MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Ingénierie des Systèmes d'Information**

Présenté par :

KHOUSSA Khoukha

THEME :

Extraction intelligente et interactive d'information à partir du texte arabe

Soutenu le : 27 juin 2021.

Devant le jury composé de :

KAID SLIMANE B	Maitre de conférence B	Université de Mostaganem	Président
HOCINE N	Maitre de conférence B	Université de Mostaganem	Examinateur
SEHABA Karim	Maitre de conférence A	Université de Mostaganem	Encadreur

Année Universitaire 2020-2021

Dédicaces

Je dédie ce mémoire

À mes chers parents

À mes sœurs et mes frères

À mes adorables nièces et neveux

À toute ma famille

À mes copines

A tous ceux qui, par un mot, m’ont donné la force de continuer...

Remerciements

Je souhaite manifester mes sincères remerciements à Allah, le tout puissant, pour ses faveurs et ses gratitudes de m'avoir donné le courage et la patience pour achever ce modeste travail.

Je tiens à présenter de tout mon cœur mes remerciements et mes reconnaissances à mon honorable encadreur M. Karim SEHABA pour son aide, ses conseils précieux, sa gentillesse, son encouragement, sa disponibilité et sa confiance qui m'a permis de ne jamais faiblir et de poursuivre toujours plus loin dans mon travail.

Je remercie naturellement ma famille pour son aide, sa générosité et son soutien moral qui ont été pour moi une source de courage et de confiance.

Je remercie aussi vivement les honorables membres du jury qui ont accepté d'évaluer ce travail.

Enfin, un grand remerciement à tous ceux qui, par un simple mot, m'ont donné la force de continuer à travailler afin d'atteindre mes objectifs.

الملخص

استخراج المعلومات من البيانات مهمة أساسية تمكن من تحليل كميات كبيرة من البيانات النصية للكشف عن معلومات جديدة أو للمساعدة في الإجابة على أسئلة بحثية محددة. وعلى الرغم من أن هذه المهمة صعبة للغاية بسبب القواعد الفريدة لكل لغة ، فإن أداء الطرق المقترحة في هذا المجال آخذ في تحسن. غير أننا لاحظنا أن الدراسات المتعمقة في النص العربي قليلة بسبب صعوبة هذه اللغة وهذا راجع الى عوامل كثيرة. ولذلك من المهم التفكير في مهمة استخراج المعلومات من النص العربي وعلى وجه التحديد استخراج الأنماط. وكجزء من هذا العمل ،أجرينا دراسة بحثية عن استخراج المعلومات من النص العربي بالاعتماد على استخراج الأنماط باستخدام تقنيات تعلم الآلية .

الكلمات المفتاحية : استخراج المعلومات ، النص العربي، استرجاع الأنماط، تعلم الآلية.

Résumé

L'extraction d'information (EI) à partir du texte est une tâche essentielle qui permet d'analyser de grandes quantités de données textuelles pour découvrir de nouvelles informations ou aider à répondre à des questions de recherche précises. Bien que cette problématique soit très difficile a cause des règles morphologiques uniques de chaque langue, les performances des approches proposées dans l'état de l'art s'améliorent. Néanmoins, nous avons remarqué que très peu d'études d'extraction d'information à partir d'un texte arabe ont été réalisé à cause de la difficulté de cette langue notamment. Notre projet porte ainsi sur l'EI à partir du texte arabe, concrètement l'extraction des motifs. Dans le cadre de ce travail, une étude de recherche sur l'extraction d'information à partir du texte arabe a été réalisée en se basant sur l'extraction des motifs en utilisant les techniques d'apprentissage machine.

Mots-clés: Extraction d'information (EI), Texte arabe, Extraction des motifs, Apprentissage machine.

Abstract

Information extracting from textual data is an essential task that enables large amounts of textual data to be analyzed in order to uncover new information or help to answer specific research questions. Although this problem is very difficult because of the unique rules of each language, the performance of the approaches proposed in the state of the art is improving. However, it was noted that in-depth studies in the Arabic text were little addressed because of the difficulty of this language due to many factors. It is therefore important to think about the problem of IE from the Arabic text, specifically patterns extraction. As part of this work, a research study on the IE from the Arabic text is carried out based on the patterns extraction using machine learning techniques.

Keywords: Information Extracting, Arabic text, Patterns Extraction, Machine Learning.

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète
IA	**I**ntelligence **A**rtificielle
ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**rocessing
DM	**D**ata **M**ining
TM	**T**ext **M**ining
EI	**E**xtraction d'**I**nformations
RI	**R**echerche d'**I**nformations
SW	**S**top **W**ords
LEMs	**L**emmatisation
PS	**P**refix **S**pan
FPG	**F**requent **P**attern **G**rowth

Table des matières

Introduction Générale

L'explosion de la masse de données textuelles et les besoins des utilisateurs sont toujours en croissance, alors la nécessité d'un système de traitement et d'analyse intelligent et interactif de cette masse de données n'est donc plus à démontrer.

Étant donné qu'il y a beaucoup d'informations présentées dans différentes versions de documents textes, la fouille de textes est une procédure d'une importance critique d'une coté et d'une autre coté le processus de rendre ce texte lisible pour les machines est très difficile. Le domaine de fouille de texte est relativement nouveau et lié aux domaines de recherche différents tels que l'extraction d'informations, l'exploration de données, l'apprentissage automatique, les statistiques et la linguistique informatique.

L'extraction d'information (EI) à partir du texte est une tâche très importante qui permet à examiner d'importantes collections de documents pour découvrir de nouvelles informations ou aider à répondre à des questions de recherche précises. Bien que cette problématique soit très difficile, les performances des approches proposées dans l'état de l'art s'améliorent car ils ont commencé à appliquer les techniques de fouille de texte pas seulement sur l'arabe standard moderne mais aussi à l'arabe ancienne de coran.

L'**EI** est une technique d'analyse de texte qui extrait des éléments de données spécifiques d'un texte. En utilisant l'extraction de texte, les entreprises peuvent éviter tous les problèmes de trier leurs données manuellement pour extraire des informations clés. Ci-dessous, nous nous référerons à certaines des principales tâches de l'extraction de texte: Extraction de mots clés (KeyWord Extraction), reconnaissance d'entité nommée (Named Entity Recognition), Extraction de caractéristiques et motif .

