nous avons identifié plusieurs problèmes :

- Les temps de réponse d'audit lents, la planification d'audit basée sur des échantillons et le recours aux recherches par mots-clés sont tous des indicateurs que l'automatisation est nécessaire pour accélérer les tâches d'audit interne.
- Difficultés à gérer la surcharge d'information.
- Risque d'erreurs au niveau de saisie et la comparaison.
- Précision faible de données pour le rapport d'audit.
- Les données étaient uniquement stockées dans plusieurs fichiers PDF non structurés.

1.4.3 Solution proposée

Après avoir analysé l'existant, nous avons proposé une solution intelligente qui exécute automatiquement des écarts à l'appui des tâches intensives en informations des auditeurs internes. Notre solution est dédiée aux équipes d'audit qui peut les aider à donner un sens à de grandes quantités de documentation.
Nous proposons également une application Web permettant aux auditeurs d'effectuer un examen. La plate-forme offre aux auditeurs une capacité simplifiée et autonome d'inclure manuellement les informations découvertes au cours de la recherche en réduisant les étapes entre l'identification des informations dans les ensembles de données de documents. Ainsi les principaux objectifs assignés au projet sont les suivants :

- Traitement plus rapide des données.
- Traitement de documents non structurés.
- Augmentation de la précision des données.
- Réduction des coûts.

La figure ci dessous schématise quelques détails que la solution proposée.

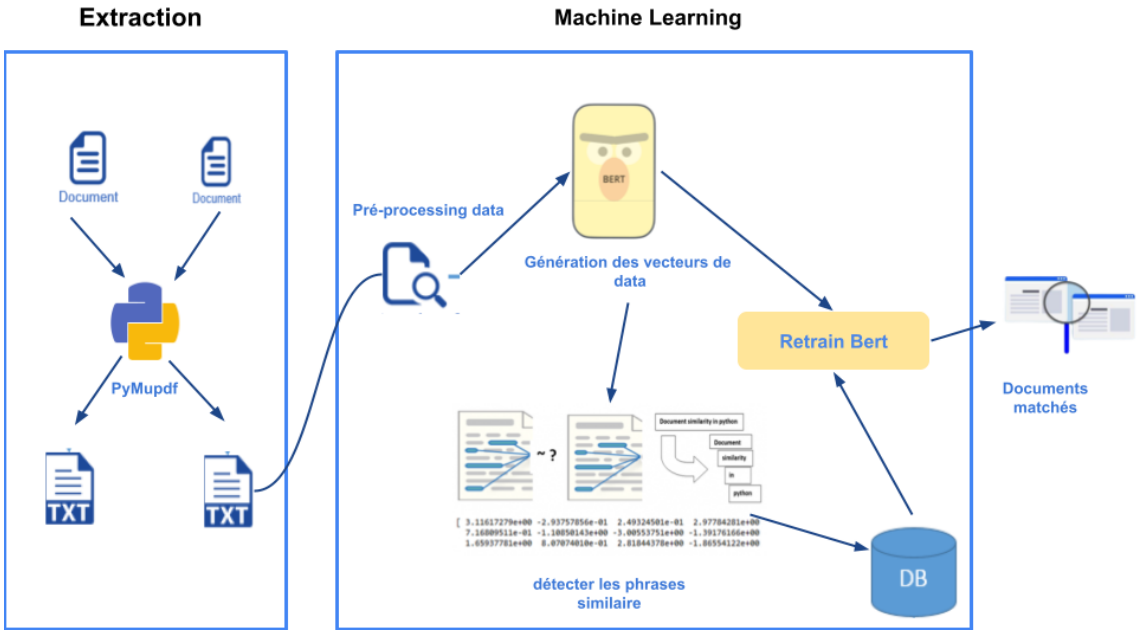


FIG. 1.2 : Schéma du solution

1.4.4 Étude concurrentiel

Dans cette partie nous allons réaliser une étude de deux solutions existantes qui ont un but principal le « matching » entre deux documents l'une est PDF24 et l'autre solution est COPYLEAKS.

PDF24 : Est une solution open source qui permet de comparer deux PDF et d'identifier les écarts. Cette solution n'est pas faisable et assez limitée pour résoudre le problème et faciliter la comparaison . Ces limites sont illustrés dans les points suivantes :

- Le résultat de comparaison est illisible et incompréhensible, il est difficile de distinguer les écarts entre les deux PDFs.
- La comparaison est effectuée phrase par phrase ce qui empêche de trouver réellement le mot différent nécessaire pour la distinction des écarts.
- La distinction des synonymes et des fautes d'orthographe ne sont pas disponibles, d'où la prise en compte de deux mots sémantiquement égaux à une déviation.

COPYLEAKS : "Détectez le plagiat, le contenu paraphrasé et le texte similaire à l'aide d'algorithmes sophistiqués basés sur l'intelligence artificielle dans plus de 100 langues avec notre logiciel anti-plagiat en ligne."[1]
Ce site contient une fonctionnalité "File Comparison Tool" qui a pour objectif de détecter la similarité entre deux PDF en surlignant avec des couleurs différentes les phrases identiques, les phrases qui ont en commun une signification associée et des phrases totalement différentes. Cependant cet outil ne répond pas aux besoins. En effet, le but de l'auditeur est de chercher la différence exacte. Par exemple, nous avons capturé les deux PDF trouvés dans la figure suivante, le résultat montre que la phrase qui contient le chiffre d'affaires est presque identique mais dans notre cas nous voulons dégager ce chiffre comme un écart qui est nécessaire dans l'analyse.

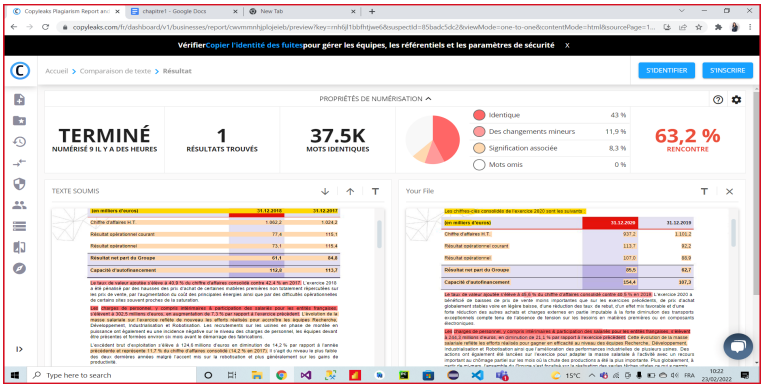


Fig. 1.3 : Résultat de comparaison deux PDF COPYLEAKS

1.5 Choix de la Méthodologie

1.5.1 Introduction

Pour l'optimisation de la performance et des possibilités d'innovation, il est nécessaire de disposer une méthodologie structurée à suivre.

Pour la définir, une méthodologie est un ensemble de méthodes, de règles et de principes employés par une discipline qui est, dans notre cas, la science des données.

Dans ce cadre nous avons choisis de travailler avec la méthodologie agile CRISP-DM qui est adaptable et itérative, elle nous permet de créer un modèle d'exploration de données adapté à nos besoins.

1.5.2 CRISP-DM

CRISP-DM est une méthodologie agile de gestion des projets de sciences des données comprend des descriptions des phases typiques d'un projet et des tâches comprises dans chaque phase, et une explication des relations entre ces tâches. Elle est un processus qui offre un aperçu sur le modèle de cycle de vie de l'exploration des données.
La méthode CRISP-DM se décompose en 6 étapes allant de la compréhension du problème métier au déploiement et la mise en production.

- **Compréhension de l'entreprise** : cette étape consiste à déterminer les besoins métiers et les objectifs attendus .

- **Compréhension des données** : Après la compréhension de l'entreprise, nous avons la phase de collecte de données identifiée afin de les analyser et à nous aider à atteindre l'objectif final . Cette phase comprend quatre étapes principales : collecter, décrire, explorer et vérifier la qualité des données.

- **Préparation des données :** Cette phase consiste à exploiter les données sources initiales, les nettoyer et les intégrer afin d'être modélisées par la suite.

- **Modélisation :** cette étape est basée sur diverses techniques de modélisation qui servent à la construction et l'évaluation de différents modèles. Cette phase comporte quatre tâches passant de la sélection des techniques de modélisation, de la génération de la conception teste et de modèle de construction jusqu'à l'évaluation de modèle.

- **Évaluation :** C'est une phase nécessaire pour évaluer les modèles choisis dans la phase de modélisation et de comparer l'efficacité de chaque modèle afin qu'il réponde au besoin de l'entreprise. Elle est résumée en trois tâches : évaluer les résultats, processus d'examen et déterminer les étapes suivantes.

- **Déploiement :** C'est l'étape finale de processus qui est dédiée à la mise en production pour les utilisateurs finaux des modèles obtenus.

