République Algérienne Démocratique et Populaire
وزارة التعليـــم العـــــالـــي والبحـــث العـلـمـــي
Ministère de l'enseignement supérieur et de la recherche scientifique
جـــامعـــــة بـرج بوعـريـــريـــج
Université de Bordj Bou Arréridj

Faculté des Mathématiques et d'informatique

Département d'Informatique

MEMOIRE
Présenté en vue de l'obtention du Diplôme de
Master en Informatique
Spécialité : Ingénierie de l'Informatique Décisionnelle

Thème :

Extraction des règles d'association à partir des textes en utilisant la mesure TF-IDF

Réalisé par :

Nebbache Khaled

Rouabah Abdelbasset

*Soutenu le **27/10/2020** devant le jury composé de:*

Présidente: Mme. ***Boutouhami Sara***	MCA à L'U.EI Bachir El Ibrahimi-BBA
Examinatrice : Mme. ***Attia Safa***	MCB à L'U.EI Bachir El Ibrahimi-BBA
Encadrante : Mme. ***Saîfi Lynda***	MCB à L'U.EI Bachir El Ibrahimi-BBA

Promotion : 2019 / 2020

Remerciements

الحمد لله الذي وفقنا و أعاننا على إكمال هذا العمل

Nous tenons à remercier d'abord notre encadrante **Mme. *Saîfi Lynda*** pour son aide, ses conseils, et ses remarques objectives qui ont contribué à la réalisation de ce mémoire.

Nos sincères gratitudes aux professeurs *du* département d'Informatique spécialement: **Mme. *Benabid.***

Nous n'oublierons pas de remercier **nos parents** pour leur contribution, leur soutien, leur patience et leur encouragement morale.

Dans l'impossibilité de citer tous les noms, nos sincères remerciements vont à tous ceux et celles, qui de près ou de loin, ont permis par leurs conseils et leurs compétences la réalisation de ce mémoire.

Dédicaces

Je dédie ce travail à ma famille particulièrement à **mes chers parents** qui m'encouragent toujours.

À **mon frère**, **ma sœur** et **mon neveu** : Muslim.

À **mon oncle** : Salim.

À **mes proches amis**: R. Abdelbasset, B. Bilel, B. Mansour, B. Adem, B. Abdelbasset, A. Khaled, B. Seifeddine, F. Ayman, F. Houdhaifa, B. Hamza, A. Mokhtar, D. Ayoub…

Khaled

Dédicaces

C’est avec grand plaisir que je dédie ce modeste travail:

À celui qui a fait de moi un homme, **mon père**.

À l’être le plus cher de ma vie, **ma mère**.

À **mes chers Frère** et **Sœurs**.

Aux **petits enfants** : Abderrahmane et Ibrahim.

À **mes proches Amis**: Khaled, Mansour, Bilal, Adem.

Abdelbasset

Sommaire

Introduction générale

1. Contexte……1
2. Problématique……1
3. Objectif……2
4. Plan de mémoire……2

CHAPITRE I : Etat de l'art

1. Introduction……4
2. Le Data Mining (DM) ……4
   - 2.1 Les techniques de Data Mining……5
     - 2.1.1 La catégorisation……5
     - 2.1.2 Le clustering……5
     - 2.1.3 Les règles d'association……5
   - 2.2 Domaines d'utilisation de Data Mining……6
3. Le Text Mining (TM)……6
   - 3.1 Les taches de Text Mining……7
   - 3.2 Le processus de Text Mining……7
   - 3.3 Les techniques de Text Mining……9
     - 3.3.1 Le traitement de langage naturel (NLP) ……9
     - 3.3.2 La recherche d'information (RI) ……10
     - 3.3.3 L'extraction d'information (EI) ……10
   - 3.4 Comparaison entre 'RI' et 'EI'……11
4. Extraction des connaissances (EC)……11
   - 4.1 A partir des données (ECD)……11
   - 4.2 A partir des textes (ECT)……11
5. Les règles d'association……12
   - 5.1 Définition……12
   - 5.2 Motif…… 13
   - 5.3 Motif fréquent…… 13
   - 5.4 Item et Itemset……14
   - 5.5 Support d'une règle d'association……14
   - 5.6 Confiance d'une règle d'association……14
   - 5.7 Extraction des règles d'association……15
   - 5.8 Algorithmes d'extraction des règles d'association……15
     - 5.8.1 Algorithme Apriori……16

5.8.1.1 Avantages et inconvénients de l'algorithme Apriori……..16
6. Conclusion…… 17

CHAPITRE II : Architecture et Implémentation

1. Introduction……....18
2. Etapes d'extraction des règles d'association……..19
   2.1 Sélection et prétraitement des données textuelles…….20
      2.1.1 Tokenisation et élimination des mots vides……...….20
      2.1.2 Racinisation (ou stemming) et lemmatisation……..……..20
   2.2 Découverte des itemsets fréquents…….22
      2.2.1 Term Frequency (TF)……..……...22
      2.2.2 Inverse of Document Frequency (IDF)…….…….24
      2.2.3 La mesure TF-IDF……..……25
   2.3 Génération des règles d'association (L'algorithme APRIORI)…….27
3. Diagramme de flux de l'algorithme APRIORI……...37
4. Schéma du processus de l'application avec APRIORI……...38
5. Conclusion……...…….. 39

CHAPITRE III : Expérimentation

1. Introduction……40
2. Outils et languages utilisées……..…….…40
   2.1 Language de programmation……..…..40
   2.2 Outils de développements…….…41
   2.3 Les packages utilisés…….……41
3. Expérimentation et discussion des résultats……45
4. Conclusion…… 52

Conclusion générale et Perspectives……..….53

Table des figures

CHAPITRE I : Etat de l'art

Fig.I. 2 - Le processus de Text Mining 9
Fig.I. 4 - Extraction des connaissances à partir des textes……..……..12

CHAPITRE II : Architecture et Implémentation

Fig.II. 1 - Code python de la fonction 'createStopWords' 19
Fig.II. 2 - Code python de la fonction 'readFilesAndReturnCleanList' 21
Fig.II. 3 - Code python de la fonction 'createUniqueWordsList' 23
Fig.II. 4 - Code python de la fonction 'termFrequency' 23
Fig.II. 5 - Code python de la fonction 'findInverseDocumentFrequecy' 24
Fig.II. 6 - Code python de la fonction 'tfIdf' 26
Fig.II. 7 - Code python de la fonction 'createInputFileForAprioriAlgorithm' 27
Fig.II. 8 - Code python pour lire le fichier 'aprioriInput.txt' 28
Fig.II. 9 - Exemple de Data Set 28
Fig.II. 10 - Code python de la fonction 'generateC1' 29
Fig.II. 11 - La première liste candidate C1 29
Fig.II. 12 - Code Python de la fonction 'generateFrequentItemSet' 30
Fig.II. 13 - Code python de la fonction 'generateCandidateSets' 31
Fig.II. 14 - la deuxième liste candidate C2 32
Fig.II. 15 - La liste des itemsets L2 32
Fig.II. 16 - La troisième liste candidate C3 33
Fig.II. 17 - La liste des itemsets L3 33
Fig.II. 18 - Code python de la fonction 'generateAssociationRule' 34
Fig.II. 19 - Code python de la fonction 'aprioriOutput' 35
Fig.II. 20 – Schéma du processus de l'application avec APRIORI 38

CHAPITRE III : Expérimentation

Fig.III. 1 - Règles d'association générées de l'exemple 01 46
Fig.III. 2 - Règles d'association générées de l'exemple 02 49
Fig.III. 3 - Règles d'association générées de l'exemple 03 51

Liste des tables

CHAPITRE I : Etat de l’art

Tab.I. 1 - Comparaison entre 'RI' et EI'..... 11
Tab.I. 2 - Matrice Terms-Documents..... 13

Glossaire

- **Corpus :** Un corpus est un ensemble de documents (textes, images, …) pouvant provenir d'une ou de plusieurs disciplines, regroupés afin d'être soumis à des traitements.

- **Base de donnée textuelle :** Une base de données textuelles, ou base de données en texte intégral, est une compilation de documents ou d'autres informations présentée sous la forme d'une base dans laquelle le texte complet de chaque document référencé peut être visualisé en ligne, imprimé ou téléchargé.

Abréviations :

1. **TM:** **T**ext **M**ining.
2. **DM:** **D**ata **M**ining.
3. **FdT:** **F**ouille **d**e **T**exte.
4. **FdD:** **F**ouille **d**e **D**onnées.
5. **ECD:** **E**xtraction de **C**onnaissance à partir des **D**onnées.
6. **ECT :** **E**xtraction de **C**onnaissance à partir des **T**extes.
7. **NLP :** **N**atural **L**angage **P**rocessing.
8. **RI :** **R**echerche de l'**I**nformation.
9. **EI :** **E**xtraction de l'**I**nformation.
10. **TF:** **T**erm **F**requency.
11. **IDF:** **I**nverse **D**ocument **F**requency.
12. **TF-IDF:** **T**erm **F**requency **I**nverse **D**ocument **F**requency.
13. **KNN**: **K** **N**earest **N**eighbors.

Introduction générale

1. Contexte

Les dernières années, le terme de fouille de textes est apparu dans un bon nombre de publications. En effet, tous les travaux en la matière ont un objectif commun qui consiste à extraire des textes une information plus précise, associée à une sémantique rigoureuse afin d'aider un expert à enrichir son modèle de connaissances ou à effectuer tout autre tâche de raisonnement comme la veille technologique.

Toussaint propose une définition calquée sur celle de l'extraction des connaissances à partir de données : « L'extraction de connaissances à partir de textes est un processus non trivial qui construit un modèle de connaissances valide, nouveau, potentiellement utile et au final compréhensible à partir de textes bruts. » **[1]**. Ce processus commence par la modélisation des textes afin de les préparer pour la fouille de données, et se termine par l'interprétation des résultats de la fouille et l'enrichissement des connaissances. La fouille des données n'est donc qu'une étape du processus de fouille de texte.

2. Problématique

L'extraction des connaissances à partir de textes répond à la problématique de gérer et de traiter une grande masse de textes qui dépasse les capacités humaines.

La problématique générale en FdT est de tirer profit d'éléments d'information extraits afin d'exprimer des connaissances utilisables pour le domaine traité par les textes. Les nouvelles connaissances extraites servent à enrichir les connaissances actuelles d'un domaine contenues, par exemple, dans une base de connaissances.

3. Objectif

Notre objectif est de concevoir et réaliser une application qui permet de faire l'extraction des connaissances à partir des textes. L'idée est d'appliquer la mesure TF-IDF (**T**erm **F**requency **I**nverse **D**ocument **F**requency) pour sélectionner les mots importants, mesurer les termes pertinents et éliminer les mots rares qui seront par la suite utilisés comme entrées dans l'algorithme **APRIORI** afin de générer un ensemble des règles d'association portant sur les termes contenus dans les textes.

4. Plan de mémoire

Notre mémoire est composé de cinq parties principales :

- Dans l'**Introduction générale** nous allons présenter le contexte général de l'étude, la problématique, l'objectif qu'on doit atteindre et le plan de ce mémoire.

- Dans le **[CHAPITRE I : Etat de l'art]** nous allons citer des notions générales sur les domaines de : Data Mining, Text Mining en donnant quelques définitions, les taches principales, les
  domaines d'applications de chacun et surtout les techniques utilisées pour l'extraction des connaissances à partir des textes.