D'après un état de l'art sur les techniques d'intelligente artificielle liée au domaine de fouille dans différentes langues, on a remarqué que les études approfondies dans le contexte arabe ont été peu abordées. Ce constat peut s'expliquer par plusieurs raisons, notamment les règles morphologiques uniques de la langue. C'est une langue difficile en raison de nombreux facteurs comme les lettres et les diacritiques ainsi que la grammaire unique de cette langue.

L'arabe est compté parmi les langues les plus populaires du monde. Autant que 280 millions de personnes l'utilisent comme langue maternelle alors qu'au moins 250 millions le considèrent comme une langue seconde. Extraire des informations à partir d'un corpus arabe

devient un avantage pour les personnes spécialisées et les chercheurs dans la littérature arabes (études islamiques…).

Trop peu de travaux ont été réalisés pour étudier le problème d'extraction d'information à partir des documents écrits en arabe. Dans le cadre de ce travail, nous nous intéressons à l'extraction d'information à partir du texte en arabe. Concrètement l'extraction des motifs qui peuvent nous aider à analyser un texte et utilise ses motifs dans d'autre travaux de recherche, en se basant sur les techniques d'apprentissage machine.

Le reste du rapport sera divisé en 5 chapitres suivis d'une conclusion générale :

- **Chapitre 1** : intitulé par « État de l'art », ce chapitre est dédié à un État de l'art sur les techniques d'intelligente artificielle liées au domaine de fouille de texte en se basant sur les étapes du Text Mining, et plus particulièrement sur l'extraction de motifs.
- **Chapitre 2** : intitulé par « Travaux de fouille de texte en arabe » , dans ce chapitre nous présenterons des jeux de données (dataset) en arabe ainsi qu'une présentation des données et des expérimentations faites, terminant par une discussion des résultats.
- **Chapitre 3** : intitulé par « Présentation de jeu de données et algorithmes utilisé » l'intitulé parle de lui-même d'où ici nous présenterons le jeu de données que nous avons utilisé ainsi que les algorithmes(PrefixSpan, Fp-Growth, Apriori).
- **Chapitre 4** : intitulé par « Contributions » dans ce chapitre nous allons mentionner notre contributions (prétraitements et traitement).
- **Chapitre 5** : intitulé par « Discussion de résultats », ce dernier chapitre est dédié à une discussion de résultats.

En dernier, nous présenterons une conclusion générale.

Chapitre 1

État de L'art

1.1 Introduction

Dans ce chapitre nous allons expliquer les différentes technique d'IA dans le domaine de fouille de texte en citant les étapes de processus de ce dernier, en se basant sur l'extraction d'information.

1.2 Intelligence Artificielle

L'Intelligence Artificielle, un terme créé en 1955 par John McCarthy. **IA** décrit une machine capable de réaliser des processus de raisonnement qui sont habituellement réalisés par des humains. Le champ d'**IA** est tellement large que l'on a du mal à en mesurer la superficie. Ce champ couvre différentes disciplines dont la compréhension, le calcul, le raisonnement, l'apprentissage, la perception, le dialogue en langage naturel...etc.**[1]**

1.2.1 Apprentissage Automatique

Le Machine Learning (ML) aussi appelé apprentissage automatique est un sous-ensemble de l'IA. Tout ML est AI, mais toute IA n'est pas ML. Dans l'apprentissage automatique, la machine exécute quelque chose qu'on ne lui a pas commandé, elle a ses propres règles. Ce système permet donc à la machine d'apprendre de ses erreurs. Concrètement, on nourrit la machine de Big Data pour l'aider à «devenir intelligente» **[1]**

1.2.2 Le Traitement de Language Naturel (Natural Language Processing)

Le traitement du langage naturel (**NLP**) est une technologie d'intelligence artificielle visant à permettre aux ordinateurs de comprendre le langage humain.

L'objectif de cette technologie est de permettre aux machines de lire, de déchiffrer, de comprendre et de donner un sens au langage humain. D'importants progrès ont été effectués

dans ce domaine au fil des dernières années, et le **NLP** est aujourd'hui exploité pour une large variété de cas d'usage. Le **NLP** permet donc d'optimiser un certain nombre de processus dans des domaines différents.**[2]**

1.2.3 Fouille de Données

Fouilles de données, en règle générale, le terme Data Mining désigne 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. Ces informations peuvent ensuite être utilisées par les entreprises pour augmenter un chiffre d'affaires ou pour réduire des coûts. Elles peuvent également servir à mieux comprendre une clientèle afin d'établir de meilleures stratégies marketing.**[3]**

1.3 Fouille de Texte

La fouille de texte (aussi appelée analyse de texte, Text Mining en anglais) est une technologie d'intelligence artificielle (**IA**) qui utilise le traitement du langage naturel (**PNL**) pour transformer le texte libre (non structuré) dans les documents et les bases de données en données normalisées et structurées adaptées à l'analyse ou au pilotage d'algorithmes d'apprentissage automatique (**ML**). Donc le Text Mining consiste à examiner d'importantes collections de documents pour découvrir de nouvelles informations ou aider à répondre à des questions de recherche précises.

1.4 Data Mining vs Text Mining

Même si la fouille de données et la fouille de textes sont souvent considérées comme des processus analytiques complémentaires qui résolvent les problèmes opérationnels par l'analyse de données, elles diffèrent quant au type de données qu'elles traitent. Alors que DM traite des données structurées comme dans les bases de données ou les systèmes ERP , le TM traite des données textuelles non structurées, par exemple : des textes qui ne sont pas prédéfinis ou organisés de quelque façon que ce soit, comme dans les flux de médias sociaux. La figure suivante montre la déférence entre données structurées et données non structurées :