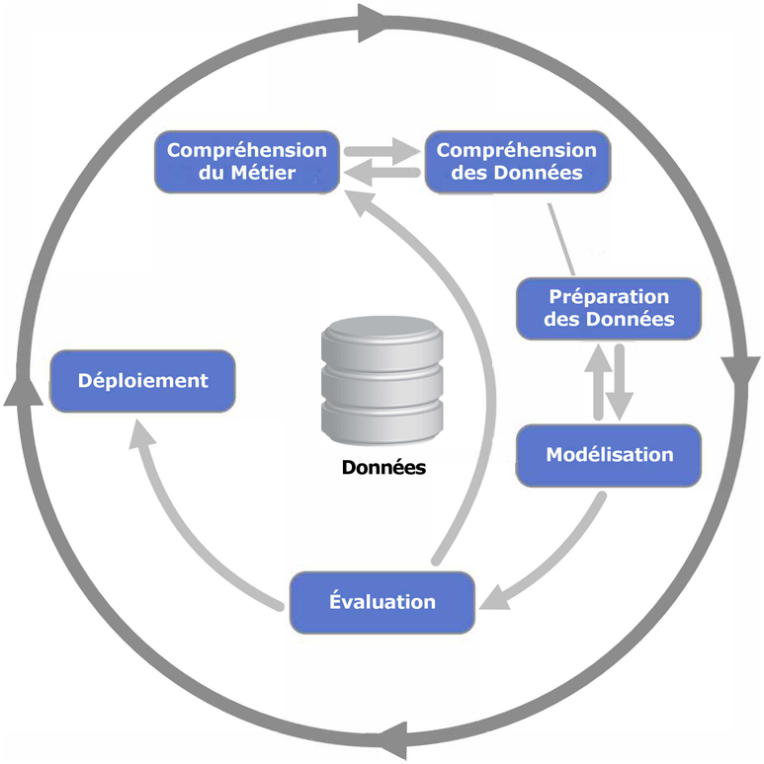


FIG. 1.4 : Diagramme de processus de CRISP-DM

1.5.3 Avantages De La Méthodologie CRISP-DM

La déclinaison de la méthodologie CRISP-DM dans un projet au sein d'une entreprise apporte :

- Une technique simple et accessible permettant aux membres de l'équipe sans connaissance en sciences des données de participer activement.
- Le mode itératif permet une intégration continue du projet.
- CRISP-DM est flexible, cette méthodologie résoudre des projets qui commence par des inconnues significatives, l'utilisateur peut parcourir les étapes, acquérant à chaque fois une compréhension plus approfondie des données et du problème.

1.6 Conclusion

Pour la réussite d'un projet Data Science, il est important de suivre une démarche agile et itérative comme la méthodologie CRISP-DM. c'est-'a-dire que chaque itération apporte de la connaissance métier supplémentaire qui permet de mieux aborder l'itération suivante.
C'est d'ailleurs dans ce chapitre nous avons effectué la première itération de notre projet qui est la compréhension du métier.
Dans cette phase nous avons réussir à définir le problème et identifier les objectifs métiers et les exigences de la solution.

Cette première étape est la plus difficile puisqu'elle va orienter toutes les autres étapes et conditionner la réussite de notre application.
Dans le chapitre suivant, nous allons présenter les notions de base que nous avons traiter au cours de notre projet.

Chapitre 2

État de l'art

2.1 Introduction

Afin de familiariser les notions de langage naturel de traitement, dans ce chapitre, nous allons introduire dans la première partie les méthodes d'extraction de texte à partir d'un PDF. Dans la deuxième nous allons définis NLP, ses domaines d'application, les techniques de prétraitement des données et les modèles et les méthodes de calcul de similarité. Et dans la dernière nous avons parlé des techniques de « Highlight » sur un PDF.

2.2 Lecture de PDF

2.2.1 Extraction de texte

Il existe deux méthodes d'extraction de texte à partir d'un PDF, une méthode dédiée pour les PDFs scannés avec la technique computer vision (OCR) et l'autre en utilisant les bibliothèques python d'extraction.

2.2.1.1 La Méthode OCR

"Le mot OCR (en anglais : optical character recognition) signifie reconnaissance optique de caractères ou reconnaissance de texte, une technologie qui vous permet de convertir différents types de documents tels que les documents papiers scannés, les fichiers PDF ou les photos numériques en fichiers modifiables et interrogeables."[2]
La figure 2.1 résume le principe de la méthode OCR qui consiste à transformer chaque page de document en image, puis extraire le texte à partir de l'image et l'enregistrer dans un fichier texte.

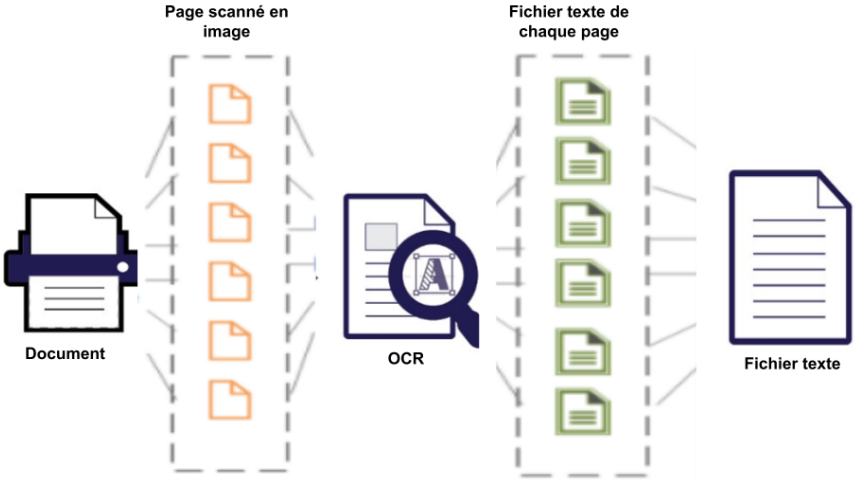


FIG. 2.1 : Principe de la méthode OCR

Cependant, l'OCR n'est pas la méthode idéale dans notre cas. il présente des inconvénients qui limitent ses applications :

- **Documents limités** : L'OCR fonctionne mieux avec des documents dactylographiés de bonne qualité. De plus, l'OCR peut ne pas fonctionner correctement s'il y a des polices tapées qui ressemblent à l'écriture manuscrite.
- **Précision** : l'OCR manque de précision. Les erreurs qui se produisent lors de la reconnaissance optique de caractères comprennent une lecture erronée des lettres, le saut de lettres illisibles ou le mélange de texte de colonnes adjacentes ou de légendes d'image.
- **Travail supplémentaire** : Une personne doit comparer manuellement le document original et le texte électronique et corriger les erreurs.

La figure ci dessous montre les inconvénients de l'extraction avec la méthode OCR.

FIG. 2.2 : Les inconvénients de la méthode OCR

2.2.1.2 Les bibliothèques d'extraction

Plusieurs librairies existent pour lecture des PDF sur Python, certaines n'arrivent pas à extraire correctement le texte s'il y a une mise en page ou un encodage particulier. Nous présentons brièvement les bibliothèques python les plus reconnues :

- **PyPDF2** : Est une bibliothèque Python impressionnante capable de lire des documents PDF et d'écrire du texte dans un fichier PDF. Cependant, elle ne fonctionne pas sur la langue française, parce qu' elle n'est pas conçue pour lire les caractères spéciaux latins.
- **PdfMiner** : Fonctionne sur les PDFs en français, cependant le temps de lecture est plus lent qu'avec PyPDF2 (une minute pour lire 300 pages contre une seconde pour les autres bibliothèques)
- **PdfLib** : Cette bibliothèque"peut s'avérer efficace et rapide mais peine sur certaines mises en forme. Notamment si le texte est présenté en colonnes, il sera quand même lu de gauche à droite, et renverra un résultat erroné." [5]
- **PyMupdf** : Est une liaison python de la bibliothèque MuPDF pour extraire le texte à partir des fichiers aux formats PDF, XPS, OpenXPS, CBZ (archives de bandes dessinées), FB2 et EPUB (e-book). MuPDF se distingue parmi tous les produits similaires par sa capacité de rendu supérieure, sa vitesse de traitement inégalée et elle fonctionne parfaitement sur les textes en français, quelle que soit la mise en page utilisée.
  C'est pourquoi nous avons choisi d'utiliser PyMuPdf. Il présente d'autres fonctionnalités : une fonction pour récupérer table des matières et une fonction pour highlighter un mot.

2.2.2 Extraction de tableaux

- **tabula-py et Camelot** : Ce sont deux bibliothèques python pour lire des tableaux dans un PDF et et les convertir au format CSV. Cependant, ces librairies fonctionnent seulement si le tableau est clairement dessiné avec des frontières.

- **CascadTabNet** : Est une méthode de reconnaissance automatique de table pour interpréter les données tabulaires dans les images de documents. C'est un modèle basé sur un réseau de neurones convolutionnels (CNN) à haute résolution basé sur une région de masque en cascade (R-CNN cascade mask HRNet), qui détecte les régions de table et identifie simultanément les cellules structurelles du corps. Deux types d'objets sont détectés : les tables avec bordures (bordered tables) et les tableaux sans bords (borderless tables). Pour l'utiliser, il faut d'abord convertir les pages PDF en images, puis appliquer le modèle, qui renvoie 2 probabilités : une pour chaque type de tableau. Le modèle renvoie également les coordonnées de la table détectée. Cette solution résolu le problème de détection de tableaux sans bordures, mais elle possède des inconvénients : Cascade TabNet est un projet Git et qu'il est plus lourd à installer que Tabula-py. Ainsi, ce modèle peut comparer deux tableaux situés dans le même emplacement d'une page PDF.

2.3 langage de traitement naturel

Le traitement Automatique du Langage Naturel (TALN) ou Natural Language Processing (NLP) en anglais est un des domaines de recherche les plus actifs en science des données Il s'agit d'une discipline qui porte essentiellement sur la compréhension, la manipulation et la gestion du langage naturel par les machines. Ainsi que, le NLP se présente comme le point d'intersection entre la science informatique et la linguistique.

2.3.1 Domaines de traitement automatique du langage naturel

Le NLP est un terme assez générique qui recouvre un champ d'application très vaste. Voici les applications les plus populaires :

- **L'analyse sentiment** : Connu également sous le nom d'Opinion Mining, il consiste à identifier les informations nécessaires d'un texte pour extraire l'opinion de l'auteur. Cette analyse permet d'exploiter les commentaires collectés sur un produit dans les réseaux sociaux et rechercher le sentiment négatif et positive pour mesurer le taux de satisfaction de client à l'égard des produits ou services fournis par une entreprise ou une organisation.

- **La traduction automatique de textes** : Est probablement l'un des domaines les plus populaires de NLP. Il existe de nos jours plusieurs applications de traduction automatique développés par des algorithmes révolutionnaires, telles que Google Translation, ces traducteurs sont capables de traduire des textes entiers sans aucune intervention humaine. Le processus de traduction se déroule en plusieurs phases successives : Compréhension et assimilation, puis ré-expression et reformulation dans la langue cible.