- Dans le **[CHAPITRE II : Architecture et Implémentation]** on a mis l'accent sur les algorithmes utilisés dans notre application en expliquant le rôle de chaque fonction.
  Notre application est dévisée en deux parties principales :
  - Application des fonctions de prétraitement et d'indexation de texte voir :
    Fonctions qui éliminent les mots vides, faire la racinisation, lemmatisation et les fonctions qui extraient les termes fréquents et le TF-IDF (**T**erm **F**requency **I**nverse **D**ocument **F**requency).
  - Application de l'algorithme APRIORI pour l'extraction des règles d'association.

- Dans le **[CHAPITRE III : Expérimentation]** on a présenté le langage de programmation utilisé, les outils logiciels, les packages et les bibliothèques et on a exécuté l'application avec des différentes bases de textes en expliquant les résultats obtenus.
- Dans la **Conclusion générale et Perspectives** on a conclu notre travail en expliquant les problèmes posés et en résumant les contributions apportées et les perspectives visées.

CHAPITRE I :

Etat de l’art

1. Introduction

La fouille de textes est une discipline qui a pour but le traitement automatique d'une base de données composée exclusivement de texte. Elle offre des perspectives nouvelles pour la statistique et répond au défi du traitement des données textuelles.

Nous présentons en premier lieu une définition de la fouille de textes en tant qu'étape particulière d'un processus plus général d'extraction de connaissances à partir des textes afin d'obtenir des informations plus précises pour aider à la décision.

Dans cette partie, nous allons donner un aperçu général sur le **Data Mining** (**DM**) ou **Fouille de données** (**FdD**), le **Text Mining** (**TM**) ou **Fouille de textes** (**FdT**), l'**Extraction de Connaissance à partir des Données** (**ECD**), l'**Extraction de Connaissance à partir des Textes** (**ECT**) et les techniques utilisées (motifs, règle d'association, classification…).

2. Le Data Mining (DM)

" *Le data mining, ou fouille de données, est l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de bases de données informatiques (souvent grandes), de façon automatique ou semi-automatique, en vue de détecter dans ces données des règles, des associations, des tendances inconnues ou cachées, des structures particulières restituant l'essentiel de l'information utile tout en réduisant la quantité de données* ". **[2]**

L'analyse de données depuis différentes perspectives et le fait de transformer ces données en informations utiles, en établissant des relations entre les données ou en repérant des patterns.

2.1 Les techniques de Data Mining :

Il existe plusieurs techniques de DM, parmi ces dernières on cite les trois suivantes :

2.1.1 La catégorisation (Classification supervisée):

Dans l'apprentissage supervisé, on fournit à l'ordinateur des exemples d'entrées qui sont étiquetés avec les sorties souhaitées. Le but de cette méthode est que l'algorithme puisse 'apprendre' en comparant sa sortie réelle avec les sorties 'enseignées' pour trouver des erreurs et modifier le modèle en conséquence. L'apprentissage supervisé utilise donc des modèles pour prédire les valeurs d'étiquettes sur des données non étiquetées supplémentaires. Parmi les méthodes de classification supervisée, on peut citer : les arbres de décision, les réseaux neurones, la méthode des k plus proche voisins (KNN) ou la classification bayésienne. **[5]**

2.1.2 Le clustering (Classification non supervisée) :

Dans l'apprentissage non supervisé ou le clustering, les données ne sont pas étiquetées à l'avance. L'idée étant que l'algorithme d'apprentissage est censé trouver tout seul des points communs parmi ses données d'entrée. Les données non étiquetées étant plus abondantes que les données étiquetées, les méthodes d'apprentissage automatique qui facilitent l'apprentissage non supervisé sont particulièrement utiles. **[5]**

2.1.3 Les règles d'association :

La recherche des règles d'association est l'un des sérieux problèmes du ECD. Le principe est de trouver des règles dans les données de types « si *Condition*, alors *Résultats* », notées *Conditions→ Résultats*. Cette technique permet la découverte de règles intelligibles et exploitables dans un ensemble de données volumineux, règles exprimant des associations entre items ou attributs dans une base de données. **[6]**

2.2 Domaines d'utilisation de Data Mining:

Les domaines d'application du DM sont vastes et variés. Cependant un trait commun les lies est le fait qu'ils traitent un volume important de données et tire des informations qui visent à améliorer la qualité du produit ou du service.

Parmi les domaines ou l'utilisation du DM est devenue monnaie courante:

- Laboratoires pharmaceutiques et médicaux.
- Assurances.
- Banques et les grandes administrations.
- Automobiles et grandes industries.
- Les transports à grandes échelles.
- Grande distribution et vente en correspondence. **[3]**

3. Le Text Mining (TM)

Le Text Mining, également appelé fouille de textes ou extraction de connaissance à partir de textes, est un ensemble de méthodes, de techniques et d'outils pour exploiter les documents non structurés que sont les textes écrits, comme les fichiers bureautiques de type word, les emails, les documents de présentation de type PowerPoint...etc. Pour extraire du sens de documents non structurés, le TM s'appuie sur des techniques d'analyse linguistique. Le TM est utilisé pour classer des documents, réaliser des résumés de synthèse automatique ou encore pour assister la veille stratégique ou technologique selon des pistes de recherches prédéfinies. **[3]**

TEXT MINING = LINGUISTIQUES + DATA MINING

3.1 Les taches de Text Mining :

Le Text Mining n'est pas un remplacement pour la recherche d'information ou le traitement du langage naturel. Les techniques qui permettent d'organiser un corpus de documents textuels selon leur contenu ont un spectre d'utilisation très large. Le TM cherche des réponses aux questions difficiles ou impossibles à résoudre avec les seuls moteurs de recherche. **[5]**

3.2 Le processus de Text Mining :

Les étapes nécessaires pour effectuer le processus de text mining sont :

- **L'acquisition ou sélection**: Source de données telle que : corpus textuels, bibliothèques électroniques, Web…
- **Le prétraitement du corpus** :
    - **Nettoyage:** Variable selon la source des données, cette phase consiste à réaliser des tâches telles que la suppression d'urls, d'emoji…
    - **Normalisation des données:**
        - **Tokenization** : ou découpage du texte en plusieurs pièces appelés *tokens*.

          Exemple: ***« Vous trouverez en pièce jointe le document en question » ; « Vous », « trouverez », « en pièce jointe », « le document », « en question ».***
        - **Stemming:** un même mot peut se retrouver sous différentes formes en fonction du genre (masculin féminin), du nombre (singulier, pluriel), la personne (moi, toi, eux…) etc. Le stemming désigne généralement le processus heuristique brut qui consiste à découper la fin des mots dans afin de ne conserver que la racine du mot.

          Exemple: ***trouverez → trouv***
        - **Lemmatisation :** cela consiste à réaliser la même tâche mais en utilisant un vocabulaire et une analyse fine de la construction des mots. La lemmatisation permet donc de supprimer uniquement les

terminaisons inflexibles et donc à isoler la forme canonique du mot, connue sous le nom de lemme.

Exemple: ***trouverez → trouver***

- **Pondération des termes :** permet de mesurer l'importance d'un terme dans un document cette importance est souvent calculées à partir des considération et interprétation statistique. L'objectif est de trouver les termes qui représente le mieux en appliquant la loi de Zipf: **rang * fréquence = constant**.

La relation entre la fréquence et le rang des termes permet de sélectionner les termes représentatifs d'un document : on élimine respectivement les termes de fréquences très élevées car ils ne sont pas représentatifs du document (on peut par exemple citer les mots outils), et les termes de fréquences très faibles (ce qui permet d'éliminer les fautes de frappes).

- **Autres opérations:** suppression des chiffres, ponctuation, symboles et *stopWords*, passage en minuscule. **[16]**

- **L'indexation** : permet de créer une représentation des documents dans le système, son objectif est de trouver les concepts les plus importants du document (ou de la requête), qui forme le descripteur de document.
- **Le Data Mining :** La fouille de données est l'étape centrale du processus d'extraction de connaissance. Elle consiste à découvrir de nouveaux modèles au sein de grandes quantités de données.
- **L'extraction des connaissances** : Application de l'un des algorithmes de la fouille de textes. **[5]**

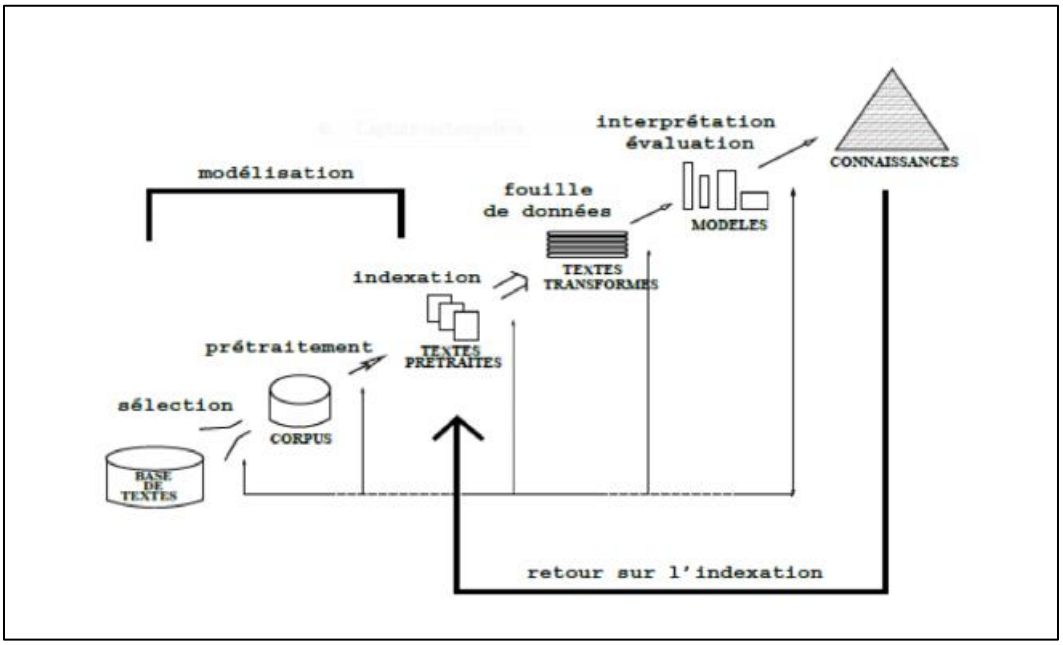

Fig.I. 1 - Le processus de Text Mining [16]

3.3 Les techniques de Text Mining :

Les techniques de Text Mining (TM) sont: Le traitement de langage naturel (NLP), la recherche d'information (RI) et l'extraction d'information (EI).

3.3.1 Le traitement de langage naturel (NLP) :

Un traitement de langage naturel est :

- Une suite d'actions ou calculs à faire par la machine. Le Traitement Automatique des Langues a pour objectif de traiter des données linguistiques (textes) exprimées dans une langue dite "naturelle".
- La Conception de programmes capables de traiter automatiquement des données linguistiques de type : textes écrits ; dialogues écrits ou oraux ; unités linguistiques (mots, phrases, énoncés, ...).

Les tâches impliquées dans cette technique peuvent inclure le nottoyage et la normalisation des données tels que : La tokenization, élimination des mots vides et filtrage de textes, Lemmatisation, La racinisation (ou troncature). **[7]**

3.3.2 La recherche d'information (RI) :

La recherche d'information « RI » s'intéresse aux documents dans leur globalité et aux thèmes qu'ils abordent, pour comparer les documents et détecter des typologies. Elle cherche à détecter tous les thèmes présents.