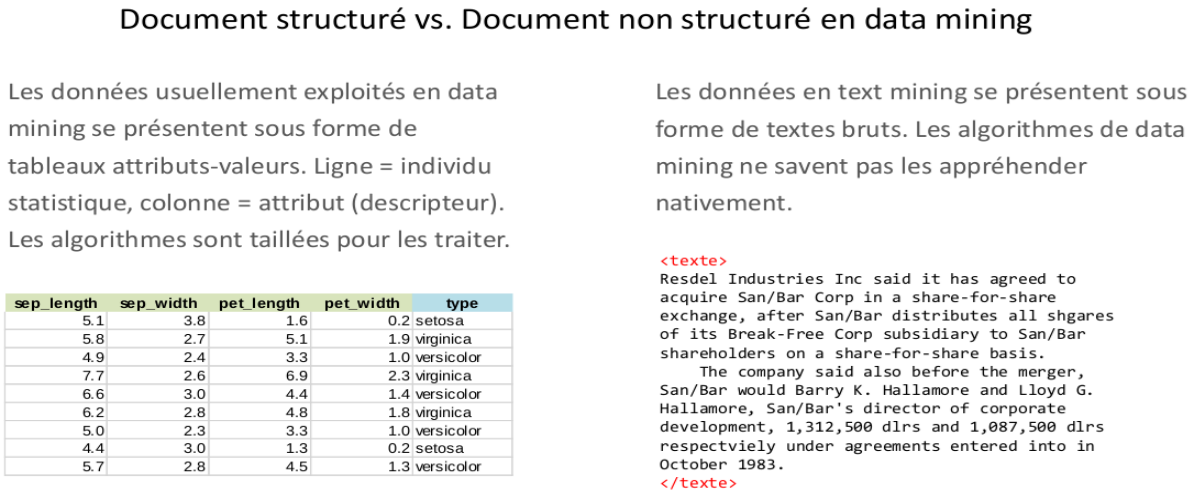

Figure 1 : Déférence entre données structurée et non structurée

1.5 Data Mining Vs Apprentissage machine (ML)

1.5.1 Les Points on commun:

L'exploration de données et l'apprentissage automatique relèvent tous deux de la science des données, ce qui est logique puisqu'ils utilisent tous deux des données.

Les deux processus sont utilisés pour résoudre des problèmes complexes, de sorte que, par conséquent, de nombreuses personnes (à tort) utilisent les deux termes de façon interchangeable. Ce n'est pas si surprenant, étant donné que l'apprentissage automatique est parfois utilisé comme un moyen de mener des recherches de données utiles. Alors que les données recueillies à partir de l'exploration de données peuvent être utilisées pour enseigner les machines, les lignes entre les deux concepts deviennent un peu flou. [4]

1.5.2 Leurs différences

Leur âge: Pour commencer, l'exploration de données précède l'apprentissage automatique de deux décennies

Leur buts: L'exploration de données est conçue pour extraire les règles à partir d'une grande quantités de données, tandis que l'apprentissage automatique enseigne à un ordinateur comment apprendre et comprendre les paramètres donnés.

Ce qu'ils utilisent: L'exploration de données repose sur de vastes réserves de données (par ex: les Big Data), qui servent ensuite à faire des prévisions pour les entreprises et d'autres

organisations. Le machine learning, par contre, fonctionne avec des algorithmes, pas des données brutes. [4]

1.5.3 Le Lien entre eux

Le Data Mining est un processus qui intègre deux éléments : la base de données et l'apprentissage automatique. Le premier fournit des techniques de gestion des données, tandis que le second fournit des techniques d'analyse des données. Ainsi, alors que le DM a besoin de l'apprentissage automatique. Le ML n'a pas nécessairement besoin de DM. Cependant, il y a des cas où l'information provenant de l'exploration de données est utilisée pour voir les liens entre les relations. Après tout, il est difficile de faire des comparaisons à moins d'avoir au moins deux éléments d'information à comparer les uns aux autres. Par conséquent, l'information recueillie et traitée par le Data Mining peut ensuite être utilisée pour aider une machine à apprendre, mais encore une fois, ce n'est pas une nécessité. Voyez ça comme une commodité à avoir. [4]

1.6 NLP vs Text Mining

NLP fonctionne avec tout produit de la communication humaine naturelle, y compris le texte, la parole, les images, les signes, etc. Il extrait les significations sémantiques et analyse les structures grammaticales que l'utilisateur saisit. Le fouille de textes fonctionne avec des documents textuels. Elle extrait les caractéristiques des documents et utilise une analyse qualitative. NLP fournit la compréhension des sentiments décrits, la structure grammaticale et le sens sémantique. Ces résultats permettent une traduction fluide du texte vers d'autres langues. L'exploration de texte montre les relations entre les mots du texte.

1.7 Tâches principales dans la Fouille de Textes

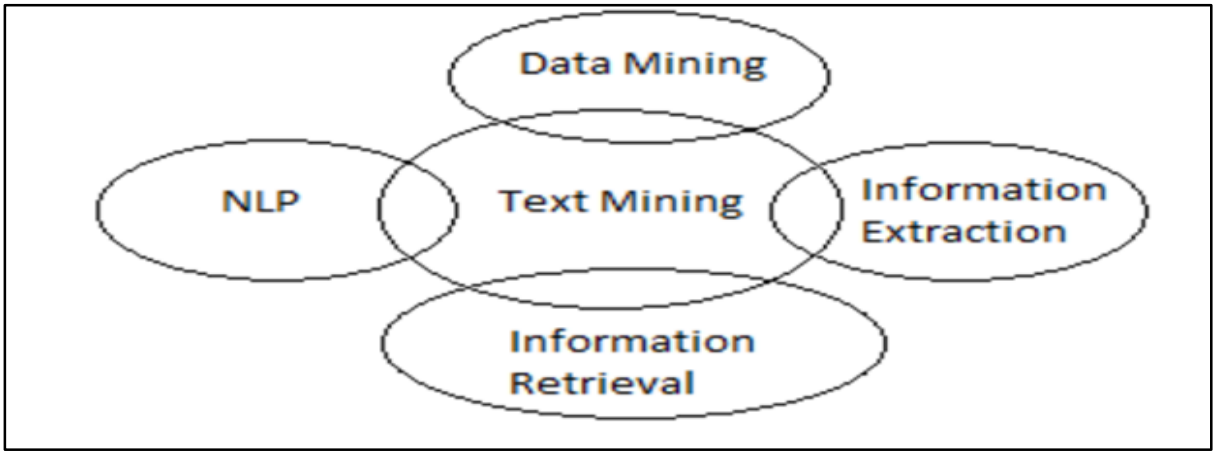

Figure 2 : Les approches liée au Text Mining. [5]

1.7.1 La recherche d'information (RI)

La recherche d'informations est considérée comme une extension du document récupération où les documents retournés sont traités pour condenser ou extraire les informations particulières recherchées par l'utilisateur. Ainsi, la récupération de documents pourrait être suivie d'un texte étape de synthèse qui se concentre sur la requête posée par l'utilisateur, ou une étape d'extraction d'informations utilisant des techniques. RI aide à réduire l'ensemble de documents qui sont pertinentes à un problème particulier. Comme l'extraction de texte implique l'application d'algorithmes très complexes pour grandes collections de documents, RI peut accélérer l'analyse considérablement en réduisant le nombre de documents pour analyse.