- **Résumé automatique** : Avec les méthodes récentes de NLP, la génération d'un résumé significatif, précis et fluide de texte à partir de plusieurs ressources textuelles est garanti.
  Il existe deux types principaux de résumés :

  - **Résumé par extraction** : Ce type de résumés consiste à extraire des phrases clés du texte d'origine sans modification puis les mettre les uns à la suite des autres pour créer le texte résumé.

  - **Résumé par abstraction** : Ces résumés conservent le sens du texte d'origine, mais le résumé est une reformulation de l'original. Ainsi, ce type utilise des techniques avancées de NLP pour générer un résumé entièrement nouveau tels que les modèles pré-entraînés basés sur l'architecture Transformer à titre d'exemple BERT.

- **La reconnaissance vocale** : "La reconnaissance vocale consiste à analyser de la voix humaine, afin de la transformer en texte. Tout passe par la voix, qui est identifiée puis captée en fréquences sonores (voice-to-text). Ensuite l'analyse de ces fichiers sonores, par les technologies du deep learning liées à l'intelligence artificielle."

[3]
Elle se déroule généralement de la façon suivante :

- La reconnaissance vocale : La machine détecte le voix et la convertis en texte.
- NLP : Pour que la machine peut analyser et comprendre les données, elle les transforme en données numériques avec NLP.
- La synthèse vocale : à l'aide des modèles Deep Learning spécifiques composées des réseaux de neurones, la machine transmise les données à l'utilisateur sous une forme sonore.

La figure 2.1 résume le déroulement de la reconnaissance vocale.

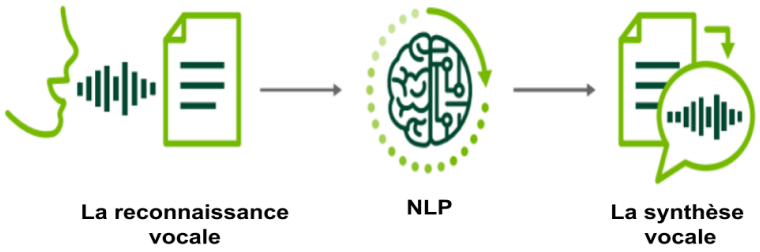


Fig. 2.3 : Les étapes de la reconnaissance vocale

- **La reconnaissance d'entités** : Un programme NLP doit reconnaître si un texte contient un nom propre de lieux, de personnes ou d'organisations et il doit pouvoir les associer au nom en question. Il permet également d'identifier qui a fait quoi à qui.

- **Système de Questions/Réponses** : Il s'agit de répondre de manière précise aux questions posées par les humains dans un langage naturel. La technologie est très utilisée aujourd'hui par des entreprises pour les « chatbots », les projets internes (RH, opérations) et externes (service client).

- **La classification de texte** : Également appelée catégorisation de texte, est le processus de balisage de texte en groupes organisés. En utilisant le traitement du langage naturel (NLP), les classificateurs de texte peuvent automatiquement analyser le texte, puis attribuer un ensemble de labels en fonction de son contenu.

- **La similarité textuelle** : La similarité de texte doit déterminer à quel point deux morceaux de texte sont proches à la fois en termes de proximité de surface et de sens c'est à dire la similarité lexicale et similarité sémantique. Dans ce contexte se situé notre sujet.

2.3.2 Prétraitmenet des données

Pour la réalisation d'une solution dans le domaines de la science des données, après l'étape de l'extraction et la collecte des données, il est nécessaire de les prétraiter. Ceci via des techniques de traitement automatique du langage. Nous présentons brièvement les techniques les plus importantes.

2.3.2.1 Tokenization

C'est l'opération la plus basique dans un processus de NLP. Toutes les phrases sont tokenisés, ce qui signifie que le texte de la page devient une liste de phrases.

2.3.2.2 Racinisation (stemming)

Le but du stemming est de réduire un mot dans sa forme « racine ». ce qui réduit le nombre des mots différents nécessaires pour représenter un document. Par exemple, une fois que l'on applique un stemming sur « million » ou « million », le mot résultant est le même.

2.3.2.3 Lemmatisation

La lemmatisation est le processus de regroupement des différentes formes fléchies d'un mot afin qu'elles puissent être analysées comme un seul élément. La lemmatisation est similaire à la radicalisation mais elle apporte un contexte aux mots. Ainsi, il relie des mots ayant des significations similaires à un seul mot. Il est généralement plus sophistiqué que les stemmers, car les stemmers travaillent sur un mot individuel sans connaître le contexte.

2.3.2.4 Suppression des mots vides (Stopwords)

La langue française est riche des mots vides comme « la, le, est, a, . . », leur présence n'affecte le plan sémantique ni le plan lexical et leur élimination réduit la dimension du documents en supprimant les mots insignifiants. Lorsque l'on effectue par exemple un taux de similarité entres des phrases par la méthode BERT, on souhaite limiter la quantité de mots dans les données d'entraînement.

2.3.2.5 Suppression du bruit

Les documents texte contiennent généralement des caractères tels que des caractères spéciaux qui ne sont pas nécessaires à l'exploitation du texte.

2.3.3 Les modèles de calcul de similarité

La similarité entre les documents est l'un des problèmes les plus curiaux de traitement automatique de langage. En effet elle est distingué à partir de mesure de similarité

sémantique entre les textes. Pour calculer ce taux il faut représenter le texte sous forme quantifiable par les modèles des incorporations de texte(représentations vectorielles du texte) suivantes :

2.3.3.1 Tf-idf

Il permet de mettre en relation le calcul de fréquence de mot dans un document et inversement c'est à dire la fréquence de documents contenant ce mot dans l'ensemble du corpus de documents.

2.3.3.2 Word2Vector

Il sert à rendre un corpus de texte en entrée et produit des incorporations de mots en sortie en produisant des représentation vectoriel.

2.3.3.3 GloVe

C'est un algorithme d'apprentissage non supervisé permettant à représenter les mots en vecteurs.

2.3.3.4 Doc2Vec

C'est un algorithme d'apprentissage non supervisé qui résult une représentation vectorielle de de phrases/paragraphes/documents.

2.3.3.5 BERT

Il s'agit d'une technologie de pointe développée par Google pour le pré-apprentissage du traitement du langage naturel. BERT est formé sur du texte non étiqueté, y compris Wikipedia et Book corpus. Il utilise Transformer Architecture, un modèle d'attention pour apprendre l'incorporation de mots.
Contrairement aux modèles directionnels, qui lisent le texte saisi de manière séquentielle (de gauche à droite ou de droite à gauche), l'encodeur Transformer lit la séquence entière de mots en une seule fois. En effet c'est un modèle bidirectionnelle composé de deux étapes pour sa pré-formation :

- **Modélisation du langage masqué (MLM)** : Consiste à remplacer les 15% des mots de chaque séquence par un jeton [MASQUE], puis le modèle tente de prédire la valeur d'origine des mots masqués tout en basant sur le contexte fourni par les autres mots non masqués de la séquence.
  La prédiction des mots de sortie nécessite :

  1. Ajout d'une couche de classification au-dessus de la sortie de l'encodeur.
  2. Multiplier les vecteurs de sortie par la matrice d'intégration, en les transformant en dimension de vocabulaire.

3. Calculer la probabilité de chaque mot du vocabulaire avec softmax.

- **Prédiction de la phrase suivante (NSP)** : Dans cette phase, le modèle apprend à prédire si la deuxième phrase de la paire en entrée est la phrase suivante dans le document d'origine.
  Pour aider le modèle à distinguer les deux phrases en formation, l'entrée est traitée de la manière suivante avant d'entrer dans le modèle :

  1. Un jeton [CLS] est inséré au début de la première phrase et un jeton [SEP] est inséré à la fin de chaque phrase.
  2. Une phrase incorporée indiquant Phrase A ou Phrase B est ajoutée à chaque jeton. Les incorporations de phrases sont similaires dans leur concept aux incorporations de jetons avec un vocabulaire de 2.
  3. Une intégration positionnelle est ajoutée à chaque jeton pour indiquer sa position dans la séquence. Le concept et la mise en œuvre de l'intégration positionnelle sont présentés dans l'article Transformer. Les étapes de pré-formation sont résumés dans la figure suivante :

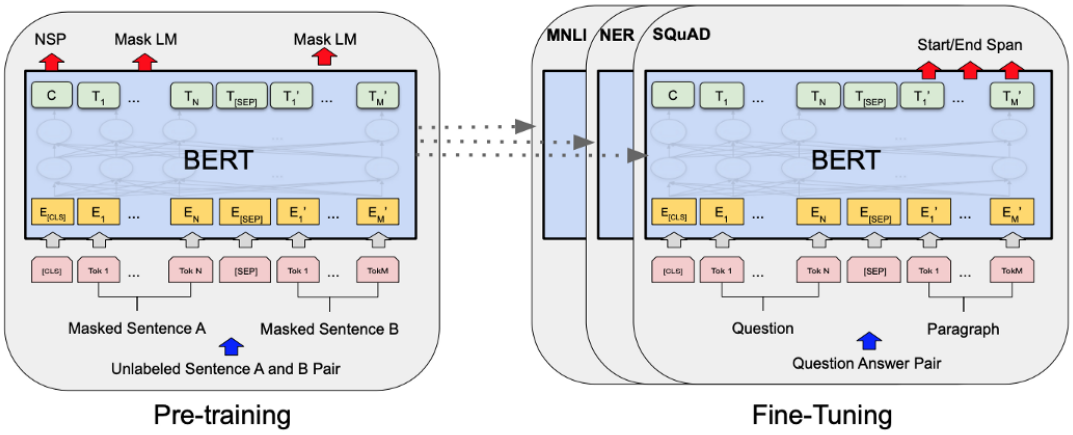


Fig. 2.4 : Étape de pré-formation de BERT

2.3.3.6 SBERT

Avant les transformateurs de phrases, l'approche pour calculer la similarité précise des phrases avec **BERT** consistait à utiliser une structure de codeurs croisés. Cela signifiait que nous transmettrions deux phrases au modèle, ajouterions une tête de classification en haut de BERT - et l'utiliserions pour produire un score de similarité. Ce réseau de codeurs croisés produit des scores de similarité très précis, mais il n'est pas évolutif, c'est à dire on a besoins d'effectuer le calcul d'inférence entre codeur 100 000 fois si on veut effectuer une recherche de similarité dans un petit ensemble de données de 100 000 phrases.
La solution : L'introduction de phrase-BERT(SBERT) et de la sentence-transformers bibliothèque proposé par Nils Reimers et Iryna Gurevych en 2019 pour que modèle précis BERT peut avoir une latence raisonnable. L'évolutivité de SBERT permet à produire des

incorporations de phrases, sans besoins d'effectuer un calcul d'inférence complète pour comparer chaque paire de phrases.