3.3.3 L'extraction d'information (EI) :

L'extraction d'information « EI » recherche des informations précises dans les documents, sans les comparer, en tenant compte de l'ordre et de la proximité des mots pour discriminer des énoncés différents ayant des mots clés identiques. L'extraction d'information consiste en l'alimentation d'une base de données structurée à partir de données exprimées en langage naturel. Il s'agit de détecter dans le texte en langage naturel les mots correspondant à chaque champ de la base de données. L'analyse est locale. L'extraction d'information est plus complexe, car elle nécessite d'effectuer une analyse lexicale et morpho-syntaxique pour reconnaître les constituants du texte (phrases, mots, verbes, adjectifs), leur nature pour détecter les phrases pertinentes et en extraire les informations voulues. **[5]**

3.4 Comparaison entre 'RI' et 'EI' :

	RI	EI
01	Récupère des informations précieuses à partir de texte non-structuré.	Extraire les informations des bases de données structurées
02	Tâche de recherche des documents textuels qui sont pertinents pour le besoin d'information d'utilisateur.	L'objectif est d'extraire les fonctionnalités pré-spécifié des documents ou d'affichage information.
03	Récupération des documents	Récupération des fonctionnalités
04	La sortie du RI est un sous-ensemble de documents qui sont pertinent pour la requête de l'utilisateur.	Plus difficile car cela nécessite des connaissances plus détaillées sur un document. Cela nécessite souvent établir des relations entre les caractéristiques.
05	**Outils:** Intelligent Miner Text Analyst	**Outils:** Text Finder Clear Forest Text

Tab.I. 1 - Comparaison entre 'RI' et EI'

4. Extraction des connaissances (EC) :

4.1 A partir des données (ECD):

L'extraction de connaissances à partir de bases de données est un processus non trivial qui construit un modèle valide, nouveau, potentiellement utile et au final compréhensible, à partir de données.

4.2 A partir des textes (ECT):

L'extraction de connaissances à partir de textes est un processus non trivial qui construit un modèle de connaissances valide, nouveau, potentiellement utile et au final compréhensible à partir de textes bruts**.**

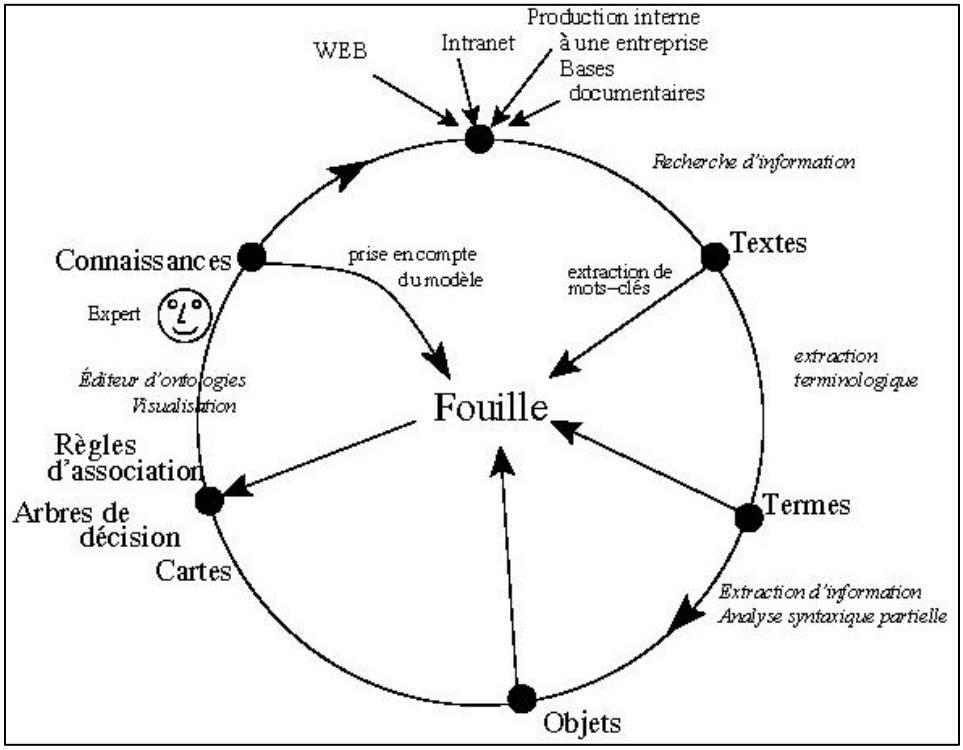

Fig.I. 2 - Extraction des connaissances à partir des textes [16]

5. Les règles d'associations :

5.1 Définition :

Une règle d' association est une application de la forme $X \rightarrow Y$ dans laquelle X et Y sont des ensembles de terme.

Il existe deux mesures importantes, le support et la confiance, la robustesse d'une règle d'association est déterminée grâce à ces deux métriques **[7]**. Une règle d'association qui a un support faible va être observée rarement. La confiance mesure la pertinence de l'inférence dans une règle, par exemple plus grande est la mesure de confiance de la règle $X \rightarrow Y$, plus cette règle sera pertinente.

5.2 Motif :

Soit ''T'' et ''D'' deux ensembles et R une matrice. 'T' est un ensemble de termes et 'D'' est un ensemble de textes tel que :

$$T = \{a, b, c, d, e\} \text{ et } D = \{d_1, d_2, \dots, d_6\}$$

La matrice R représente la relation binaire qui existe entre l'ensemble T et D. **[8]**

R	d_1	d_2	d_3	d_4	d_5	d_6
a	1	0	1	0	1	0
b	0	1	1	1	1	1
c	1	1	1	0	1	1
d	1	0	0	0	0	0
e	0	1	1	1	1	1

Tab.I. 2 - Matrice Terms-Documents

On appelle un motif tous les sous-ensembles de T. Un motif ''t'' est inclus dans un texte '*di*' si $\forall t \in t', R(t', di) = 1$. Un motif de taille K est appelé k-motif.

t: terme / **t'**: ensemble de terme

Par exemple : d3 et d5 contiennent le 4-motif.

5.3 Motif fréquent:

Si un motif ''t'' apparaît un nombre de fois et ce nombre est supérieur à un support minimal dans l'ensemble de textes D alors on dit qu'il est fréquent, i.e. $\text{support}(t) \geq \text{minsup}$ où minsup est le support minimal qui est donné par l'utilisateur.

Exemple :

Si $\text{minsup} = 3$ alors le motif $\{a, c, e\}$ n'est pas fréquent car $|f(\{a, c, e\})| = |\{d_3 d_5\}| = 2$.

5.4 Item et Itemset :

Un Item est un objet, élément ou un article d'une base de données.

Exemple 1 : **a** représente un item.

Exemple 2 : **c** représente un item.

Un Itemset est un ensemble d'items, d'objets ou d'articles d' une base de données.
Exemple :{ item2, item3, item4, item6}

Un K-Itemset est un ensemble de k éléments, ou k-Items, il est aussi un Itemset.
Exemple 1: {item2, item3, item4, item6} représente un 4-Itemset.
Exemple 2: {item2, item4, item6} représente un 3-Itemset.

5.5 Support d'une règle d'association :

Le support d'une règle d'association A→C définit le pourcentage de documents qui contiennent A et C. Il représente le nombre de documents qui contiennent A et C $\text{support}(A \cup C)$ divisé par le nombre total des documents :

$$\text{support}(A \rightarrow C) = \frac{\text{support}(A \cup C)}{|D|} \ . \ \text{support}(A \rightarrow C) \in [0,1] \ .$$

5.6 Confiance d'une règle d'association :

La confiance est une mesure permettant d'évaluer la validité d'une règle d'association. La confiance d'une règle d'association $A \rightarrow C$, notée $\text{conf}(A \rightarrow C)$ représente la proportion de documents qui contient A et qui contient aussi C. Elle est définie comme suit :

$$\text{conf}(A \rightarrow C) = \frac{\text{support}(A \cup C)}{\text{support}(A)} \ . \ \text{conf}(A \rightarrow C) \in [0,1] \ .$$

5.7 Extraction des règles d'association :

Avant tous on doit générer tous les motifs possibles à l'aide des termes d'une collection de documents textuels. Puis, calculer le support de chaque motif en parcourant la collection pour trouver le nombre de documents qui le contient, comparer la valeur trouvée avec la valeur de support minimal, et enfin ne garder que les motifs dont le support est supérieur au support minimal. Elle permet d'éliminer les motifs non fréquents qui sont sans intérêt pour générer les règles d'association. Les résultats de cette étape sont utilisés dans l'étape suivante, donc elle est primordiale.

A partir de l'ensemble des itemsets fréquents pour un seuil minimal de support minsup, la génération des règles d'association est un problème qui dépend exponentiellement de la taille de l'ensemble des itemsets fréquents. **[8]**

5.8 Algorithmes d'extraction des règles d'association :

Il existe plusieurs façons d'explorer les règles d'association, l'une de ces méthodes est la méthode naïve, on utilise alors toutes les combinaisons possibles des attributs et de leurs valeurs pour créer toutes les règles d'association possibles. Ce qui pose problème sur le plan complexité computationnelle du fait de l'explosion combinatoire. En effet, le nombre de règles générées est énorme. On peut optimiser cette méthode en gardant juste les règles avec un support et une confiance minimum. Cela reste insuffisant et les résultats sont insatisfaisants.

L'algorithme Apriori représente une approche révolutionnaire dans l'apprentissage et l'exploration des règles d'association.

5.8.1 Algorithme Apriori :

L'algorithme Apriori créé par Agrawal et Srikant en 1994, procède en deux temps. Il est basé sur le principe lié à l'approche de support et de confiance.
L'algorithme parcourt le treillis des itemsets pour rechercher les itemsets fréquents et en déduire les règles d'association dont la confiance dépasse le seuil de confiance minconf. Le treillis des itemsets permet d'utiliser plus efficacement cet algorithme d'extraction en admettant les propriétés suivantes :

- Tout sous-ensemble d'un itemset fréquent est fréquent.
- Tout sur-ensemble d'un itemset non fréquent est non fréquent.

Exemple de treillis des itemsets.

Le nombre d'itemsets fréquents qui peuvent être générés de n items est de $\mathbf{2^n - 1}$, la génération des itemsets fréquents est de complexité exponentielle, il est alors essentiel de trouver la méthode de recherche la plus optimale.

L'algorithme Apriori se fait comme suit :

- Générer les Règles candidates.
- Calculer le support pour chaque règle candidate.
- Apparier les règles dont on a calculé le support avec le support choisi.
- On rejette les candidats dont le support est inférieur au support minimale.
- On termine en sortie avec toutes les règles dont le support est supérieur au support minimal. **[9]**

5.8.1.1 Avantages et inconvénients de l'Algorithme Apriori :

Avantages: Il existe une multitude d'avantages dans l'utilisation de l'algorithme Apriori. On en énumère quelques-uns:

- La découverte rapide de règles d'association pertinentes entre objets.
- La facilité d'interprétation des résultats lors de l'extraction des règles d'association, malgré le nombre important de ces dernières. **[10] [11]**

Inconvénients : Les inconvénients auxquels on fait face lors d'une utilisation de l'algorithme Apriori sont les suivants:

- Les algorithmes d'extraction liés à l'approche support / confiance génèrent un grand nombre de règles d'association.
- Un nombre important de configurations d'items ne peuvent pas engendrer de règles d'association.
- La recherche de règles d'association impose un temps considérable qui peut s'avérer désavantageux si l'on fait face à une énorme base de données.