1.7.2 La tâche de fouille de "données" dans le fouille de "texte"

L'objectif global du processus d'exploration de données(DM) est d'extraire des informations d'un ensemble de données et de les transformer en une structure compréhensible pour une utilisation ultérieure.

1.7.3 le NLP dans le fouille de texte

Le rôle du PNL dans l'extraction de texte est de fournir le système à la phase d'extraction de l'information comme input.

1.7.4 L'extraction d'information

L'**EI** est une technique d'analyse de texte qui extrait des éléments de données spécifiques d'un texte, comme des mots clés, des noms d'entités, des adresses, des courriels, etc. En utilisant l'extraction de texte, les entreprises peuvent éviter tous les tracas de trier leurs données manuellement pour extraire des informations clés.

Ci-dessous, nous nous référerons à certaines des principales tâches de l'extraction de texte :

A-Extraction de mots clés (KeyWord Extraction) : mots clés sont les termes les plus pertinents dans un texte et peuvent être utilisés pour résumer son contenu. L'utilisation d'un extracteur de mots clés vous permet d'indexer les données à rechercher, de résumer le contenu d'un texte ou de créer des nuages de balises, entre autres choses. **[6]**

B-Named Entity Recognition : vous permet d'identifier et d'extraire les noms d'entreprises, d'organisations ou de personnes d'un texte**. [6]**

C. Extraction de motifs : vous permet d'extraire des connaissances à partir de données volumineuses, notamment sous la forme de motifs, grâce au développement d'algorithmes adaptés au format des données (ensembliste, relationnel, séquentiel, etc.)**[7].**

C.1. Les motifs séquentiels : Dans le contexte du NLP, il est important de prendre en compte la position d'un mot dans une phrase ou la position d'une phrase dans un paragraphe et ainsi de suite. Par exemple, "I want to see this film, do not tell me the end" n'a pas du tout le mème sens que "I do not want to see this film, tell me the end". Pour permettre cette notion de séquentialité, plus communément appelée notion de temporalité, nous utilisons les motifs séquentiels . L'extraction de motifs séquentiels rajoute une dimension temporelle à l'extraction de motifs ensemblistes. Historiquement, cette notion vient du besoin d'analyse et d'extraction de règles du problème bien connu du panier de la ménagère, soit l'étude des habitudes d'achats des clients. Pour le NLP, cela permet de trouver par exemple des motifs grammaticaux dans une phrase, comme un sujet suivi d'un verbe suivi d'un complément. Cela s'applique à de nombreux domaines, par exemple la découverte de motifs, utilisés comme patron linguistique, montrant la relation entre gène et maladie. Depuis leur introduction , les motifs séquentiels sont toujours définis de la manière suivante. **[7]**

- ✓ Soit $I = \{i1, i2, \ldots, in\}$ un ensemble fini de littéraux appelés *items*.
- ✓ Un *itemset* est un ensemble non-ordonné d'*items* distincts.
- ✓ Une séquence S sur I, de longueur $|S|$, est une liste ordonnée $<it_1,\ldots,$ $it_{|s|}>$ où les *it* sont des *itemsets* composés d'items de I.
- ✓ Une *k*-séquence est une séquence de *k itemsets* (i.e. une séquence de longueur *k*).
- ✓ $S[i, j]$ représente la sous-séquence qui commence à partir de *iti* jusqu'à *itj*. Un cas particulier est $S[0, j]$ qui représente la *j*séquence identifiée comme un préfixe de longueur *j* de la séquence S.
- ✓ $T(I)$ représente l'ensemble de toutes les séquences possibles sur I.
- ✓ Une base de séquences annotées D sur I est un ensemble ni de transactions (SID, T), avec $SID \in \{1, 2, \ldots\}$ un identifiant et $T \in T(I)$ une séquence sur I.

C.2. La représentation condensée des motifs

L'extraction de motifs fréquents pose encore aujourd'hui un problème quant à l'utilité des motifs fréquents extraits. En effet selon les paramètres utilisés lors de la fouille, les résultats peuvent être trop génériques pour avoir une quelconque valeur ou être trop nombreux pour pouvoir être traités par des experts. C'est pour cela que les représentations condensées ont émergé. Elles permettent de limiter le nombre de motifs extraits tout en gardant la même puissance d'expression.**[7]**

Depuis, la plupart des travaux portent sur les motifs ensembliste (non-séquentiels) principalement parce qu'il existe des relations fortes entre les motifs ensemblistes et de puissants outils mathématiques comme la théorie des ensembles, la combinatoire et les correspondances de Galois. Ces outils jouent un rôle important dans la construction des représentations condensées fondées sur les motifs clos , les motifs essentiels [Casali et al., 2005], les motifs δ-libres [Boulicaut et al., 2003a] (également appelés clés ou générateurs dans le cas particulier où $\delta = 0$) et les motifs non-dérivables [Calders and Goethals, 2007]. Quelques-unes des représentations condensées les plus utilisées :

i. **Motifs maximaux :** Un itemset *iti* est dit maximal si *iti* est fréquent et s'il n'existe pas d'itemsets fréquents *itj* tels que $iti \subset itj$. Cette représentation condensée est dite approximative. En effet, elle ne permet pas de calculer précisément le support des itemsets inclus dans le motif maximal. Mais elle permet de dériver une borne inférieure

sur le support d'un itemset : le support de chaque motif inclus dans *iti* apparait autant ou plus souvent que *iti*. Cette représentation condensée s'étend simplement au cadre des motifs séquentiels.[7]

Exemple motifs maximaux:

1-Supposons que le nombre de support minimum est 2, pour le premier exemple, supposons qu'il y ait un total de **3** éléments : **a, b, c**. et qu'un motif **ab** a un nombre de support de **3** et un motif **abc** a un nombre de support de **2**. Le motif **ab** est-il un motif max ? , le motifs **ab** est un modèle fréquent, mais il a un super-motif qui est fréquent ainsi. Donc, le motif ab n'est pas un motif max.