SBERT est similaire à Bert qui utilise l'architecture de codeur croisé mais supprime la tête de classification finale, et traite une phrase à la fois. SBERT utilise ensuite la mise en commun des moyennes sur la couche de sortie finale pour produire un encastrement de la phrase.

La comparaison entre le BERT et SBERT est illustré dans le figure ci dessous :

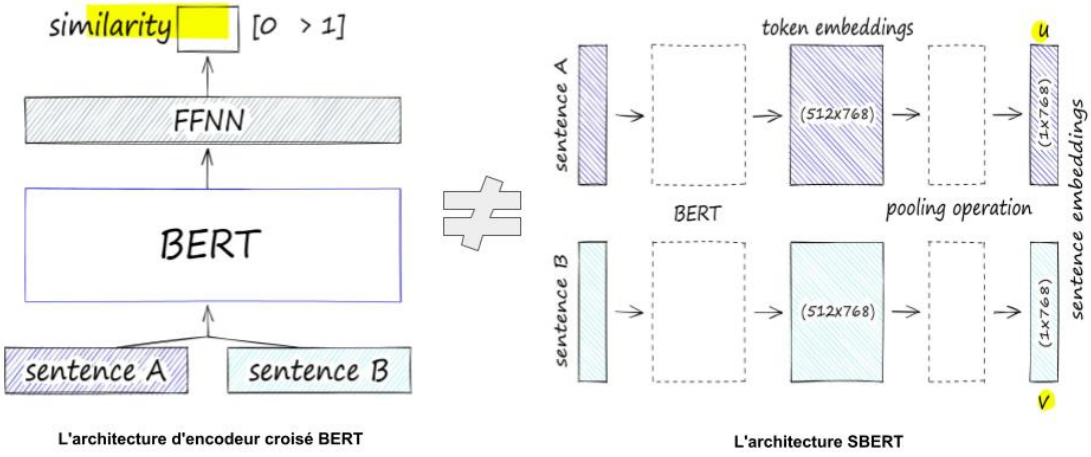


Fig. 2.5 : Comparaison entre Modèle BERT et SBERT

2.3.4 Les méthodes de calcul

Après avoir les incorporations de texte pour que les machines déterminent la similitude avec les fonctions de calcul comme :

- **Cosinus Distance/Similarité** :C'est la distance angulaire entre les vecteurs c'est à dire le cosinus de l'angle entre les deux vecteurs produites. La formule de calcul est définie par :

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Fig. 2.6 : Formule de Cosinus Distance/Similarité

- **Distance euclidienne** :C'est une formule qui permet de calculer la distance entre deux points dans l'espace Euclidien. Il est définit comme suit ,

$$d(\boldsymbol{a}, \boldsymbol{b}) = d(\boldsymbol{b}, \boldsymbol{a}) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2}$$

FIG. 2.7 : Formule de Cosinus Euclidienne

2.4 Les méthodes de « Highlight » dans un documents

Pour le « Highlight » un mot dans un emplacement bien déterminé sur le document PDF, nous avons présenté les méthodes suivante :

2.4.1 Bounding Box

Est une boîte rectangulaire imaginaire qui contient un objet ou un ensemble de points. Lorsqu'elle est utilisée dans le traitement d'images numériques, la boîte englobante fait référence aux coordonnées de la bordure qui entourent une image. Ils sont souvent utilisés pour lier ou identifier une cible et servir de point de référence pour la détection d'objet et créer une boîte de collision pour cet objet.
La figure ci dessous montre le processus de surlignage avec le bounding box :

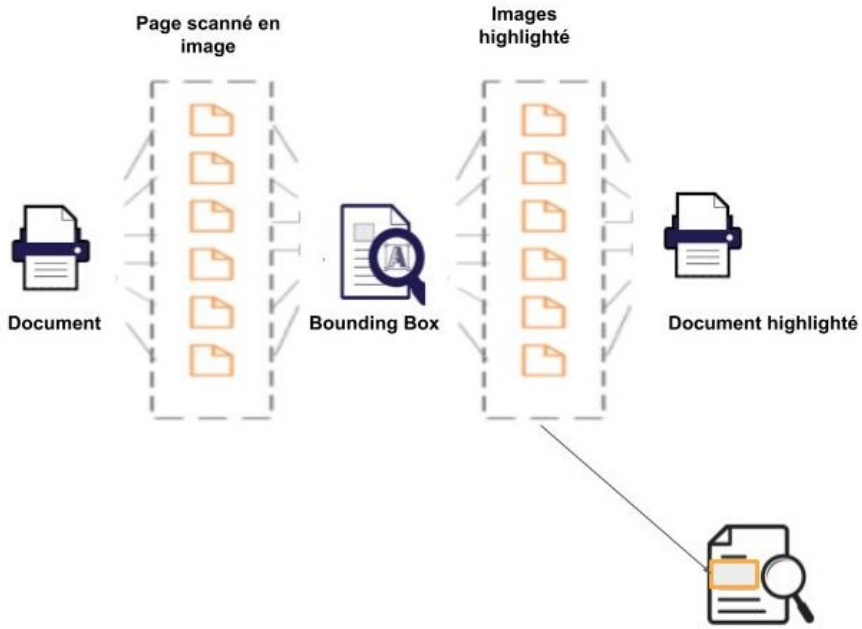


FIG. 2.8 : Démonstration de méthode de bounding box

2.4.2 Les fonctions de la bibliothèque PyMupdf

- **SearchFor** : C'est une fonction python de la bibliothèque PyMupdf qui permet de chercher un text donné en paramètre et retourne ces coordonnées.

- **addHighlightAnnot** : C'est une fonction python de la bibliothèque PyMupdf qui permet de surligner un texte avec ces coordonnées donnée en paramètre à une page bien déterminé.

2.5 Conclusion

Une fois que nous avons étudie notre art, nous allons passer aux étapes de la compréhension et la préparation de données de notre méthodologie CRISP-DM.

Chapitre 3

Compréhension et préparation des données

3.1 Introduction

Dans ce chapitre nous allons présenter la deuxième et la troisième phase de la méthodologie CRISP-DM. Cette étape a une grande importance car elle nous permet non seulement de comprendre, d'explorer et de préparer nos données, mais aussi d'éviter les problèmes imprévus durant les phases suivantes.

3.2 Compréhension des données

Comme nous avons indiqué dans la première phase, notre projet consiste à la mise en place d'une platforme qui permet le « matching » entre deux documents financières en deduissant les écarts. La source des données est donc, d'une part, les deux documents PDF entrés afin d'être analysés ; et d'autre part, les données fournies par l'auditeur pour ré-entrainer notre modèle BERT avec les vocabulaires dans le domaine de comptabilité.

3.2.1 Description des données

3.2.1.1 Données de PDF

Afin de faire un pipeline de solutions, la première étape est de savoir quelles sont les données et quelles sont ses différentes caractéristiques. Dans un document, il existe de nombreux types de pages, mais généralement, celles-ci peuvent être classées en trois types :

- Structuré | Formulaires et modèles cohérents
- Non structuré | Texte, pas de mise en forme et tableaux
- Semi-structuré | Hybride des deux ci-dessus, peut avoir une structure partielle

Généralement les documents financières sont rédigés par la même façon, c'est pour cela nous avons choisi deux documents 'Rapport Annuel AKWEL 2018' et 'Rapport Annuel AKWEL 2020' pour élaborer une description de rapports financières.
En effet ils sont des données non-structurés avec des types de pages Structuré,Non structuré et Semi-structuré, chaque page peut être composé de tableaux ou de texte ou la combinaison entre texte, images et tableaux.

- **Description de texte** : A partir de format de texte nous avons distingué la différence entre les titres et les paragraphes. En effet les titres se différencient par leur style ou par les numérotations situé dans la page de sommaire ou bien dans le reste de document.
- **Description de tableaux** : Il existe deux types de tableaux tels que, les tableaux avec bordures et d'autres sans bordures.
- **Description des images** : Les images sont rarement apparu dans les documents de type financiers, s'ils existent leur importance dans le « matching » est assez limités.

Les documents financières d'une entreprise d'une année à une autre se diffèrent par :

- le nombre de page,
- la structure : ils ne sont pas dans un ordre cohérent. Par exemple, dans une section, le document **A** peut venir après le document **B** et dans l'autre, c'est l'inverse,
- le nombre et le contenu de section.