6. Conclusion

La réalisation de cet état de l'art a mis en exergue une suite logique à nos travaux sur l'extraction de règles d'association à partir des textes, à savoir la recherche des motifs fréquents, et la génération de règles d'association au moyen d'une mesure de qualité plus pertinente, par rapport à ladite mesure **confiance** d'Agrawal.

CHAPITRE II:

Architecture et Implémentation

1. Introduction

L'extraction des règles d'association est une méthode qui a vu le jour avec la recherche en bases de données (Documents textuels) pour retrouver les relations entre les termes d'une collection de document.

Par exemple on sera capable de dire que 70% des clients qui achètent du lait achètent en même temps des œufs (lait → œuf : 0.70), une telle constatation est très intéressante puisqu'elle aide le gestionnaire d'un supermarché à ranger ses rayons de telle sorte que le lait et les œufs soient à proximité.

Plusieurs travaux basés sur la recherche des règles d'association ont été appliqués dans des applications réelles comme :

- La planification commerciale.
- Les réseaux de télécommunication.
- Les applications biomédicales.
- Les données de web.
- Les applications de sécurité.
- La gestion des connaissances.
- Le recherche d'information.

Dans notre application on a utilisé l'algorithme **'APRIORI'** pour extraire ces règles d'associations passant aux plusieurs étapes tels que :

- Sélection et prétraitement des données textuelles.
- Découverte des itemsets fréquents.
- Génération des règles d'association (En appliquant l'algorithme APRIORI).

2. Etapes d'extraction des règles d'association

2.1 Sélection et prétraitement des données textuelles:

2.1.1 Tokenisation et élimination des mots vides:

Cette étape permet de préparer les données afin de leur appliquer les algorithmes d'extraction des règles d'association.

Quand on exécute le code il recherche le dossier nommé **TextBase** et lit tous les fichiers (.txt) qu'il contient. Chaque fichier texte agit comme un document distinct. Étant donné que l'entrée du code doit être une base de données de documents, nous avons plusieurs documents dans le dossier **TextBase**.

On a utilisé plusieurs fonctions pour trouver les motifs fréquents dans les textes.

```
def createStopWords(filename):
    stopWords = []
    with open(filename, "r") as file_object:
        for line in file_object:
            stopWords.append(line.rstrip())
    return stopWords
```

Fig.II. 1 - Code python de la fonction 'createStopWords'

Puisque l'objectif du text mining est de trouver des documents semblables ou encore d'extraire des informations uniques à chaque document, on peut éliminer les tokens (mots, lemmes, etc.) qui ne vont pas influence un classement, une recherche d'information etc. Les stop words incluent surtout les mots vides (déterminants, prépositions, conjonctions, verbes auxiliaires).

Selon le type d'analyse on va aussi éliminer les mots les plus fréquents (qui se retrouvent dans la plupart des documents) car elles ne permettent pas de discriminer entre les documents et aussi les mots les moins fréquents, car elles n'apportent que peu à l'analyse. Autrement dit on garde les mots représentatifs de chaque texte dans un corpus. L'élimination

des mots fréquents ou rares varient selon le type d'analyse. On les donne souvent comme paramètre à une analyse spécifique au lieu de le faire au niveau du nettoyage du corpus.

Si les mots sont étiquetés avec leur type ("part-of-speech"), on peut aussi sélectionner certains mots, par exemple seulement les verbes et les adjectifs.

Quand on appelle la fonction de la **Fig.II.1**:

`[stopWords = createStopWords("listOfStopWords.txt")]`, des mots vides sont généré à partir d'une liste de mots dans un fichier texte **ListOfStopWords.txt**

2.1.2 Racinisation (ou stemming) et lemmatisation :

Après on appelle la fonction de la **Fig.II.2:**

`[documentsInADictionary = readFilesAndReturnCleanList(stopWords)]`, elle va lire tous les fichiers et créer un dictionnaire propre en éliminant les ponctuations et en utilisant la racinisation (ou stemming) et la lemmatisation qui consistent à réduire une liste de mots à une liste plus courte qui ne contient qu'une variante du même mot. Le premier algorithme connu et toujours populaire est le **PorterStemmer** (importé depuis la bibliothèque **nltk.stem** du python).

La **racine** correspond à la partie du mot qui reste, une fois qu'on élimine son préfixe et son suffixe. Contrairement à un lemme, il ne s'agit pas forcément d'un "vrai" mot.

Exemples 1 :

recherche, chercher, cherches, cherchions, cherchait → cherch

fill, filled, filling → fill

Il existe un indice de compression *IC*:

- IC = (M - R) / M
- M = nombre de mots uniques avant racinisation
- R = nombre de racines uniques après racinisation

La lemmatisation cherche une forme canonique d'un mot. Le **lemme** est un mot de base, comme:

- Un verbe à l'infinitif
- Un adjectif, etc. au singulier masculin.

Exemples 2 :

cherches, cherchions → chercher

ai, as, a, eussions → avoir

belles, bel, beaux, beau → beau

```
def readFilesAndReturnCleanList(stopWords):
    punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~/+-1234567890'''
    cleanDocuments = {}
    individualDocument = []
    for file in os.listdir("./TextBase"):
        if file.endswith(".txt"):
            documentName = os.path.join(file)
            with open("./TextBase/" + documentName, "r") as document:
                for line in document:
                    cleanLine = line.rstrip()
                    listofWordsInALine = cleanLine.split(" ")
                    cleanWord = ""
                    ps = PorterStemmer()
                    for word in listofWordsInALine:
                        if word.lower() not in stopWords:
                            word = ps.stem(word)
                            for letter in word:
                                if letter not in punctuations:
                                    cleanWord = cleanWord + letter
                            if (cleanWord != ""):
                                individualDocument.append(cleanWord)
                            cleanWord = ""
                cleanDocuments[documentName] = individualDocument
                individualDocument = []
    return cleanDocuments
```

Fig.II. 2 - Code python de la fonction 'readFilesAndReturnCleanList'

2.2 Découverte des itemsets fréquents:

C'est l'étape la plus coûteuse en termes de temps d'exécution car, le nombre d'itemsets fréquents dépend exponentiellement du nombre d'items manipulés (pour n items, on a 2^n itemsets potentiellement fréquents). On doit trouver le TF-IDF de ses itemsets fréquents.

TF-IDF signifie "Term Frequency, Inverse Document Frequency". Il s'agit d'un moyen d'évaluer l'importance des mots (ou 'termes') dans un document en fonction de leur fréquence d'apparition dans plusieurs documents.

Intuitivement ...

Si un mot apparaît fréquemment dans un document, c'est important. Donnez au mot un score élevé. Mais si un mot apparaît dans de nombreux documents, ce n'est pas un identifiant unique. Donnez au mot une note faible. Par conséquent, les mots courants tels que 'le' et 'pour', qui apparaissent dans de nombreux documents, seront réduits. Les mots qui apparaissent fréquemment dans un seul document seront mis à l'échelle.

2.2.1 Term Frequency (TF) :

On donne la fréquence du mot dans chaque document du corpus. C'est le rapport du nombre de fois où le mot apparaît dans un document par rapport au nombre total de mots dans ce document. Il augmente à mesure que le nombre d'occurrences de ce mot dans le document. Chaque document a son propre **TF**.

TF(t) = (Nombre de fois où le terme 't' apparaît dans un document) / (Nombre total de termes dans le document)

On exécute ce code :

```
[listOfUniqueWords = createUniqueWordsList(documentsInADictionary)]
```

pour obtenir les mots uniques.

```
def createUniqueWordsList(documentsInADictionary):
    listOfUniqueWords = []
    for document, words in documentsInADictionary.items():
        for word in words:
            if word not in listOfUniqueWords:
                listOfUniqueWords.append(word)
    return listOfUniqueWords
```

Fig.II. 3 - Code python de la fonction 'createUniqueWordsList'

Ensuite on appelle la fonction **termFrequency :**

`[tf = findTermFrequency(documentsInADictionary)]`, et le **TF** sera obtenu.

```
def termFrequency(documentName, document):
    lengthOfDocument = len(document)
    wordInformation = []
    wordInfoDictionary = {}
    documentAndWordInfo = {}
    wordCounter = Counter(document)
    if lengthOfDocument != 0:
        for word, repetation in wordCounter.items():
            frequency = repetation / lengthOfDocument
            # wordInformation.append(repetation)
            wordInformation.append(frequency)
            # wordInformation.append(lengthOfDocument)

            wordInfoDictionary[word] = wordInformation
            wordInformation = []
    documentAndWordInfo[documentName] = wordInfoDictionary
    return documentAndWordInfo
```

Fig.II. 4 - Code python de la fonction 'termFrequency'

2.2.2 Inverse of Document Frequency (IDF):

Cette relation est utilisée pour calculer le poids des mots rares dans tous les documents du corpus. Les mots qui apparaissent rarement dans le corpus ont un score IDF élevé. Il est généralement exprimé comme suit :

IDF(t) = Log (N/df)

Où : ***df*** est le nombre de documents contenant le terme.

N est le nombre total de documents de la base documentaire.

```
def findInverseDocumentFrequency(listOfUniqueWords, documentsInADictionary, tf):
    idf = {}
    numberOfDocuments = len(documentsInADictionary)
    for word in listOfUniqueWords:
        for document, words in documentsInADictionary.items():
            if word in words:
                if word in idf:
                    idf[word] = idf[word] + 1
                else:
                    idf[word] = 1
    for word, repetation in idf.items():
        idf[word] = math.log10(numberOfDocuments / repetation)

    return idf
```

Fig.II. 5 - Code python de la fonction 'findInverseDocumentFrequecy'

L'appel de cette fonction est fait par ce code :

`[idf = findInverseDocumentFrequency(listOfUniqueWords, documentsInADictionary, tf)]`, Le IDF est extrait.

2.2.3 La mesure TF-IDF :

En combinant les deux techniques précédentes, elle donne une bonne approximation de l'importance du terme dans le document, particulièrement dans les corpus de documents de taille homogène. Cependant, elle ne tient pas compte d'un aspect important du document: sa longueur. En général, les documents les plus longs ont tendance à utiliser les mêmes termes de façon répétée, ou à utiliser plus de termes pour décrire un sujet. Par conséquent, les fréquences des termes dans les documents seront plus élevées, et les similarités à la requête seront également plus grandes. Pour pallier cet inconvénient, il est possible d'intégrer la taille des documents à la formule de pondération : on parle de facteur de normalisation. **[15]**

TF-IDF = TF * IDF

Où : ***TF*** est le Term Frequency.

IDF = Log(N/df).