2- Supposons qu'il y a un total de **3** éléments : **x, y, z**. Supposons qu'un motif **xy** a un nombre de support de **3** et qu'un motif **xyz** a un nombre de support de **1**. Le motif **xy** est-il un motif max ?,le Motif **xy** est un motif fréquent et aussi le seul super-motif **xyz** n'est pas un motif fréquent. Par conséquent, **xy** est un motif max.

ii. **Motifs fermés** : Basé sur l'opérateur de fermeture délimitant les classes d'équivalence, l'itemset fermé sera l'unique plus grand élément de chaque classe d'équivalence. Un itemset *iti* est donc dit fermé si et seulement si il n'existe pas d'itemsets tels que $iti \subset itj$ et Support(*iti*) = Support(*itj*). Grâce à cette propriété, il est possible de régénérer de manière exacte l'ensemble des itemsets fréquents à partir uniquement de l'ensemble des itemsets fré- quents fermés extraits préalablement. Cette représentation condensée est un peu plus compliquée dans le cadre des motifs séquentiels, les auteurs de [Yan et al., 2003] s'intéressent donc aux motifs séquentiels fermés et propose cette définition : Soit σ, le support minimal et $FSeqs(D, \sigma)$ l'ensemble des motifs séquentiels fréquents correspondants.

L'ensemble des motifs séquentiels clos $CFSeqs(D, \sigma)$ est défini comme :

$$CFSeqs(D, \sigma) = \{ s \mid s \in FSeqs(D, \sigma) \wedge 6\, \exists s0 \mid s \ \ s0 \ avec \ Support(s) = Support(s0)\}$$ [7]

Exemple motifs fermés :

1-Supposons que le nombre de support minimum est 2. Pour le premier exemple, supposons qu'il y ait un total de **3** éléments : **a, b, c**, et qu'un motif **ab** a un nombre de support de 2 et un motif **abc** a un nombre de support de **2**. Le motif **ab** est-il un motif fermé ? le motif **ab** est fréquent, mais il a **un super-motifs(abc)** qui n'est pas moins fréquent que **ab**.

2-supposons qu'il y a un total de **3** éléments : **x, y, z**. supposons qu'un motif **xy** a un nombre de support de 3 et qu'un motif **xyz** a un nombre de support de **2.** Le motif **xy** est-il un motif **fermé** ?,oui le Motif **xy** est un motif fréquent et aussi le seul super-motif **xyz** est moins fréquent que xy, Par conséquent, xy est un motif fermé.

iii. **Motifs libres** : Les motifs libres (ou générateurs) sont les motifs minimaux à fréquence égale. Dans les classes d'équivalences, les motifs libres sont les plus petits itemsets. Un itemset *iti* est dit libre si son support est distinct de celui de chacun de ses sous-ensembles. L'extraction des motifs libres est par contre très difficile dans le cadre des séquences. En effet, nous perdons la propriété clé des techniques d'élagage en extraction de motifs : l'anti monotonie. Cette perte souligne la déférence fondamentale entre séquences et motifs ensemblistes pour leur extraction et la complexité algorithmique qui va en résulter. Ainsi, avec les séquences, il devient impossible d'utiliser des mécanismes qui sont efficaces pour les motifs ensemblistes, tels que l'inférence du support de certains motifs. C'est pour cette raison que relativement peu de travaux ont été effectués sur les motifs libres séquentiels. On citera notamment [Soulet and Rioult, 2014] qui ont présenté un cadre générique et efficace pour l'extraction de motifs minimaux qui, grâce à la notion de système minimisable d'ensembles, permet d'extraire des chaînes minimales. Cependant, ces chaînes minimales sont des sous-segments contigus**.[7]**

C.3. Les algorithmes d'extraction de motifs

Nous allons maintenant voir les algorithmes d'extraction de motifs les plus importants. Ces algorithmes diffèrent par leurs manières de parcourir l'espace de recherche et sur les structures de données utilisées.

Algorithme GSP et PSP

L'algorithme GSP (Generalized Sequential Patterns) est le pionnier de l'extraction de motifs séquentiels. Les auteurs en ont déni la problématique et ont proposé un algorithme « générer-élaguer » utilisant les bases d'Apriori mais avec une phase de génération adaptée aux séquences. Pour générer les motifs de taille k+1, au lieu de simplement utiliser les motifs fréquents de taille k, GSP fait une opération de jointure sur l'ensemble des motifs fréquents de taille k-1. La jointure a lieu s'il y a concordance entre deux sous-séquences après la suppression du premier item de la première séquence testée. Cette opération est illustrée en Figure suivante.**[7]**

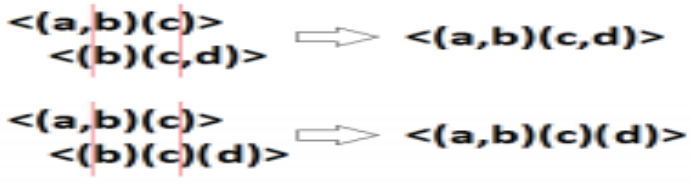

Figure 3 : Opération de jointure

GSP utilise une structure d'arbre de hachage pour parcourir les candidats. Il stocke tous les candidats potentiels en fonction de leur préfixe, ce faisant une feuille de l'arbre contient plusieurs candidats. Lors du parcours de l'arbre, on perd donc la relation entre les motifs de taille k et k-1. On ne peut donc pas savoir si un motif a été généré grâce à une S-extension ou une I-extension.