3.3 Préparation des données

3.3.1 Extraction de text

Comme nous avons mentionné dans le chapitre précédent, il y a plusieurs bibliothèques python pour lire un document PDF. Mais nous avons choisi MuPDF puisqu'elle se distingue par sa vitesse et la fonction de « highlight » qu'elle est nécessaire pour les autres étapes.
La première étape de notre solution consiste à extraire le contenu et le numéro de page de chaque page PDF, et à ajouter séquentiellement l'index de la page PDF sous forme (Page.numPage.), ainsi que les lignes de textes correspondantes. La figure ci-dessous montre des lignes de commande pour retourner une liste de contenu de PDF.

```
In [3]: import fitz
        doc=fitz.open("C:/files/AKWEL2.pdf")
        listfile=[]
        for i  in range(len(doc)) :
                ch="Page."+str(i)+"."
                text = doc[i].get_text("text")
                listfile.append(ch)
                listfile=listfile+(text.split('\n'))
        print(listfile)
```

```
u COUTIER ', 'Président du Directoire ', ' ', '', 'Page.3.', 'BROUILLON  ', ' ', ' ', 'AKWEL ', 'RAP
PORT ANNUEL ', '_____ ', ' ', ' ', '2020 ', '4/177 ', 'ADMINISTRATION, DIRECTION ET CONTROLE
', '1. Conseil de surveillance ', 'André COUTIER ', 'Président du Conseil de surveillance  ', 'Nicol
as JOB ', 'Vice-président du Conseil de surveillance ', 'Geneviève COUTIER ', 'Membre  ', 'Emilie CO
UTIER ', 'Membre  ', 'COUTIER DEVELOPPEMENT  ', 'représentée par Christophe COUTIER ', 'Membre ', 'C
hristophe BESSE (*) ', 'Membre  ', 'Anne VIGNAT DUCRET ', 'Membre ', '(*) Membre élu par les salarié
s. ', "Vous trouverez dans le Rapport du Conseil de surveillance sur le gouvernement d'entreprise in
tégrant les ", "observations du Conseil de surveillance sur le rapport de gestion et sur les comptes
de l'exercice, les informations ", 'indiquant leur âge, leur qualité d'indépendant, de membre des Co
mités d'audit et des rémunérations, la date ', 'd'expiration de leurs mandats exercés au sein de la
Société ainsi que les fonctions et mandats exercés dans ', 'd'autres sociétés, cotées ou non. ', '2.
Directoire ', 'Mathieu COUTIER ', 'Président du Directoire  ', 'Jean-Louis THOMASSET ', 'Vice-présid
```

Fig. 3.1 : Extraction texte avec la bibliothèque MuPDF

La liste retournée dans la figure ci-dessus nécessite la suppression d'éléments vides et d'exclure le mot 'BROUILLON' accompagné parfois avec l'extraction de page. Ce mot indique le début et la fin de chaque page, c'est une résultat de la mal-extraction.
Dans le processus de fractionnement du texte en phrases complètes, nous utilisons les sauts de ligne comme référence de segmentation. Mais cette méthode nous donne des phrases incomplètes, ce qui nous amène à utiliser des jointures pour obtenir des phrases complètes.

3.3.2 Extraction des tableaux

Comme nous avons indiqué, il existe des bibliothèques pour extraire des tableaux mais chacune d'eux manque des fonctions pour accomplir notre solution. Par exemple, tabula Py et camelot ne détectent pas les tableaux sans bordures, cependant, ce type de tableau est très courant dans notre base de données PDF. Nous pouvons aussi utiliser la modèle cascade TabNet puisqu'il l'un des tableaux à comparer peut diviser sur deux pages de PDF, et l'autre dans une seule page.
Par conséquent, nous avons trouvé d'autre solution qui permet d'identifier dans chaque ligne la première case du tableau à comparer qui doit être identique à celle dans le PDF2 puis détecter les autres cases dans le même ligne, finalement nous avons comparé ces cases détectées.

3.3.3 Extraction de titres

Il existe deux méthodes pour détecter la table de matière :
- Extraction à partir de métadonnées de PDF, ceci par la fonction proposée par PyMuPDF pour récupérer table des matières qui retourne pour chaque entrée le titre et le numéro de

page. Mais cette solution est limitée sur les PDFS écrites en métadonnées. C'est pourquoi nous avons éliminé cette solution.
- Bien que l'aspect des tables des matières varie d'un document à l'autre, il existe des patterns. En effet, nous avons extrait le sommaire par certaines expressions comme "Sommaire", "Table des matières", "Plan" etc.

Pour détecter le contenu de sommaire, le point de début est le premier titre et point de fin est la première apparition de ce titre. Mais, dans certains documents le sommaire n'est pas complet, il y a des sous-titres manquants. Nous pouvons donc facilement utiliser des expressions régulières, et éventuellement un simple classificateur, pour identifier les phrases à comparer. Les titres nous permettent de filtrer les pages qui nous intéressent. Même si nos PDF ont une structure différente, il est facile d'identifier la partie qui nous intéresse à l'aide de certains mots-clés et il suffit ensuite de parcourir la page cible. Après l'extraction de deux types de titres : les titres de sommaire et les sous-titres, nous avons préparé une liste contenant tout les tires ordonnés selon leur apparition dans le document.

Ainsi, nous avons remarqué des sous-titres qui se répètent dans des sections différents. Par conséquent, nous avons réaliser une combinaison de grand titre et ses sous-titres.

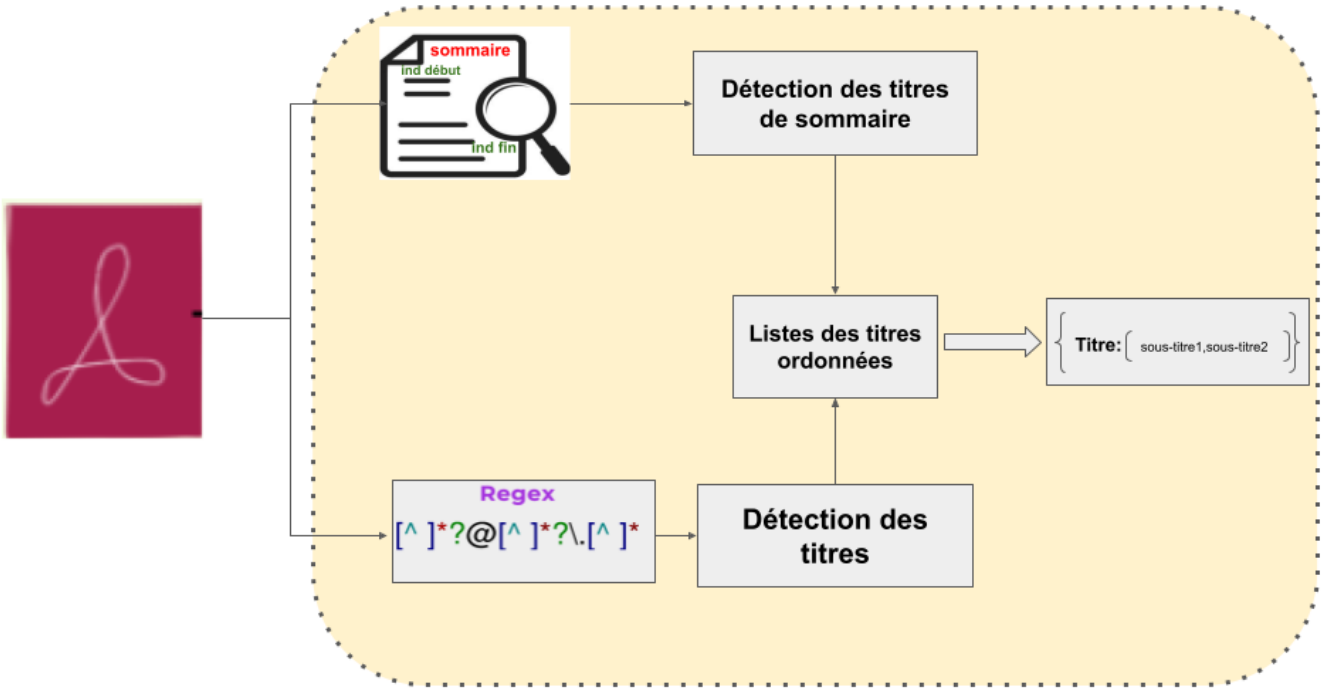


Fig. 3.2 : Extraction des titres

3.3.4 Extraction le contenu de chaque titre

Nous avons extrait le contenu de chaque titre avec la considération de titre actuel comme indice début jusqu'à l'apparition de titre suivant qui est l'indice fin.
Donc le résultat de l'extraction est illustré par l'apparition de chaque titre avec son contenu. Donc l'output final est la combinaison [Titre,paragraphe,numéro de page].

3.3.5 Pré-traitement de données

La phase de pré-traitement de données est une phase primordiale pour effectuer la comparaison entre les paires de phrases efficacement, en effet elle sert à standardiser le texte afin de rendre son usage plus facile
La figure 3.3 présente le processus de pré-traitement des données textuelles :

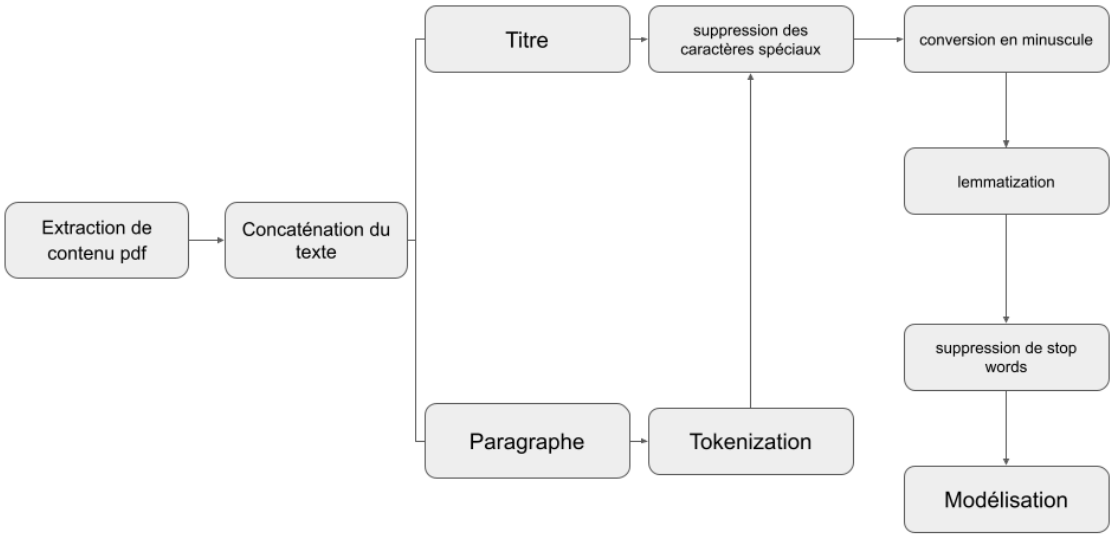


Fig. 3.3 : Processus de pré-traitement des données textuelles

3.4 Conclusion

Dans ce chapitre, nous avons commencé par la réalisation une étude approfondie sur nos données ; d'abord la description de nos documents puis la préparation des données nécessaires pour notre solution (titre, texte,tableau) et nous avons fini par la présentation des différentes étapes de préparation et de nettoyage des données.
Il est le temps maintenant d'exécuter la quatrième phase de CRISP-DM.