Par l'exécution du code ci-dessous la fonction **tfIdf** sera appelée:

```
[tf_idf = tfIdf(tf, idf)]
```

```python
def tfIdf(tf, idf):
    tfIdfReturnDictionary = {}
    listForWordAndValue = []
    mainDocumentList = []

    for documentAndWords in tf:
        documentName = ""
        for document, wordsInfo in documentAndWords.items():
            documentName = document
            if len(wordsInfo) != 0:
                for word, wordTf in wordsInfo.items():
                    tfIdfValue = wordTf[0] * idf[word]

                    listForWordAndValue.append(word)
                    listForWordAndValue.append(tfIdfValue)
                    mainDocumentList.append(listForWordAndValue)
                    listForWordAndValue = []
        mainDocumentList = sorted(mainDocumentList, key=itemgetter(1), reverse=True)
        tfIdfReturnDictionary[documentName] = mainDocumentList
        documentName = ""
        mainDocumentList = []
    return tfIdfReturnDictionary
```

Fig.II. 6 - Code python de la fonction 'tfIdf'

Finalement, On appelle la fonction **createInputFileForAprioriAlgorithm** pour obtenir un fichier **aprioriInput.txt** qui contient le résultat final des algorithmes TF-IDF précédents (le fichier contient les mots clés fréquents)

`[createInputFileForAprioriAlgorithm(tf_idf, topN)]` ou N est le nombre maximale des mots clés qu'on veut extraire.

```
def createInputFileForAprioriAlgorithm(tf_idf, topN):
    count = 0
    writeToFile = open("aprioriInput.txt", "w")
    for documentName, values in tf_idf.items():
        length = len(values[:topN])
        if len(values) != 0:
            for i in values[:topN]:
                count = count + 1
                if count == length:
                    writeToFile.write(i[0])
                else:
                    writeToFile.write(i[0] + ", ")
            count = 0
            writeToFile.write("\n")
```

Fig.II. 7 - Code python de la fonction 'createInputFileForAprioriAlgorithm'

2.3 Génération des règles d'association (L'algorithme APRIORI) :

Apriori est un algorithme classique de recherche de règles d'association introduit par [Agrawal et Srikant]. C'est le premier algorithme destiné à la recherche de règles d'association. Apriori génère les motifs fréquents puis les relie entre eux pour générer les règles d'association. Il se base essentiellement sur la propriété d'anti-monotonie qui caractérise les motifs. Elle est utilisée à chaque itération de l'algorithme Apriori afin de minimiser au maximum le nombre de motifs candidats à tester.

Premièrement on va lire le Fichier texte (aprioriInput.txt) ligne par ligne pour récupérer son contenu.

```
with open("aprioriInput.txt") as fp:
    lines = fp.readlines()

for line in lines:
    line = line.rstrip()
    dataSet.append(line.split(","))

noOfTransactions = len(dataSet)
```

Fig.II. 8 - Code python pour lire le fichier 'aprioriInput.txt'

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Fig.II. 9 - Exemple de Data Set

Exemple : On considère que le minimum support = 2 % et le minimum confiance = 60 %.

Ensuite on exécute la fonction de la **Fig.II.10** afin de créer une table (la première liste candidate) contenant le nombre de supports de chaque élément présent dans l'ensemble de données - Appelé C1 (Candidate Set) :

```
[firstCandidateSet = generateC1(dataset)]
```

Etape 1 : k = 1

- Générer des ensembles d'éléments fréquents de longueur 1.
- Répétez jusqu'à ce qu'aucun nouvel ensemble d'éléments fréquents ne soit identifié.
  - Générer des ensembles d'éléments candidats de longueur (k + 1) à partir de la longueur k fréquente.
  - Elaguer les ensembles d'éléments candidats contenant des sous-ensembles de longueur k qui sont infréquents.
  - Comptez le support de chaque candidat en scannant le DB.
  - Éliminez les candidats non-fréquents, ne laissant que ceux qui sont fréquents.

```python
def generateC1(dataSet):
    productDict = {}
    returneSet = []
    for data in dataSet:
        for product in data:
            if product not in productDict:
                productDict[product] = 1
            else:
                productDict[product] = productDict[product] + 1
    for key in productDict:
        tempArray = []
        tempArray.append(key)
        returneSet.append(tempArray)
        returneSet.append(productDict[key])
        tempArray = []
    return returneSet
```

Fig.II. 10 - Code python de la fonction 'generateC1'

Résultat de l'exemple :

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Fig.II. 11 - La première liste candidate C1

On appelle la fonction **generateFrequentItemSet (la fonction récursive):**

`[frequentItemSet = generateFrequentItemSet(firstCandidateSet, noOfTransactions, minimumSupport, dataset, fatherFrequentArray)],`
qui compare le support de l'élément de l'ensemble candidat avec le support minimum (ici min-support = 2 si support-count d'éléments de l'ensemble candidat est inférieur à min-support, supprimez ces éléments). Cela nous donne le itemset L1.

```
def generateFrequentItemSet(CandidateList, noOfTransactions, minimumSupport, dataSet, fatherFrequentArray):
    frequentItemsArray = []
    for i in range(len(CandidateList)):
        if i % 2 != 0:
            support = (CandidateList[i] * 1.0 / noOfTransactions) * 100
            if support >= minimumSupport:
                frequentItemsArray.append(CandidateList[i - 1])
                frequentItemsArray.append(CandidateList[i])
            else:
                eleminatedItemsArray.append(CandidateList[i - 1])

    for k in frequentItemsArray:
        fatherFrequentArray.append(k)

    if len(frequentItemsArray) == 2 or len(frequentItemsArray) == 0:
        # print("This will be returned")
        returnArray = fatherFrequentArray
        return returnArray

    else:
        generateCandidateSets(dataSet, eleminatedItemsArray, frequentItemsArray, noOfTransactions, minimumSupport)
```

Fig.II. 12 - Code Python de la fonction 'generateFrequentItemSet'

```python
def generateCandidateSets(dataSet, eleminatedItemsArray, frequentItemsArray, noOfTransactions, minimumSupport):
    onlyElements = []
    arrayAfterCombinations = []
    candidateSetArray = []
    for i in range(len(frequentItemsArray)):
        if i%2 == 0:
            onlyElements.append(frequentItemsArray[i])
    for item in onlyElements:
        tempCombinationArray = []
        k = onlyElements.index(item)
        for i in range(k + 1, len(onlyElements)):
            for j in item:
                if j not in tempCombinationArray:
                    tempCombinationArray.append(j)
            for m in onlyElements[i]:
                if m not in tempCombinationArray:
                    tempCombinationArray.append(m)
            arrayAfterCombinations.append(tempCombinationArray)
            tempCombinationArray = []
    sortedCombinationArray = []
    uniqueCombinationArray = []
    for i in arrayAfterCombinations:
        sortedCombinationArray.append(sorted(i))
    for i in sortedCombinationArray:
        if i not in uniqueCombinationArray:
            uniqueCombinationArray.append(i)
    arrayAfterCombinations = uniqueCombinationArray
    for item in arrayAfterCombinations:
        count = 0
        for transaction in dataSet:
            if set(item).issubset(set(transaction)):
                count = count + 1
        if count != 0:
            candidateSetArray.append(item)
            candidateSetArray.append(count)
    generateFrequentItemSet(candidateSetArray, noOfTransactions, minimumSupport, dataSet, fatherFrequentArray)
```

Fig.II. 13 - Code python de la fonction 'generateCandidateSets'

Etape 2 : k = 2

- Génération de l'ensemble candidat C2 à l'aide de L1 (c'est ce qu'on appelle l'étape de jointure). La condition de jonction de L(k-1) et L(k-1) est qu'il doit avoir des éléments (k-2) en commun.
- Vérification que tous les sous-ensembles (sub-sets) d'un ensemble d'éléments (Itemsets) sont fréquents ou non, et s'ils ne le sont pas, supprimez cet ensemble d'éléments. (Exemple

de sous-ensemble de {I1, I2} sont {I1}, {I2} ils sont fréquents. Vérifiez pour chaque ensemble d'éléments).

- Recherche du support de ces itemsets en effectuant une recherche dans l'ensemble de données.

Résultat de l'exemple :

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

Fig.II. 14 - la deuxième liste candidate C2

Maintenant il compare le support candidat (C2) avec le nombre minimum de support (ici min-support = 2 si support-count de l'itemset candidat est inférieur à min-support, donc ces itemsets sera supprimé) cela nous donne l'ensemble d'éléments L2.

Résultat de l'exemple :

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

Fig.II. 15 - La liste des itemsets L2

Etape 3 : k = 3

- Génération de l'ensemble candidat C3 à l'aide de L2 (étape de jointure). La condition de jonction de L(k-1) et L(k-1) est qu'il doit avoir des éléments (k-2) en commun. Ici pour L2, le premier élément doit correspondre.

Donc l'ensemble d'éléments généré en rejoignant L2 est {I1, I2, I3} {I1, I2, I5} {I1, I3, i5} {I2, I3, I4} {I2, I4, I5} {I2, I3, I5}

- Vérification si tous les sous-ensembles de ces ensembles d'éléments sont fréquents ou non, et s'ils ne le sont pas, supprimez cet ensemble d'éléments. (Ici, les sous-ensembles de {I1, I2, I3} sont {I1, I2}, {I2, I3}, {I1, I3} qui sont fréquents. Pour {I2, I3, I4}, le sous-ensemble {I3, I4} n'est pas fréquent, alors supprimez-le. De même, vérifiez pour chaque ensemble d'éléments).
- Détermination du support de ces éléments restants en effectuant une recherche dans l'ensemble de données.

Résultat de l'exemple :

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Fig.II. 16 - La troisième liste candidate C3

Ensuite, il compare le support candidat (C3) avec le nombre minimum de support (ici min-support = 2 si support-count de l'élément de l'ensemble candidat est inférieur à min-support, supprimez ces éléments) cela nous donne l'ensemble d'éléments L3.

Résultat de l'exemple :

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Fig.II. 17 - La liste des itemsets L3

Etape 4 : k = 4

- Génération de l'ensemble candidat C4 à l'aide de L3 (étape de jointure). La condition de jonction de L(k-1) et L(k-1) (ou k = 4) est qu'ils doivent avoir des éléments (k-2) en commun. Ici pour L3, les 2 premiers éléments (items) doivent correspondre.
- Il vérifie que tous les sous-ensembles de ces ensembles d'éléments sont fréquents ou non (ici l'ensemble d'éléments formé en joignant L3 est {I1, I2, I3, I5} donc son sous-ensemble contient {I1, I3, I5}, ce qui n'est pas fréquent). Donc pas d'itemset dans C4.
- Nous nous arrêtons ici car aucun itemsets fréquents sont trouvés.