PSP (Prefx Tree for Sequential Pattern) reprend GSP en remplaçant cette structure de données par un arbre préfixé. Chaque nœud de l'arbre est un item et uniquement un item, mais il existe deux types de branches : l'un reliant deux items d'itemsets déférents, et l'autre reliant deux items d'un même itemset. On peut alors reconstituer une séquence en partant de la racine et en descendant. Cette structure améliore nettement la performance par rapport à GSP.[7]

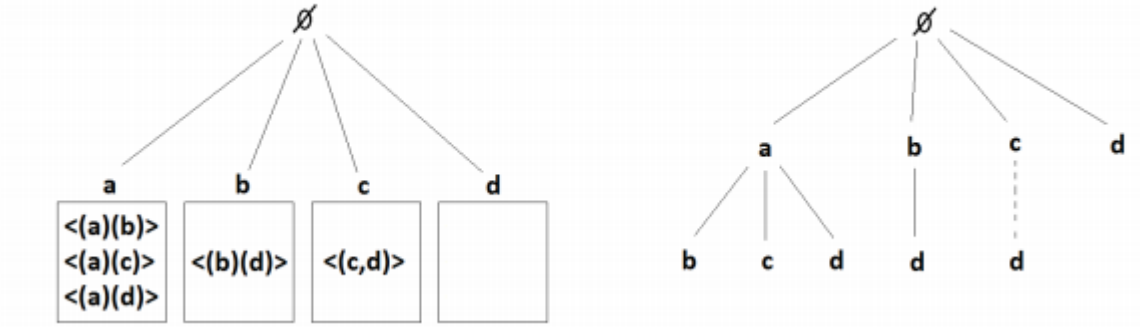

Figure 4: Comparaison entre les structures de données de GSP à gauche et PSP à droite.

Algorithme Spade

L'algorithme SPADE (Sequential Pattern Discovery using Equivalence classes) et sa variante cSPADE (constrained SPADE, [Zaki, 2000]) utilisent les propriétés et les techniques de recherche des treillis. La caractéristique principale de SPADE est la représentation verticale de la base de données tout en décomposant l'espace de recherche en sous-treillis selon des classes d'équivalence pour extraire plus e-cacement les motifs séquentiels. Cela consiste à inverser la méthode d'indexation de la base de données. La génération de candidats se fait par opération de jointures entre les différents éléments des classes d'équivalences. La transformation revient à associer à chaque k-séquence l'ensemble des couples (SID, EID) qui lui correspondent dans la base. Le support d'une séquence est le cardinal de l'ensemble constitué par les identifiants de séquences**. [7]**

Algorithme FreeSpan

L'algorithme FreeSpan (Frequent pattern projected Sequential Pattern mining, [Han et al., 2000a]) a pour idée générale d'utiliser des projections de la base de données. C'est le paradigme de FP-Growth (frequent pattern growth) qui est une alternative à Apriori. En effet, l'utilisation de base projetée permet d'éviter l'étape coûteuse de génération et d'élagage de motifs candidats en compressant la base de données représentant les séquences fréquentes en un arbre de motifs

fréquents et en divisant cet arbre en un ensemble de bases projetées qui seront fouillées séparément. [7]

1.8 Le processus de fouille de texte

1.8.1 Prétraitement de texte [17]

- **Text normalisation :** Ce processus implique la conversion des données dans un format standard. Ici, le texte entier est converti en majuscules ou en minuscules, les chiffres, la ponctuation, les accents, les espaces blancs, les mots stop et autres signes diacritiques sont supprimés. Python peut être utilisé pour implémenter ceci.
- **Tokenization :** Dans ce processus, le texte entier est divisé en petites parties appelées jetons. Les nombres, signes de ponctuation, mots, etc. peuvent être considérés comme des jetons(tokens). Natural Language Toolkit (NLTK), Spacy et Gensim sont quelques outils qui peuvent être utilisés pour la tokénisation.
- **Stemming :** C'est le processus de réduction des mots à leur forme de tige, de base ou de racine. Les deux principaux algorithmes utilisés pour ce processus sont l'algorithme de Porter et l'algorithme de Lancaster. NLTK ainsi que Snowball peuvent être utilisés pour cela.
- **Lemmatisation :** Le but de la lemmatisation, comme le stemming, est de réduire les formes inflectives à une forme de base commune. Mais, par rapport à l'endiguement, la lemmatisation ne supprime pas simplement les inflexions. Au lieu de cela, il utilise des informations provenant de différents dépôts informatiques pour obtenir les formes de base correctes des mots.
- **Part of speech tagging :** Il vise à assigner des parties de discours à chaque mot d'un texte donné en fonction de sa signification et de son contexte. NLTK, spaCy, Pattern sont quelques logiciels qui peuvent être utilisés pour cela.
- **Chunking :** C'est un processus de langage naturel qui identifie les parties constitutives des phrases et les relie à des unités d'ordre supérieur qui ont des significations grammaticales discrètes. NLTK est un bon outil pour cela.
- **Named Entity recognition :** Il vise à trouver des entités nommées dans le texte et à les classer dans des catégories prédéfinies. NLTK, spaCy peut être utilisé, Ex:

«دايملر» تستثمر في التاكسي الطائر «فولوكوبتر»

القاهرة: «الشرق الأوسط» أونلاين»

قالت شركة «فولوكوبتر» الألمانية إنها حصلت على 25 مليون يورو (30 مليون دولار) لتمويل مشروع تصنيع سيارة أجرة طائرة تعمل بالكهرباء، إذ انضمت شركة «دايملر» لصناعة السيارات والشاحنات إلى شركة تقدم أموالا جديدة.

وقالت «فولوكوبتر» أمس (الثلاثاء) إن «دايملر» انضمت إلى كونسورتيوم يضم لوكاس جادوسكي المستثمر التكنولوجي وآخرين.

وتقول الشركة إنها تطور مركبة كهربائية ذات خمسة مقاعد يمكنها الإقلاع والهبوط بهدف دخول سوق سيارات الأجرة وتعتزم إجراء الاختبارات الأولية للسيارة الجديدة في الربع الأخير من العام الحالي.

والمنافسون المحتملون لـ«فولوكوبتر» هم ليليوم جيت وإيفولو الألمانيتان وتيرافوجيا وجوبي أفييشن ومقرهما الولايات المتحدة.

كما تطور شركة إيرباص «سيارة طائرة» ذات مقعد واحد.

Entity	Type	Subtype		Appearances	Relevance
دايملر	Organization	AutomobileCompany		3	100
القاهرة	Location	City		1	7
الشرق الأوسط	Location	GeoPoliticalEntity		1	7
الولايات المتحدة	Location	Country		1	7
إيرباص	Organization	AerospaceDefense		1	7

Figure 5 : un exemple de reconnaissance des entités nommées.

- **Relationship extraction:** Cela aide à identifier les relations entre les entités nommées comme les personnes, les organisations, etc. Il permet d'obtenir des informations structurées à partir de sources non structurées telles que le texte brut.