Chapitre 4

Modélisation et évaluation

4.1 Introduction

4.2 Modélisation

Dans cette section nous allons discuté d'abord le modèle choisi et les étapes a suivre sur un ensemble de données pour produire les phrases les plus similaires.

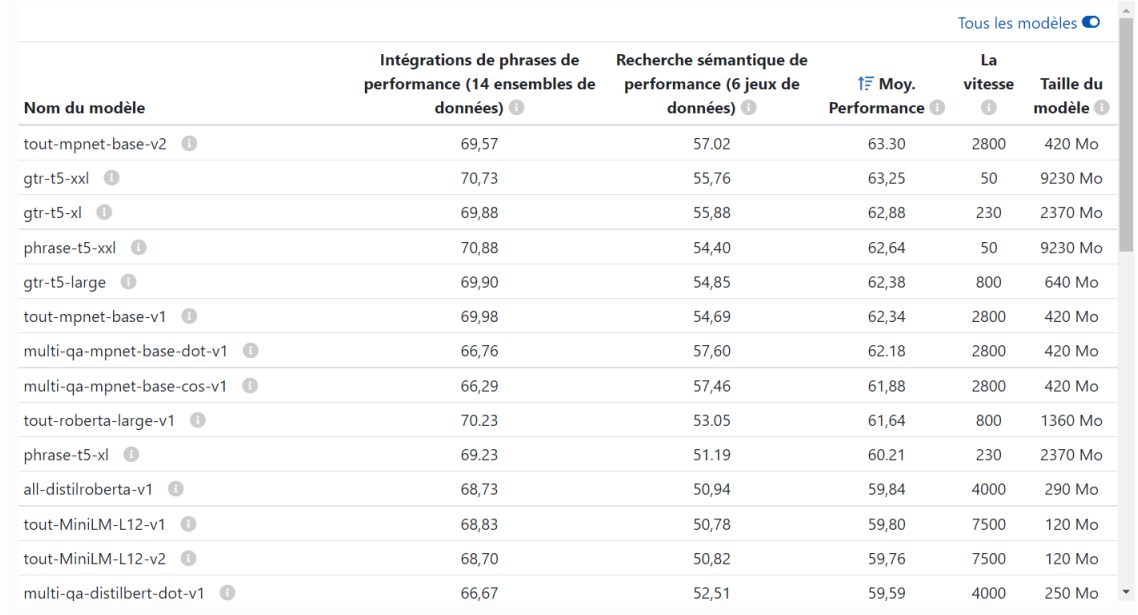
4.2.1 Le choix de modèle

Comme nous avons parlé dans le chapitre 2 : étude de l'art, il existe plusieurs modèles de calcul de similarité textuelle. Nous avons alors choisi le modèle BERT, plus précisément, SBERT qui est basé sur les sentences-transformers.
SBERT se diffère par :

- Sa performance en terme de résultats.
- Sa performance en terme de temps.
- Réduire la complexité de calcul contrairement au modèle BERT.

Il existe plusieurs modèles de sentences-transformers chacune se diffère par ensemble de données, performance , vitesse et sa taille. Il existe aujourd'hui le 17.04.2022 435 modèles basé sur les sentences-transformers.

La figures ci-dessous illustre des exemples de modèles et ses caractéristiques.



Tous les modèles

Nom du modèle	Intégrations de phrases de performance (14 ensembles de données)	Recherche sémantique de performance (6 jeux de données)	Moy. Performance	La vitesse	Taille du modèle
tout-mpnet-base-v2	69,57	57.02	63.30	2800	420 Mo
gtr-t5-xxl	70,73	55,76	63,25	50	9230 Mo
gtr-t5-xl	69,88	55,88	62,88	230	2370 Mo
phrase-t5-xxl	70,88	54,40	62,64	50	9230 Mo
gtr-t5-large	69,90	54,85	62,38	800	640 Mo
tout-mpnet-base-v1	69,98	54,69	62,34	2800	420 Mo
multi-qa-mpnet-base-dot-v1	66,76	57,60	62.18	2800	420 Mo
multi-qa-mpnet-base-cos-v1	66,29	57,46	61,88	2800	420 Mo
tout-roberta-large-v1	70.23	53.05	61,64	800	1360 Mo
phrase-t5-xl	69.23	51.19	60.21	230	2370 Mo
all-distilroberta-v1	68,73	50,94	59,84	4000	290 Mo
tout-MiniLM-L12-v1	68,83	50,78	59,80	7500	120 Mo
tout-MiniLM-L12-v2	68,70	50,82	59,76	7500	120 Mo
multi-qa-distilbert-dot-v1	66,67	52,51	59,59	4000	250 Mo

FIG. 4.1 : Exemple des modèles et ses caractéristiques

Prenons l'exemple de modèle RoBERTa, DistilBERT et distilRoBERTa :

- **RoBERTa** : est un recyclage du BERT avec une méthodologie de entraînement amélioré, 1000 % de données en plus et une puissance de calcul. En effet, RoBERTa exclue la tâche de prédiction de la prochaine phrase (NSP) de la pré-entraînement de BERT et introduit un masquage dynamique afin que le jeton masqué change pendant les époques de formation. Il a également amélioré le réglage des hyperparamètres pour BERT. De plus, en s'entraînant avec des mini-lots et des taux d'apprentissage beaucoup plus importants, RoBERTa est capable d'améliorer l'objectif de modélisation du langage masqué par rapport à BERT. Cela permet d'aboutir à de meilleures performances sur un grand nombre de tâches NLP.

- **DistilBERT** : une version distillée (approximative) de BERT, conservant 97% de performances mais n'utilisant que la moitié du nombre de paramètres. Plus précisément, il n'a pas d'incorporations de type jeton et ne conserve que la moitié des couches du BERT de Google. Ce modèle utilise une technique appelée distillation, qui se rapproche du BERT de Google, c'est-à-dire après l'entraînement d'un grand réseau de neurones, le résultat de ses distributions de sortie complètes peuvent être approximées à l'aide d'un réseau plus petit.

- **distilRoBERTa** : est une version distillée du modèle à base de RoBERTa. Il suit la même procédure de entraînement que DistilBERT. Le modèle a 6 couches, 768 dimensions et 12 têtes, totalisant 82 millions de paramètres.

La le tableau 4.2 montre une comparaison entre bert et RoBERTa, DistilBERT et distilRoBERTa.

	BERT	**RoBERTa**	**DistilBERT**
Taille	Base : 110 Large :340	Base : 110 Large :340	Base : 66
Durée	Base : 8*V100*12 jours Large : 4 jours avec 4 pods TPU	Large 1024*V100*1jour 4-5 durée plus que BERT	Base 8*V100*3.5 jours 4 durée moins que bert
Performance		2-20% amélioration	3% dégradation depuis bert
Données	16GB bert data 3.3 billion mot	160GB	16GB bert data 3.3 billion mot
Méthode	BERT avec MLM & NSP	avec MLM seulement	bert distilation

Tab. 4.2 : Comparaison entre BERT et ses améliorations récentes (RoBERTa, DistilBERT)

Nous avons alors choisi le modèle distilRoBERta puisqui'il est une combinaison de Roberta et distilBERT, c'est à dire, il améliore à la fois la performance et le vitesse d'inférence.

4.2.2 Processus d'organisation les phrases les plus similaires

Pour distinguer les similitudes entre les phrases, qui peuvent être des titres ou des paragraphes sous des titres similaires. En fait, nous avons suivi les étapes ci-dessous, en divisant d'abord la liste des phrases en deux sous-ensembles, l'un contenant des phrases similaires et l'autre contenant des phrases différentes, sur la base de l'application de l'opérateur logique "==" est la similarité sans BERT. Nous utilisons ensuite le modèle BERT formé pour mesurer la similarité sémantique entre les paires de phrases en utilisant le résultat de la liste différente.

Les étapes de distinction des phrases similaires (sans et avec BERT) est illustré dans la figure suivante :

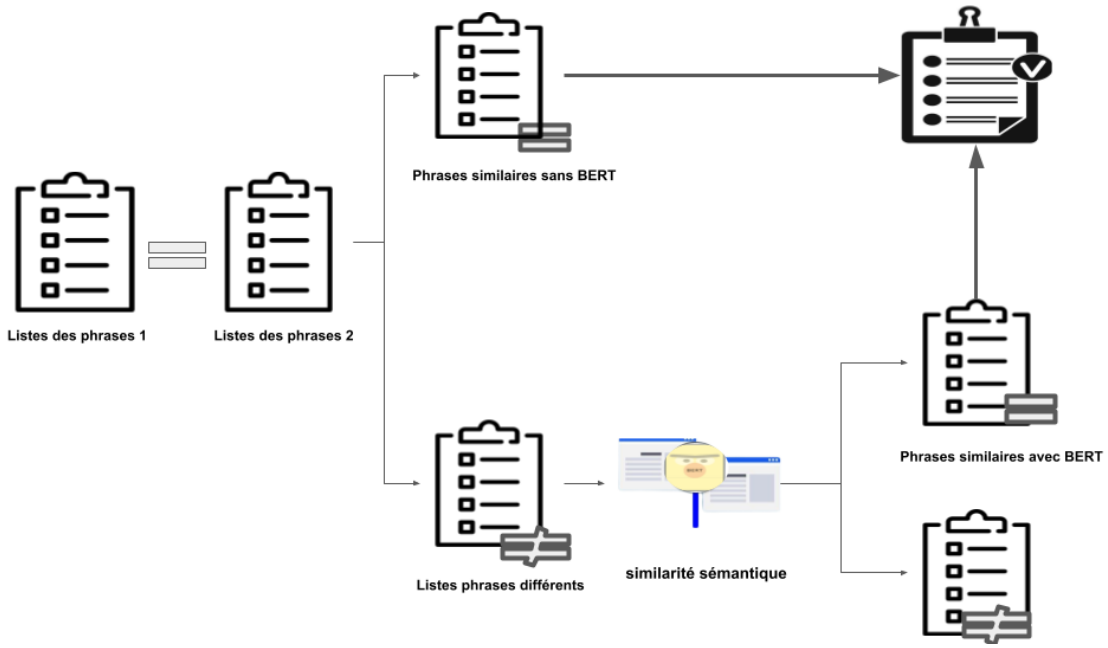


FIG. 4.2 : Processus de distinction des phrases similaires(sans et avec BERT)