Maintenant c'est l'étape pour extraire les règles d'associations. On exécute la fonction **generateAssociationRule** :

```
[associationRules = generateAssociationRule(fatherFrequentArray)]
```

```python
def generateAssociationRule(freqSet):
    associationRule = []
    for item in freqSet:
        if isinstance(item, list):
            if len(item) != 0:
                length = len(item) - 1
                while length > 0:
                    combinations = list(itertools.combinations(item, length))
                    temp = []
                    LHS = []
                    for RHS in combinations:
                        LHS = set(item) - set(RHS)
                        temp.append(list(LHS))
                        temp.append(list(RHS))
                        # print(temp)
                        associationRule.append(temp)
                        temp = []
                    length = length - 1
    return associationRule
```

Fig.II. 18 - Code python de la fonction 'generateAssociationRule'

```
def aprioriOutput(rules, dataSet, minimumSupport, minimumConfidence):
    returnAprioriOutput = []
    for rule in rules:
        supportOfX = 0
        supportOfXinPercentage = 0
        supportOfXandY = 0
        supportOfXandYinPercentage = 0
        for transaction in dataSet:
            if set(rule[0]).issubset(set(transaction)):
                supportOfX = supportOfX + 1
            if set(rule[0] + rule[1]).issubset(set(transaction)):
                supportOfXandY = supportOfXandY + 1
        supportOfXinPercentage = (supportOfX * 1.0 / noOfTransactions) * 100
        supportOfXandYinPercentage = (supportOfXandY * 1.0 / noOfTransactions) * 100
        confidence = (supportOfXandYinPercentage / supportOfXinPercentage) * 100
        if confidence >= minimumConfidence:
            supportOfXAppendString = " Support Of A: " + str(round(supportOfXinPercentage, 2))+ " %" + " / "
            supportOfXandYAppendString = " Support of A & B: " + str(round(supportOfXandYinPercentage))+ " %" + " / "
            confidenceAppendString = " Confidence of A & B: " + str(round(confidence))+ " %" + "      "
            returnAprioriOutput.append(supportOfXAppendString)
            returnAprioriOutput.append(supportOfXandYAppendString)
            returnAprioriOutput.append(confidenceAppendString)
            returnAprioriOutput.append(rule)
    return returnAprioriOutput
```

Fig.II. 19 - Code python de la fonction 'aprioriOutput'

L’exécution de cette fonction va générer tous les règles d’associations possibles considérant le **min-sup** et le **min-conf** entrés par l’utilisateur.

Nous avons découvert tous les itemsets fréquents. Maintenant, la génération d'une règle d'association forte entre en scène. Pour cela, l’algorithme va calculer la confiance de chaque règle.

Confiance

Une confiance de 60% signifie que 60% des clients qui ont acheté du lait et du pain ont également acheté du beurre.

```
Confidence(A->B)=Support_count(AUB)/Support_count(A)
```

Résultat de l'exemple :

Si en prenant un exemple de n'importe quel itemset fréquent, nous montrerons la génération de règle.

Itemset {I1, I2, I3} // à partir de L3

Donc les règles générées :

[I1^I2]=>[I3] // confiance = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%
[I1^I3]=>[I2] // confiance = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%
[I2^I3]=>[I1] // confiance = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%
[I1]=>[I2^I3] // confiance = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%
[I2]=>[I1^I3] // confiance = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%
[I3]=>[I1^I2] // confiance = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%

Remarque : Si la confiance minimale est = 50%, les 3 premières règles peuvent être considérées comme des règles d'association fortes.

3. Diagramme de flux de l'algorithme APRIORI :

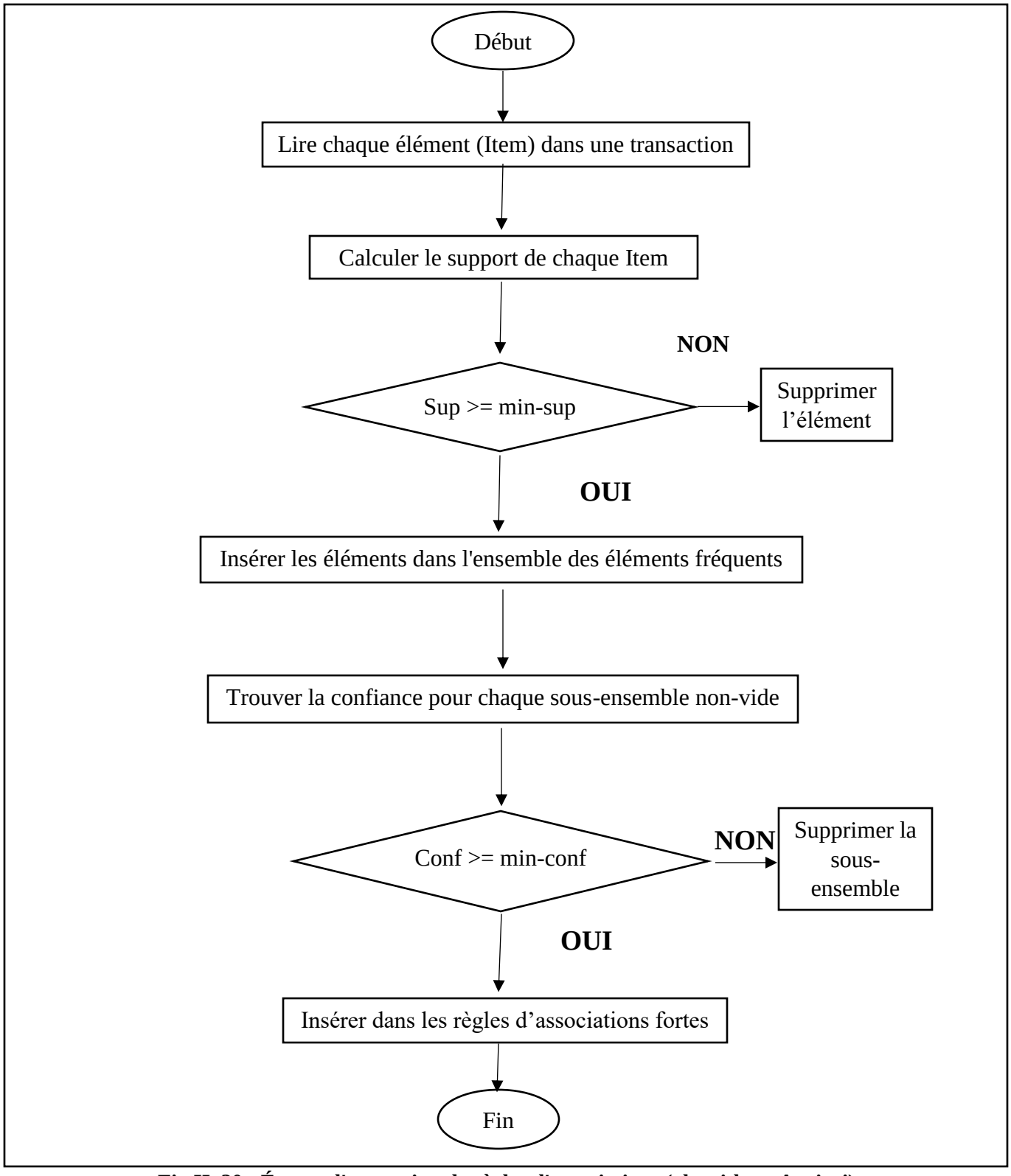

Fig.II. 20 - Étapes d'extraction de règles d'associations (algorithme Apriori)

4. Schéma du processus de l'application avec APRIORI :

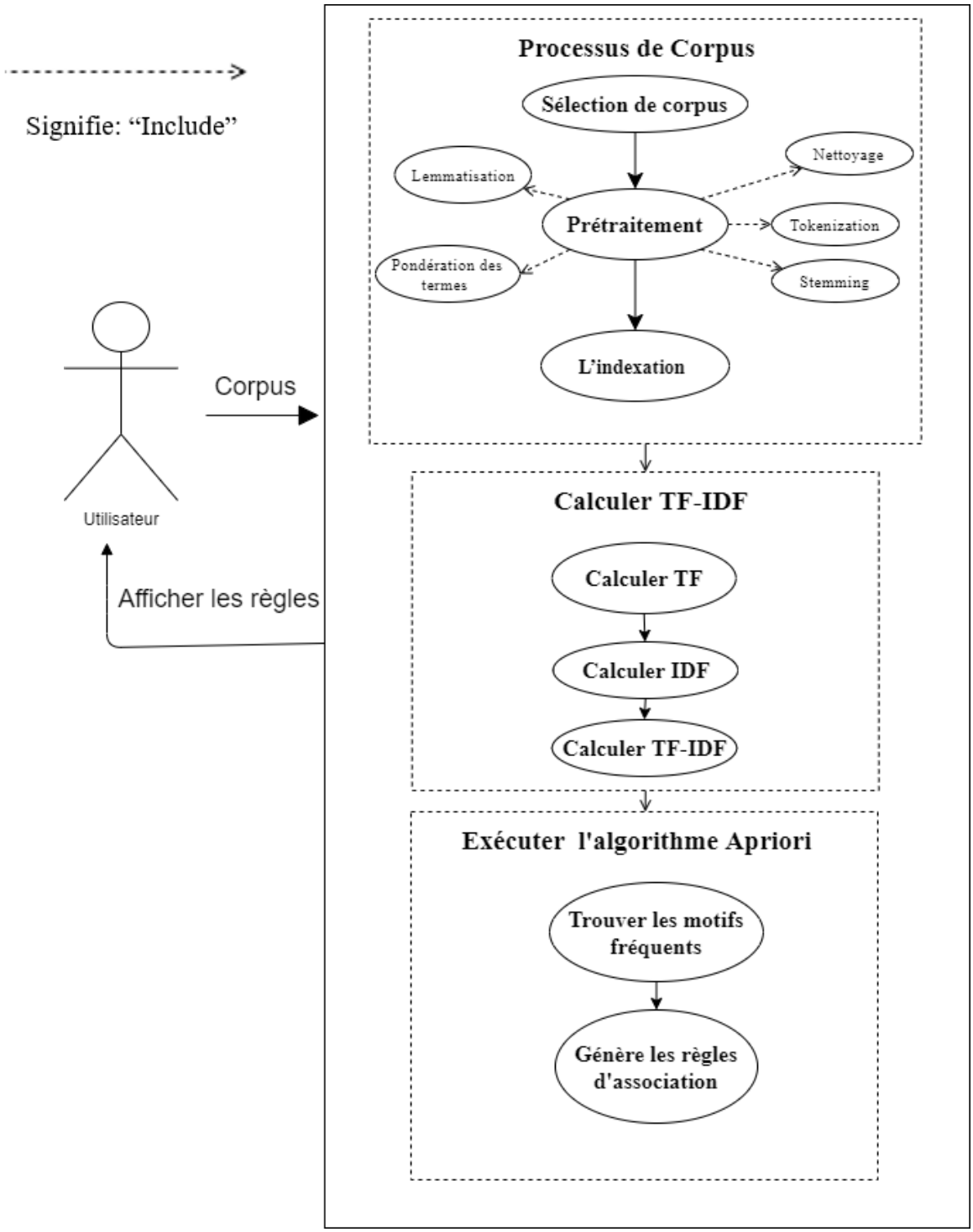

Fig.II. 20 – Schéma du processus de l'application avec APRIORI

5. Conclusion

Au cours de ce chapitre on a expliqué le fonctionnement du code en détail, expliquant le rôle de chaque fonction de détermination de TF-IDF. Puis on a expliqué les étapes de l'algorithme Apriori en donnant des exemples pour extraire les règles d'associations.

CHAPITRE III :

Expérimentation

1. Introduction

Ce chapitre est essentiellement axé sur les grandes lignes qui nous ont permis de réaliser et de mettre en œuvre ce projet, et les outils exploités pour le développement du logiciel tels que l'environnement de programmation, les principales fonctions de traitement à utiliser.

Aussi nous mettrons en évidence les différentes propriétés de notre application, et donnerons des captures-écrans, des principales interfaces et montrerons les spécifications et fonctionnalités qu'offre à l'utilisateur notre code, mais aussi une présentation du mode de fonctionnement.

On va tester notre application sur différents base de texte dans plusieurs domaines pour extraire les règles d'association et expliquer le résultat obtenu.

2. Outils et languages utilisés

2.1 Langage de programmation

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom *Python* vient d'un hommage à la série télévisée *Monty Python's Flying Circus* dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

La dernière version de Python est la version 3. Plus précisément, la version 3.7 a été publiée en juin 2018. La version 2 de Python est désormais obsolète et cessera d'être maintenue après le 1er janvier 2020. Dans la mesure du possible évitez de l'utiliser.