1.8.2 Text tronsformation

bag of words : En général, il s'agit d'un ensemble de mots pour représenter une phrase avec le nombre de mots et la plupart du temps en ignorant l'ordre dans lequel ils apparaissent.

vector space : est un modèle algébrique pour représenter des documents texte (et tout objet, en général) comme vecteurs d'identificateurs.

1.8.3 Feature selection

La sélection de caractéristiques est également connue sous le nom de sélection d'attributs ou de sélection de variables. C'est la sélection des caractéristiques les plus pertinentes à partir des variables disponibles qui donne le plus d'information pour vos variables de prédiction. Les caractéristiques non pertinentes peuvent accroître la complexité et diminuer l'exactitude de l'analyse. Le coefficient de corrélation de Pearson, Chi — squared, élimination des caractéristiques récursives, régression au lasso, algorithmes basés sur les arbres sont quelques méthodes qui peuvent être utilisées pour cela. Python peut être utilisé pour toute l'analyse.

1.8.4 Data Mining

Ici, nous combinons le processus d'extraction de texte avec les techniques traditionnelles d'extraction de données. Une fois les données structurées après les processus ci-dessus, les techniques classiques de fouille de données sont appliquées sur les données pour extraire l'information. Ces techniques comprennent la classification, le regroupement, la régression, les motifs externes et séquentiels, les règles de prédiction et d'association.

1.8.5 Evaluation

Après l'application des techniques d'exploration de données, nous obtenons un résultat final qui doit être évalué et vérifié pour l'exactitude de la prévision.

1.9. Conclusion

A travers ce chapitre nous avons pu définir les techniques d'IA utilisées dans le fouille de texte ainsi que les différentes étapes de processus de **TM**. Nous avons parlé aussi de l'extraction de motifs en général et les algorithmes utilisé dans la fouille de motifs, on peut déduire qu'il existe de nombreux travaux qui utilisent les motifs séquentiels en NLP. L'approche la plus symbolique est l'utilisation des motifs fréquents comme patrons linguistiques dans le but de découvrir de nouvelles constructions linguistiques sans utiliser de connaissances a priori. L'objectif est d'aider les experts linguistiques, en leur fournissant des motifs pertinents et compréhensibles qui caractérisent un genre de texte, afin qu'ils puissent réaliser une analyse en s'appuyant sur ces motifs. Le chapitre suivant est dédié au fouille de texte arabe et les travaux réalisé dans ce domaine.

Chapitre 2

Travaux de fouille de texte en arabe

2.1 Introduction

Dans la suite de ce chapitre nous allons parler des techniques de fouille de texte utilisés dans la littérature arabe ainsi que les travaux réalisé dans cette dernière.

2.2 Text Mining en arabe

L'arabe est la première langue sémitique en nombre de locuteurs, elle est parlée et écrite dans le monde arabe par plus de 360 millions et utilisée dans les cultes par 1.6 milliards de musulmans soit 23,4 % de la population mondiale. Elle est classée dans les dernières années comme quatrième langue mondiale en nombre d'utilisateurs dans l'Internet Utilisée par plus de 22 pays dans le monde, la langue arabe est parlée en plusieurs dialectes non écrites. La seule langue arabe utilisée par les administrations étatiques, dans les médias officiels et dans l'éducation du monde arabe c'est l'Arabe Moderne Standard (ASM). **[8]**

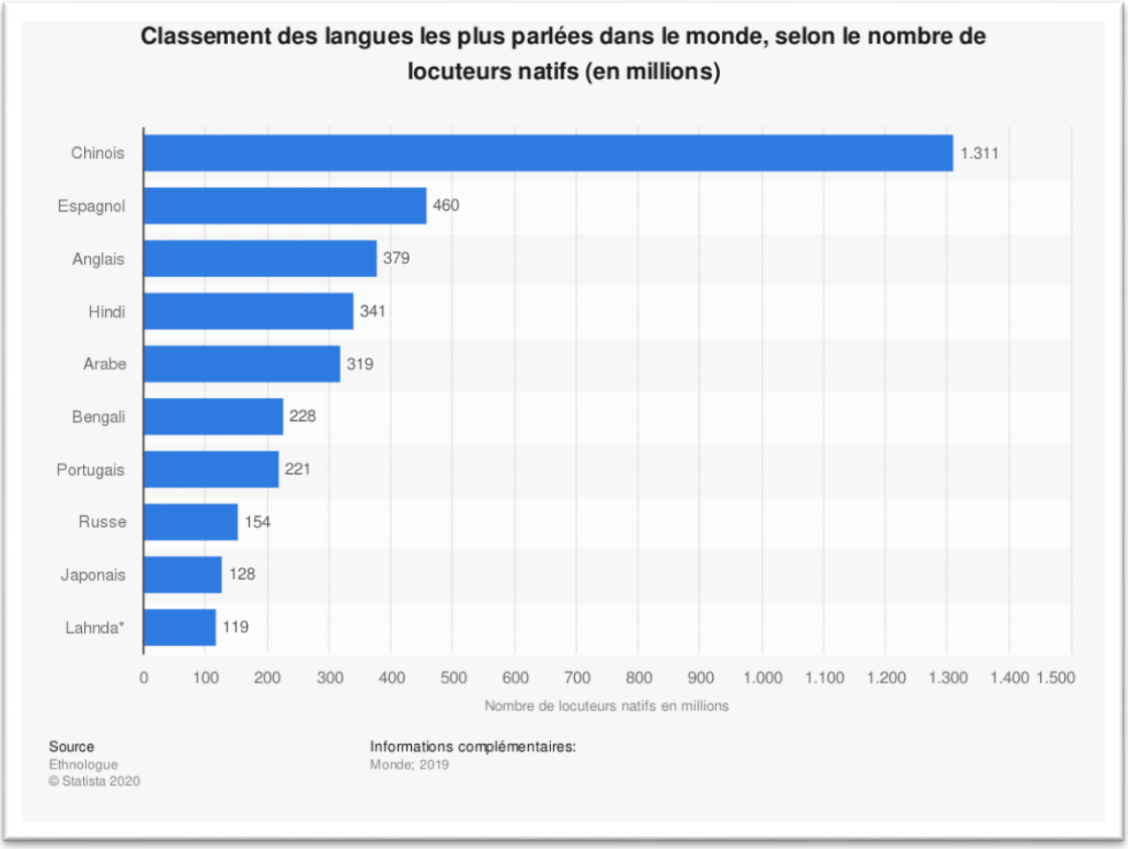

Figure 6 : Les dix langues les plus utilisées sur Internet en 2020.