- **Étapes pour distinguer des phrases similaires avec BERT**
  En vue de traiter les phrases les plus similaires sur un ensemble de données nous devons appliquer les étapes suivantes après l'étape de pré-traitement des données :

  1. Enregistre l'ensemble des phrases sans non pré-traités dans un DataFrame sous un nom de colonne choisi.
  2. Enregistre l'ensemble des phrases pré-traités dans la même DataFrame sous un autre nom de colonne.
  3. Appeler notre modèle sentence-transformers.
  4. Convertir les phrases pré-traités en vecteurs, afin de réduire la complexité des documents et de faciliter leur manipulation. Cela constitue un avantage de ce type de représentations pour nos traitements surtout lorsque les représentations utilisées tiennent compte de l'éloignement sémantique et linguistique existant entre les textes.
  5. Calculer la similarité cosinus entre chacune des phrases.
  6. Sauvegarder le tuple (phrase 1, phrase 2, taux de similarité) dans une liste de tuples.
  7. Filtrer les tuples qui ont un taux de similarité inférieur à 0.94.
  8. Prend le maximum de taux de similaire pour chaque paire de phrases.

  La figure ci-dessous contient les lignes de commandes de 5 premières étapes mentionnés.

  Le deux figures ci-dessous illustrent successivement les résultats de l'exécution de l'incorporation de texte en vecteur et le calcul cosinus similarité.

```
sentences = [
    "analyse des comptes consolidés 2020",
    "Le taux de valeur ajoutée s'élève à 40,9 % du chiffre d'affaires consolidé contre 42,4 % en 2017",
    "Le taux de valeur ajoutée s'élève à 45,6 % du chiffre d'affaires consolidé contre 40,5 % en 2019.
    "mise en place d'un protocole de nettoyage et de désinfection des sites",
    "produire un protocole de nettoyage"
]
documents_df=pd.DataFrame(sentences,columns=['documents'])#Etape 1
documents_df['documents_cleaned']=documents_df.documents.apply(lambda x:preprocessing_data(x))#Etape 2
sbert_model = SentenceTransformer(model_save_path)#Etape 3
document_embeddings = sbert_model.encode(documents_df['documents_cleaned'])#Etape 4
pairwise_similarities=cosine_similarity(document_embeddings)#Etape 5
```

FIG. 4.3 : Les étapes de détecter les phrases les plus similaires

```
print(document_embeddings)
```

```
[[ 0.28731737  0.29587993  0.04347432 ... -0.46074945  0.50697875
   0.24473612]
 [ 0.18512511 -0.41703367  0.36771283 ...  0.27903014  0.10276275
   0.3928574 ]
 [ 0.18512511 -0.41703367  0.36771283 ...  0.27903014  0.10276275
   0.3928574 ]
 [ 0.0975979  -0.04371179 -0.16661416 ... -0.19086021  0.64947003
   0.00588473]
 [ 0.28041914 -0.16977409 -0.05053229 ...  0.16909765  0.47537753
  -0.11872403]]
```

FIG. 4.4 : Transformation phrases en vecteurs

```
print(pairwise_similarities)
```

```
[[0.9999998  0.55155843 0.55155843 0.5495258  0.4502511 ]
 [0.55155843 0.9999998  0.9999998  0.6671032  0.59691155]
 [0.55155843 0.9999998  0.9999998  0.6671032  0.59691155]
 [0.5495258  0.6671032  0.6671032  1.         0.7915198 ]
 [0.4502511  0.59691155 0.59691155 0.7915198  0.9999997 ]]
```

FIG. 4.5 : Calcul Cosinus similarité

4.3 Évaluation de modèle par l'auditeur

Dans cette étape, nous devons évaluer le modèle choisi. L'évaluation par l'auditeur est une évaluation humaine de la faisabilité, de la performance du modèle dans la reconnaissance de phrases sémantiquement similaires. Cette phase comprend l'octroi aux utilisateurs du droit de critiquer le modèle et de corriger les résultats de correspondance en donnant les paires de phrases avec leur taux de similarité.
La figure ci-dessous montre la possibilité de l'utilisateur à évaluer le modèle :

Fig. 4.6 : Évaluation de modèle par l'auditeur

4.4 Ré-entraînement de modèle

Après l'étape de l'évaluation de résultats de correspondance entre les phrases de deux documents extraits. Nous avons besoins d'améliorer la performance et le résultat de modèle utilisé .
Pour la phase d'amélioration nous devons définir la notion de réglage fin qui est le principe de ré-entaînement :

- **Définition** : Réglage fin(Fine-Tune) est défini par l'utilisation d'un modèle pré-formé sur un énorme jeu de données comme point de départ. Nous pouvons ensuite former davantage le modèle sur un ensemble de données relativement petit.

Dans notre cas, nous avons utilisé le modèle pré-formé 'nli-distilroberta-base-v2', qui est affiné sur d'autres base de données et nous avons poursuit la formation sur une base de données personnalisé. Cette tâche nécessite Préparation et découpage de données et l'entraînement de modèle :

4.4.1 Préparation de données

Cette étape est primordiale pour le ré-entraînement de modèle. Commençant par la compréhension de dataset utilisé dans le modèle pré-entraîné. En effet le modèle utilisé 'nli-distilroberta-nli-distilroberta-base-v2' est entraîné avec les sources de données comme MNLI,SNLI,XNLI,STSBENCHMARK et d'autres.
La figure ci-dessous montre la structure d'une des datasets utilisé XNLI :

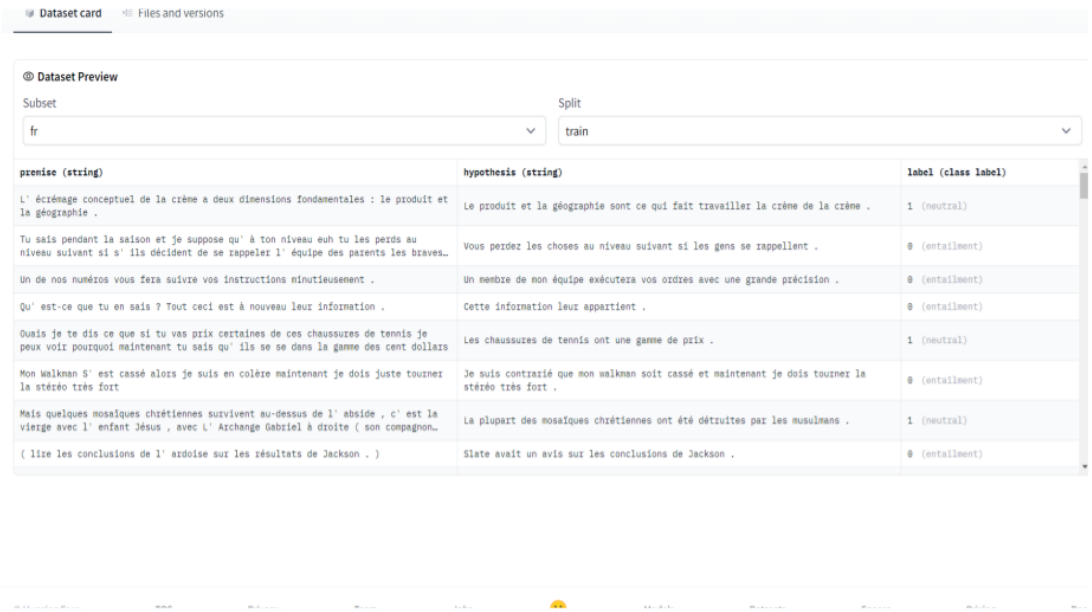


Fig. 4.7 : Compréhension de données de modèle pré-entraîné

Après la compréhension de données utilisés dans le modèle, nous avons commencé à préparer le data pour le nouveau modèle. En effet nous avons créer une base de données postgresql contenant une table composé des colonnes sentence1, sentence2 et label(taux de similarité entre les deux phrase compris entre 0 et 1).
La base de données créées va être remplie suite à l'évaluation de l'auditeur de résultat de matching. Après nous avons exploité l'avis de l'auditeur pour améliorer le modèle, nous avons considéré cette étape comme un boucle de feedback.
Le format de la base de données de Fine-Tune est illustré dans la figure ci-dessous :