La *Python Software Foundation* est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs.

Ce langage de programmation présente de nombreuses caractéristiques intéressantes :

— Il est multiplateforme. C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.

— Il est gratuit. On peut l'installer sur autant d'ordinateurs qu'on veut (même sur le téléphone!).

— C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.

— C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté,

contrairement à des langages comme le C ou le C++.

— Il est orienté objet. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d'interactions.

— Il est relativement *simple* à prendre en main.

— Enfin, il est très utilisé en bio-informatique et plus généralement en analyse de données. **[13]**

2.2 Outil de développement :

PyCharm est un environnement de développement intégré utilisé pour programmer en Python.

Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django.

Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache. **[17]**

2.3 Les packages utilisés :

On a utilisé différentes bibliothèques de Python dans notre application tels que :

'NLTK', 'OS', 'COLLECTIONS', 'MATH', 'OPERATOR', 'TIME', 'INTERTOOL'

- **NLTK:**

Natural Language Toolkit est une bibliothèque logicielle en Python permettant un traitement automatique des langues, développée par Steven Bird et Edward Loper du département d'informatique de l'université de Pennsylvanie. En plus de la bibliothèque, NLTK fournit des démonstrations graphiques, des données-échantillon, des tutoriels, ainsi que la documentation de l'interface de programmation (API). **[18]**

- **PorterStemmer** : accepte la liste des mots tokenisés et la transforme en mot racine. **[19]**

- **OS:**

L'objectif principal du module **OS** est d'interagir avec votre système d'exploitation. La principale utilisation que j'en trouve est de créer des dossiers, de supprimer des dossiers, de déplacer des dossiers et parfois de changer le répertoire de travail. Vous pouvez également accéder aux noms des fichiers dans un chemin de fichier en faisant **listdir ()**. Nous ne couvrons pas cela dans cette vidéo, mais c'est une option.

Le module **os** fait partie de la bibliothèque standard, ou **stdlib**, dans Python 3. Cela signifie qu'il est livré avec votre installation Python, mais vous devez toujours l'importer. **[20]**

- **Collections :**

Ce module implémente des types de données de conteneurs spécialisés offrant des alternatives aux conteneurs, dict, liste, ensemble et tuple intégrés à usage général de Python. **[21]**

- **Collections.Counter :** est une sous-classe de dict pour compter les objets hachables. Il s'agit d'une collection non ordonnée où les éléments sont stockés en tant que clés de dictionnaire et leurs nombres sont stockés en tant que valeurs de dictionnaire. Les nombres peuvent être n'importe quelle valeur entière, y compris des nombres nuls ou négatifs. La classe Counter est similaire aux sacs ou multisets dans d'autres langues. **[21]**

- **Math :**

Ce module permet d'accéder aux fonctions mathématiques définies par le standard C. Ces fonctions ne peuvent pas être utilisées avec des nombres complexes; utilisez les fonctions du même nom du module **cmath** si vous avez besoin de la prise en charge des nombres complexes. La distinction entre les fonctions qui prennent en charge les nombres complexes et celles qui ne le sont pas est faite car la plupart des utilisateurs ne veulent pas apprendre autant de mathématiques que nécessaire pour comprendre les nombres complexes. La réception d'une exception au lieu d'un résultat complexe permet une détection plus précoce du nombre complexe inattendu utilisé comme paramètre, afin que le programmeur puisse déterminer comment et pourquoi il a été généré en premier lieu. **[22]**

- **Operator :**

Le module **Operator** exporte un ensemble de fonctions efficaces correspondant aux opérateurs intrinsèques de Python. Par exemple, **operator.add (x, y)** équivaut à l'expression x + y. Les noms de fonction sont ceux utilisés pour les méthodes de classe spéciales; des variantes sans début ni fin __ sont également fournies pour plus de commodité.

Les fonctions appartiennent à des catégories qui effectuent des comparaisons d'objets, des opérations logiques, des opérations mathématiques, des opérations de séquence et des tests de type abstrait. **[23]**

  - **Operator.itemgetter :** Renvoie un objet appelable qui extrait l'élément de son opérande à l'aide de la méthode **__getitem __ ()** de l'opérande. Si plusieurs éléments sont spécifiés, renvoie un tuple de valeurs de recherche. **[24]**

- **Time :**

Ce module fournit diverses fonctions liées au temps. Pour les fonctionnalités associées, voir également les modules **datetime** et **calendar**.

Bien que ce module soit toujours disponible, toutes les fonctions ne sont pas disponibles sur toutes les plateformes. La plupart des fonctions définies dans ce module appellent les fonctions de la bibliothèque de la plateforme C avec le même nom. Il peut parfois être utile de consulter la documentation de la plateforme, car la sémantique de ces fonctions varie selon les plateformes. **[25]**

- **Intertools :**

Ce module implémente de nombreuses briques d'itérateurs inspirées par des éléments de APL, Haskell et SML. Toutes ont été retravaillées dans un format adapté à Python.

Ce module standardise un ensemble de base d'outils rapides et efficaces en mémoire qui peuvent être utilisés individuellement ou en les combinant. Ensemble, ils forment une ‘algèbre d'itérateurs’ rendant possible la construction rapide et efficace d'outils spécialisés en Python. **[26]**

3. Expérimentation et discussion des résultats:

Remarque : Les textes utilisés sont en anglais.

Exemple 01 : Domaine médicale (COVID-19)

Texte 01 :

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus.

Most people infected with the COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.

The best way to prevent and slow down transmission is to be well informed about the COVID-19 virus, the disease it causes and how it spreads. Protect yourself and others from infection by washing your hands or using an alcohol based rub frequently and not touching your face.

The COVID-19 virus spreads primarily through droplets of saliva or discharge from the nose when an infected person coughs or sneezes, so it's important that you also practice respiratory etiquette (for example, by coughing into a flexed elbow).

Texte 02 :

Coronaviruses are a large family of viruses that cause respiratory infections. These can range from the common cold to more serious diseases.

COVID-19 is a disease caused by a new form of coronavirus. It was first reported in December 2019 in Wuhan City in China.

Symptoms of COVID-19 can range from mild illness to pneumonia. Some people will recover easily, and others may get very sick very quickly. People with coronavirus may experience symptoms such as: fever, respiratory symptoms...

Texte 03 :

Coronaviruses are a large group of viruses that cause diseases in animals and humans. They often circulate among camels, cats, and bats, and can sometimes evolve and infect people.

In animals, coronaviruses can cause diarrhea in cows and pigs, and upper respiratory disease in chickens. In humans, the viruses can cause mild respiratory infections, like the common cold, but can lead to serious illnesses, like pneumonia.

Coronaviruses are named for the crown-like spikes on their surface. Human coronaviruses were first identified in the mid-1960s. They are closely monitored by public health officials.

Texte 04 :

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus.

Most people infected with the COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.

Résultats de l'exemple 01 :

Support = 70 / Confiance = 80

```
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' respiratori']------>[' coronavirus']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' coronavirus']------>[' respiratori']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' diseas']------>[' coronavirus']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' coronavirus']------>[' diseas']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' mild']------>[' coronavirus']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' coronavirus']------>[' mild']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' respiratori']------>[' diseas']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' diseas']------>[' respiratori']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' respiratori']------>[' mild']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' mild']------>[' respiratori']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' mild']------>[' diseas']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' diseas']------>[' mild']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' respiratori']------>[' coronavirus', ' diseas']
Support Of A: 75.0 % /  Support of A & B: 75 % /  Confidence of A & B: 100 %      [' diseas']------>[' coronavirus', ' respiratori']
```

Fig.III. 1 - Règles d'association générées de l'exemple 01

Discussion des résultats :

- Dans la première règle on a : [respiratori → coronavirus] avec un support = 75 % et une confiance = 100 %. On conclut que Coronavirus est une maladie respiratoire (chronique).
- Dans la première règle on a : [coronavirus → diseas] avec un support = 75 % et une confiance = 100 %. On conclut que Coronavirus est une maladie.

Exemple 02 : Domaine scientifique (Earthquake)

Texte 01 :

An earthquake is the shaking of the surface of the Earth, resulting from the sudden release of energy in the Earth's lithosphere that creates seismic waves. Earthquakes can range in size from those that are so weak that they cannot be felt to those violent enough to toss people around and destroy whole cities. The seismicity or seismic activity of an area refers to the frequency, type and size of earthquakes experienced over a period of time.

At the Earth's surface, earthquakes manifest themselves by shaking and sometimes displacement of the ground. When the epicenter of a large earthquake is located offshore, the seabed may be displaced sufficiently to cause a tsunami. Earthquakes can also trigger landslides, and occasionally volcanic activity.

In its most general sense, the word earthquake is used to describe any seismic event — whether natural or caused by humans — that generates seismic waves. Earthquakes are caused mostly by rupture of geological faults, but also by other events such as volcanic activity, landslides, mine blasts, and nuclear tests. An earthquake's point of initial rupture is called its focus or hypocenter. The epicenter is the point at ground level directly above the hypocenter.

Texte 02 :

An earthquake is an intense shaking of Earth's surface. The shaking is caused by movements in Earth's outermost layer.

Texte 03 :

An earthquake is a sudden shaking movement of the surface of the earth. It is known as a quake, tremblor or tremor. Earthquakes can range in size from those that are so weak that they cannot be felt to those violent enough to toss people around and destroy whole cities. The seismicity or seismic activity of an area refers to the frequency, type and size of earthquakes experienced over a period of time.

An earthquake is measured on Richter's scale. A seismometer detects the vibrations caused by an earthquake. It plots these vibrations on a seismograph. The strength, or magnitude, of an earthquake, is measured using the Richter scale. Quakes measuring around 7 or 8 on the Richter scale can be devastating.

The effects of an earthquake are terrible and devastating. Many building, hospitals, schools, etc are destroyed due to it. A lot of people get killed and injured. Many people lose their money and property. It affects the mental health and emotional health of people.

Texte 04 :

An earthquake (also known as a quake, tremor or temblor) is the shaking of the surface of the Earth resulting from a sudden release of energy in the Earth's lithosphere that creates seismic waves. Earthquakes can range in size from those that are so weak that they cannot be felt to those violent enough to propel objects and people into the air, and wreak destruction across entire cities. The seismicity, or seismic activity, of an area is the frequency, type, and size of earthquakes experienced over a period of time. The word tremor is also used for non-earthquake seismic rumbling.

At the Earth's surface, earthquakes manifest themselves by shaking and displacing or disrupting the ground. When the epicenter of a large earthquake is located offshore, the seabed may be displaced sufficiently to cause a tsunami. Earthquakes can also trigger landslides and occasionally, volcanic activity.

In its most general sense, the word earthquake is used to describe any seismic event—whether natural or caused by humans—that generates seismic waves. Earthquakes are caused mostly by rupture of geological faults but also by other events such as volcanic activity, landslides, mine blasts, and nuclear tests. An earthquake's point of initial rupture is called its hypocenter or focus. The epicenter is the point at ground level directly above the hypocenter.

Résultats de l'exemple 02 :

Support = 90 / Confiance = 90

```
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' shake']------>[' earthquak']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' earthquak']------>[' shake']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' earthquak']------>[' caus']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' caus']------>[' earthquak']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' shake']------>[' caus']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' caus']------>[' shake']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' shake']------>[' caus', ' earthquak']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' earthquak']------>[' caus', ' shake']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' caus']------>[' earthquak', ' shake']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' earthquak', ' shake']------>[' caus']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' shake', ' caus']------>[' earthquak']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' earthquak', ' caus']------>[' shake']
```

Fig.III. 2 - Règles d'association générées de l'exemple 02

Discussion des résultats :

- Dans la première règle on a : [shake → earthquake] avec un support = 100 % et une confiance = 100 %. On conclut qu'un tremblement de terre est un mouvement de secousse soudain de la surface de la terre.
- Dans la huitième règle on a : [cause → shake] avec un support = 100 % et une confiance = 100 %. On conclut que le tremblement de terre est causé par le mouvement des plaques continentales.

Exemple 03 : Domaine sportif (Football)

Texte 01 :

Association football more commonly known as football or soccer, a sport played between 2 slides of 11 players with a spherical ball. Football is the world's most popular sport.

Outfield players move the ball with any part of the body except their hands or arms, whilst the ball is in play. Only the goal keeper can use their hands and this is only in their penalty area. When the ball goes out of play at either side of the pitch an opposing player from the side which did not put the ball out of play can throw the ball back into play using their hands. Both feet must remain on the floor behind the throw line. Both hands must remain on the ball until it is released from behind the throwers head.

The object of football is to outscore your opponents. A goal is scored when the entire ball crosses the goal line between the goal posts. A draw is when both teams score the same amount of goals during the allotted time.

Texte 02 :

Football is a family of team sports that involve, to varying degrees, kicking a ball to score a goal. Unqualified, the word football normally means the form of football that is the most popular where the word is used. Sports commonly called football include association football (known as soccer in some countries); gridiron football (specifically American football or Canadian football); Australian rules football; rugby football (either rugby union or rugby league); and Gaelic football. These various forms of football share to varying extent common origins and are known as football codes.

There are a number of references to traditional, ancient, or prehistoric ball games played in many different parts of the world. Contemporary codes of football can be traced back to the codification of these games at English public schools during the 19th century. The expansion and cultural influence of the British Empire allowed these rules of football to spread to areas of British influence outside the directly controlled Empire. By the end of the 19th century, distinct regional codes were already developing: Gaelic football, for example, deliberately incorporated the rules of local traditional football games in order to maintain their heritage. In 1888, The Football League was founded in England, becoming the first of many professional football competitions. During the 20th century, several of the various kinds of football grew to become some of the most popular team sports in the world.

Texte 03 :

What is football? Football (American: soccer) is a sport between two teams of 11 players that involves kicking a ball with their feet with the objective of scoring more goals than the opposing team in a fixed time (usually 2 x 45 minutes).

Résultats de l'exemple 03:

Support = 80 / Confiance = 70

```
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' soccer']------>[' footbal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' footbal']------>[' soccer']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' sport']------>[' footbal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' footbal']------>[' sport']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' footbal']------>[' ball']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' ball']------>[' footbal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' goal']------>[' footbal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' footbal']------>[' goal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' score']------>[' footbal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' footbal']------>[' score']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' team']------>[' footbal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' footbal']------>[' team']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' sport']------>[' soccer']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' soccer']------>[' sport']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' soccer']------>[' ball']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' ball']------>[' soccer']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' soccer']------>[' goal']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' goal']------>[' soccer']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' soccer']------>[' score']
Support Of A: 100.0 % /  Support of A & B: 100 % /  Confidence of A & B: 100 %      [' score']------>[' soccer']
```

Fig.III. 3 - Règles d'association générées de l'exemple 03

Discussion des résultats :

- Dans la troisième règle on a : [sport → football] avec un support = 100 % et une confiance = 100 %. On conclut que le football est un sport.
- Dans la cinquième règle on a : [football → ball] avec un support = 100 % et une confiance = 100 %. On conclut qu'on joue le football avec un ballon.

Discussion générale des résultats :

En implémentant le code et expérimentant des exemples**,** on a observé plusieurs remarques comme :

- La valeur de support et de confiance que nous saisissons ne donnent pas toujours un résultat : **'There are no association rules for this support and confidence'**.
- La génération des règles d'association prend un temps différent selon la valeur de support et confiance saisie.
- L'algorithme Apriori que nous utilisons génère un grand nombre de règles.

4. Conclusion

Dans ce chapitre on a implémenté notre code en testant des exemples venant de plusieurs domaines et on a obtenu les règles d'association.

Conclusion générale Et Perspectives

Conclusion générale et Perspectives

Les travaux présentés dans ce mémoire ont porté sur l'extraction des connaissances à partir des textes. Au cours de ce travail on a tout d'abord présenté un état de l'art qui explique brièvement le concept de Fouille de texte (Text Mining) en précisant le processus, les techniques et les domaines d'application de ce dernier. On a choisi l'un des méthodes qui est l'extraction des règles d'association.

Puis, on a présenté le concept de notre travail en expliquant le fonctionnement des algorithmes qui permettent la recherche des motifs fréquents et la génération des règles d'association valides entre les concepts à l'aide de l'algorithme Apriori.

Finalement, on a présenté l'application en exécutant notre code python de l'algorithme APRIORI. On a aussi fait des expérimentations pour extraire les règles d'association à partir des bases textuelles dans différents domaines.

Malheureusement, le temps attribué à ce travail a passé rapidement, d'où il était difficile d'enrichir notre travail et étudier d'autres approches et algorithmes. Nous proposons comme perspectives :

- Appliquer d'autres méthodes de l'extraction de connaissances à partir des textes.
- Tester d'autres algorithmes de génération des règles d'association tels que : FP-GROWTH, ECLAT et CLOSE ensuite comparer les résultats avec APRIORI.

Résumé

Les masses de données textuelles aujourd'hui disponibles engendrent un problème difficile pour les traiter. Dans ce cadre, des méthodes de Fouille de Texte (Text Mining) sont nécessaires pour extraire les connaissances à partir des textes.

Notre travail consiste à étudier l'une des méthodes d'extraction des connaissances, ou on traite le corpus textuel puis extrait les motifs fréquents pour générer les règles d'association entre les concepts avec leurs supports et leurs confiances à l'aide de l'algorithme APRIORI.

Nous avons finalisé ce mémoire par l'implémentation de cet algorithme, l'évaluation des exemples et la discussion des résultats.

Les mots clés : Fouille de texte, Text Mining, connaissances, corpus textuel, motifs fréquents, règles d'association, support, confiance, APRIORI.

Abstract

The masses of textual data available today create a difficult problem to process. In this context, Text Mining methods are necessary to extract knowledge from texts.

Our work consists in studying one of the knowledge extraction methods, where we process the textual corpus then extract the frequent patterns to generate the association rules between the concepts with their supports and their confidences using the APRIORI algorithm.

We finalized this dissertation by implementing this algorithm, evaluating the examples and discussing the results.

Keywords: Text Mining, knowledge, textual corpus, frequent patterns, association rules, support, confidence, APRIORI.

ملخص

تخلق كتل البيانات النصية المتاحة اليوم مشكلة صعبة في معالجتها. في هذا السياق ، تعتبر طرق التنقيب عن النص ضرورية لاستخراج المعرفة من النصوص.

يتمثل عملنا في دراسة إحدى طرق استخلاص المعرفة، حيث نقوم بمعالجة مجموعة النصوص ثم استخراج الأنماط المتكررة لتوليد قواعد الارتباط بين المفاهيم مع support و confidence الخاص بكل علاقة وذلك باستخدام خوارزمية APRIORI.

ختمنا هذه المذكرة بتطبيق هذه الخوارزمية وتقييم الأمثلة ومناقشة النتائج.

الكلمات المفتاحية: التنقيب عن النص، المعرفة، مجموعة النصوص، الأنماط المتكررة، قواعد الارتباط، support، confidence، APRIORI .

Bibliographie

[1] TOUSSAINT, Yannick : « Extraction de connaissances à partir de textes structurés », Vol. 8, p. 11-34, 2004/3.

[2] Kantardzic M: "Data mining concepts, models, methods and algorithms". Press ,Piscataway, NJ, USA, 2003

[3] R. Lefébure, G.Venturi : « Le Data Mining » Edition EYROLLES, deuxième tirage 1998

[4] A.Taibi, H.LAZREG : «Utilisation des algorithmes d'apprentissage dans la catégorisation automatique thématique de documents Etude de cas : les algorithmes K_PPV, Naïve Bayes», Mémoire de Licence, Université de M'sila, 2011-2012

[5] S.Raheel : « L'Apprentissage Artificiel pour la Fouille de Données Multilingues: Application à la Classification Automatique des Documents Arabes », Thèse de doctorat en Sciences de l'Information et de la Communication, Université Lumière Lyon 2, 2010

[6] R. Agrawal, T. Imielinski, A. N. Swami: "Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference"*, volume 22, pages 207–216, Washington, DC, 1993.

[7] Blanchard Julien, Kuntz Pascale, Guillet Fabrice, Gras Régis : "Mesure de la qualité des règles d'association par l'intensité d'implication entropique", IRIN - École polytechnique de l'université de Nantes, 2002.

[8] H. CHERFI : « Etude et réalisation des d'un système d'extraction de connaissance à partir de textes », 2006

[9] ABDERRAOUF NOUASRIA : « EXTRACTION D'ASSOCIATIONS LEXICALES FORTES DANS LES COMMENTAIRES », L'UNIVERSITÉ DU QUÉBEC À TROIS RIVIÈRES, JUIN 2016.

[10] Achouri Abdelghani : "Extraction de relations d'association maximales dans les textes :

représentation graphique", Université du Québec à Trois-Rivières, 2012.

[11] Pagé, Christian : "Bases de règles multi-niveaux", Université du Québec à Montréal, Fevrier 2008.

[12] Shashikumar G. Totad, Geeta R. B,Prassad Reedy: "Batch Processing for Incrementing FP-tree Construction", international Journal of Computer Applications, 2010.

[13] Fuchs. Patrick, Poulain. Pierre : "Cours Python", Université de Paris France, 2020

[14] R. Agrawal and R. Srikant: “Fast algorithms for mining association rules”. In *Proceedings of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[15] Dawid Weiss: «Descriptive Clustering as a Method for Exploring Text Collections», PhD thesis, Institute of Computing Science, Poznàn, Poland. 2006.

Webographie

[16] https://www.cairn.info/revue-document-numerique-2004-3-page-11.htm?contenu=article consulté le 25 septembre 2020

[17] https://fr.wikipedia.org/wiki/PyCharm consulté le 28 septembre 2020

[18] https://fr.wikipedia.org/wiki/Natural_Language_Toolkit consulté le 12 octobre 2020

[19] https://www.guru99.com/stemming-lemmatization-python-nltk.html consulté le 12 octobre 2020

[20] https://pythonprogramming.net/python-3-os-module/ consulté le 14 octobre 2020

[21] https://docs.python.org/2/library/collections.html consulté le 14 octobre 2020

[22] https://docs.python.org/3/library/math.html consulté le 14 octobre 2020

[23] https://docs.python.org/2/library/operator.html consulté le 14 octobre 2020

[24] https://docs.python.org/3/library/operator.html consulté le 14 octobre 2020

[25] https://docs.python.org/3/library/time.html consulté le 15 octobre 2020

[26] https://docs.python.org/fr/3.9/library/itertools.html consulté le 15 octobre 2020