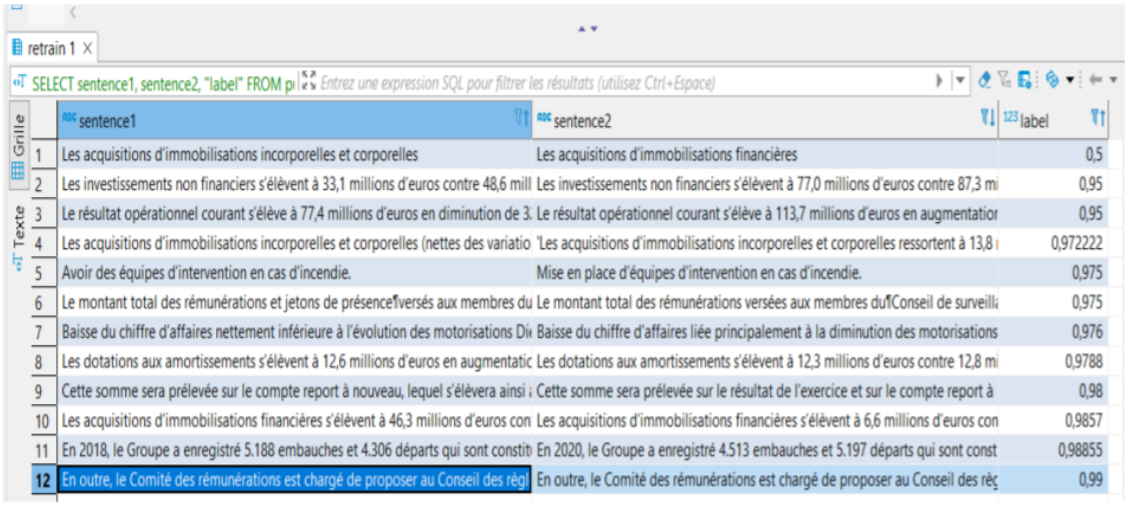


Fig. 4.8 : Dataset de Fine-Tune

4.4.2 Construction de modèle

4.4.2.1 Découpage de données

Avant de commencer à créer le modèle, nous devons diviser les données en ensembles d'entraînement, de test et de validation. Dans notre exemple, nous avons spécifié 60% de l'ensemble de données d'entraînement, 20% pour la validation et utilisé les 20% restants pour les tests. Pour ce faire, nous utilisons la fonction split() de la bibliothèque numpy, avec [int(.6*len(df)), int(.8*len(df))] un indices_or_sections tableau . Nous obtenons l'ensemble d'apprentissage train_samples,l'ensemble de validation dev_samples et le l'ensemble de test test_samples.

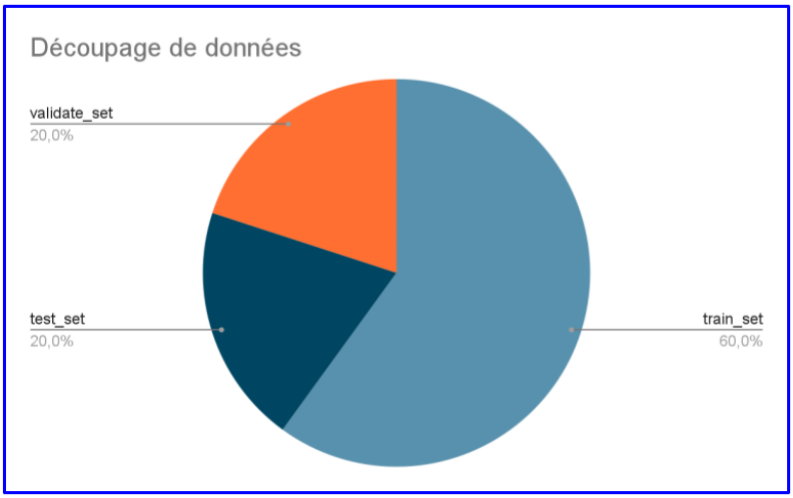


FIG. 4.9 : Découpage de données

4.4.2.2 Chargement de données

- **Structuration de données** Après avoir le data divisé en 3 ensembles, nous avons utilisé une classe prédéfini **InputExample** par l'instanciation d'un objet de type ce classe avec deux attributs en entrée d'une part **texts** qui est une liste contenant le paire de phrases, et d'autre part **label** qui est le label dans le dataset construit précédemment. Cette instanciation est primordiale pour que le format de la base de données devient exploitable dans l'ancien modèle et produit le modèle affiné.
- **Encapsulation de données** : train_dataloader est une résultat de l'encapsulation en utilisant la classe **DataLoader** pour encapsuler le jeu de données de traitement et permet de les requêter de diverses manières en spécifiant la taille de batch avec l'attribut **batch_size** et ordre aléatoire en spécifiant **True** à l'attribut **Shuffle**.

```
Entrée [ ]: #Etape1
for i in range(len(train_samples)):
  input=InputExample(texts=[train_samples['sentence1'].iloc[i],train_samples['sentence2'].iloc[i]],
                     label=float(train_samples['label'].iloc[i]))
  TrainExample.append(input)
TestExample=[]
for i in range(len(test_samples)):
   input=InputExample(texts=[test_samples['sentence1'].iloc[i],test_samples['sentence2'].iloc[i]],
                      label=test_samples['label'].iloc[i])
   TestExample.append(input)
DevExample=[]
for i in range(len(dev_samples)):
   input=InputExample(texts=[dev_samples['sentence1'].iloc[i],dev_samples['sentence2'].iloc[i]],
                      label=float(dev_samples['label'].iloc[i]))
   DevExample.append(input)
#Etape2
train_dataloader = DataLoader(TrainExample, shuffle=True, batch_size=train_batch_size)
```

FIG. 4.10 : Chargement de données

4.4.2.3 Préparation des évaluateurs de modèle

: Pour évaluer le modèle on a certaines fonctions qui retourne un taux pour mesurer la performance de modèle formé.

- **evaluator** : Un évaluateur (sentence_transformers.evaluation) évalue les performances du modèle pendant la formation sur les données de développement retenues. Il est utilisé pour déterminer le meilleur modèle qui est enregistré sur le disque.
  En effet, pour déterminer le taux de performance nous avons utilisé la fonction **from_input_examples** de la classe **EmbeddingSimilarityEvaluator** prenanant l'ensemble de données de validation qui permet d'évaluer un modèle basé sur la similarité des plongements en calculant la corrélation des rangs de Spearman et Pearson par rapport aux étiquettes de référence. Les métriques sont la similarité cosinus ainsi que la distance euclidienne et Manhattan. Le score renvoyé est la corrélation de Spearman avec une métrique spécifiée.

  – La coefficient de Spearman : Est fondé sur l'étude de la différence des rangs entre les attributs des individus pour les deux caractères X et Y :

$$r_s = 1 - \frac{6\sum(x_i - y_i)^2}{n(n^2-1)}$$

FIG. 4.11 : La coefficient de Spearman

– La coefficient de Pearson : Ce coefficient permet de détecter la présence ou l'absence d'une relation linéaire entre deux caractères quantitatifs continus. Calculer par la formule suivante :

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

FIG. 4.12 : La coefficient de Pearson

- **losses** : Le résultat de la fonction de perte joue un rôle crucial dans le réglage du modèle et détermine la performance de notre modèle d'ensemble sur une tâche en aval spécifique. La classe **CosineSimilarityLoss** permet de calculer le taux de performance pour évaluer le modèle prenant en paramètre le modèle sentence-transformers chargé. Il calcule les vecteurs **u = model(input_text[0])** et **v = model(input_text[1])** et mesure la similarité en cosinus entre les deux. Par défaut, il minimise la perte suivante :
  input_label - cos_score_transformation(cosine_sim(u,v))

4.4.2.4 Entraînement et évaluation de modèle

Entraîner un modèle avec une cible d'entraînement donnée. Chaque cible d'entraînement est échantillonnée à son tour sous forme de lot. Nous n'échantillonnons que le plus petit lot de chaque cible pour garantir un entraînement égal sur chaque ensemble de données. Pour la formation de modèle avec la nouvelle data on a la fonction **fit()**, cette fonction a l'ensemble des attributs suivantes :

- **train_objectives** qui se compose de tuple(DataLoader,LossFunction),avec DataLoader classe pour l'encapsulation et LossFunction pour mésure la corrélation entre cosine score et label donné .
- **evaluator** est l'une de méthode de mésure de la performance de modèle que nous avons déjà parler avec détails dans la section précédente,
- **epochs** c'est le nombre d'époque pour la formation,
- **evaluation_steps** défini par le nombre d'étapes d'entraînement par époque,

- **warmup_steps** c'est le comportement dépend du planificateur. En effet,il est utilisé pour indiquer un ensemble d'étapes d'entraînement avec un taux d'apprentissage très faible,
- **output_path** attribut qui spécifie le chemin de stockage pour le modèle et les fichiers d'évaluation.

En effet, cette fonction utilise en paramètres des attributs pour évaluer le modèle, l'évaluation se fait dans l'entraînement pour mesurer la performance de modèle au cours de construction de modèle de modèle d'une part et d'autre part après l'entraînement pour évaluer le résultat final de l'entraînement dans cette phase nous avons utilisé que la fonction **EmbeddingSimilarityEvaluator.from_input_examples**.

```
Entrée [ ]: #Mesure n°1
            evaluator = EmbeddingSimilarityEvaluator.from_input_examples(DevExample)
            #Mesure n°2
            train_loss = losses.CosineSimilarityLoss(model=model)
            #10% of train data for warm-up
            warmup_steps = math.ceil(len(train_dataloader) * num_epochs * 0.1)
            # Train the model
            model.fit(train_objectives=[(train_dataloader, train_loss)],
                      evaluator=evaluator,
                      epochs=num_epochs,
                      evaluation_steps=1000,
                      warmup_steps=warmup_steps,
                      output_path=model_save_path)
            model = SentenceTransformer(model_save_path)
            #Evaluation après l'entrainement
            test_evaluator = EmbeddingSimilarityEvaluator.from_input_examples(TestExample)
            test_evaluator(model, output_path=model_save_path)
```

Fig. 4.13 : Entraînement et évaluation de modèle

4.5 Conclusion

Bibliographie & Webographie

1] https ://copyleaks.com/fr/. Consulté le 22/02/2022
[2] https ://pdf.abbyy.com/fr/learning-center/what-is-ocr/. Consulté le 29/03/2022
[3] https ://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501849-reconnaissance-vocale-definition-algorithmes-et-fonctionnement/. Consulté le 01/04/2022

[5] http ://www.novagen.tech/data-science-tirer-meilleur-parti-de-patrimoines-de-documents-pdf-complexes-2/. Consulté le 02/04/2022