2.2.1 Le système d'écriture

L'arabe s'écrit de droite vers la gauche en utilisant deux types de caractères : les lettres et les diacritiques. Les lettres arabes renferment les consonnes et les voyelles longues alors que les diacritiques servent à vocaliser le texte i.e. déterminer la phonétique de ses mots. Quelques diacritiques à savoir فـتـحـة\fat.Haћ\6, ضمـة\Dam~aћ\ et كـسـر ة\kas.raћ\ sont considérés comme des voyelles courtes.**[8]**

Dans ce système d'écriture les lettres sont obligatoires alors que les diacritiques sont optionnelles : l'omission des diacritiques dans le texte arabe augmente le degré d'ambiguïté pour les analyseurs morphologiques et syntaxiques. [8]

Dans la littérature du NLP, il y a un désaccord sur le nombre de lettres arabes, ce désaccord est dû à la classification de ce qui est ou non diacritique et à l'ignorance de certaines lettres. Généralement on parle de 28 lettres mais les nombres 36 et 40 sont considérés aussi dans la littérature du domaine (Buckwalter 2004b; N. Habash 2010). Dans notre travail on considère l'alphabet de 36 lettres car il correspond au tableau de l'Unicode arabe7 et il est compatible avec les caractères du clavier arabe.

Les 36 lettres de l'ASM sont classées comme suit : 28 lettres de base, 6 formes possibles de la lettre Hamza, la lettre Ta-Marbouta et la lettre Alif-Maqsoura.[8]

En plus de ce nombre de lettres, on trouve dans les textes d'ASM autres lettres non arabe telles que پ, چ, گ, pour représenter une phonétique non arabe correspondant aux lettres latines (p, j, v, g) qui n'existent pas originellement en ASM (Buckwalter 2004b). Ces lettres sont empruntées aux alphabets des langues écrites en alphabet arabe8, tel que le persan/farsi. En arabe les diacritiques sont des symboles optionnels dans l'écriture, i.e. que l'écriture arabe peut se trouver complètement, partiellement ou non discrétisée. L'ajout de diacritiques dans un texte a pour but d'aider le lecteur à lire et prononcer correctement les mots.

2.2.2 Morphèmes à schème

Les morphèmes à schème sont de trois types qui sont tous aussi nécessaires pour créer une base d'un mot à schème : **racines, patrons et vocalisme.**[8]

- **La racine :** Le morphème racine est une séquence de trois, de quatre ou très rarement de cinq consonnes (appelées radicaux) (N. Habash 2010). La racine signifie un sens abstrait partagé par tous ses dérivés. En effet, à chaque racine correspond un champ sémantique et à l'aide de différents patrons, on peut générer une famille de mots appartenant à ce champ sémantique, par exemple la racine ك-ت-ب peut engendrer quinze mots autour de la notion de écriture. [8]

- **Le patron :** Le morphème patron est un modèle abstrait dans lequel les racines et vocalisme sont insérés, on représente le patron comme une chaîne de caractères y compris des symboles spéciaux pour marquer l'emplacement des radicaux et vocalismes où seront insérés. On utilise les numéros 1, 2, 3, 4 ou 5 pour indiquer les positions des radicaux et le symbole V est utilisé pour indiquer la position du vocalisme. Par exemple, le patron 1V22V3 indique que le second radical de la racine doit être doublé. Un schème peut inclure des lettres pour les consonnes et les voyelles supplémentaires, par exemple, le patron verbal V1tV2V3 [8]

- **Le vocalisme :** Le morphème vocalisme spécifie les voyelles courtes à utiliser avec un schème. Les descriptions traditionnelles de la morphologie arabe incluaient le vocalisme dans le patron. La séparation de vocalisme a été introduite avec l'émergence de schèmes plus sophistiqués qui font abstraction de certains traits flexionnels qui varient constamment avec les patrons complexes, tels que la voix (passif ou actif) Une base de mot est construite par l'entrelacement des trois types de morphèmes à schème. Par exemple, la base de mot كتب \kataba\ écrire est construite à partir du triplet (ك-ت-ب1v2v3, aa) de la racine ك ت ب , du patron 1v2v3 et du vocalisme aa.[8]

2.2.3 les utilisations de fouille de texte en arabe

Bien que l'arabe soit une langue largement utilisée, il y a encore très peu de recherche effectuée sur la récupération ou l'extraction de documents écrits dans la littérature arabe.

Clairement la raison c'est les règles morphologiques uniques de la langue. C'est un langage difficile en raison de nombreux facteurs. L'orthographe à côté des diacritiques comme on a mentionné ci-dessous tend à être plus phonétique et très clair en arabe et certaines combinaisons des lettres peuvent être écrites de façons uniques.**[9]**

le TM en arabe est souvent utilisé dans : le Coran, l'analyse des sentiments et les documents web etc.

2.2.3.1 Quran text mining

Le Coran est le livre de référence pour plus de 1,6 milliard de musulmans dans le monde entier. Extraire des informations et des connaissances du Coran est un avantage élevé pour les personnes spécialisées dans les études islamiques ainsi que pour les personnes non spécialisées. Le Coran est la parole d'Allah et doit donc être manipulé avec soin lorsqu'il est traité par des méthodes automatisées d'apprentissage automatique, de traitement du langage naturel(NLP) et d'intelligence artificielle(AI)**.[9]**

Le Coran a environ 78 mille mots. Ces mots sont regroupés en versets. Un ensemble de versets sont regroupés en : parties, chapitres, groupe ou quart (Hizb). Découvrir et extraire d'intéressant, non négligeable connaissance du Coran en appliquant les méthodes de fouille de texte par exemple les plus utilisé ces derniers années sont: Information extraction, text categorization, Clustering, concept linkage, discovery of associations,Vector space model, The pronoun tagging et the part-of-speech tags(POS)

2.2.3.2 Analyse du sentiment du texte arabe (Opinion Mining)

Opinion mining est le processus de classification des opinions négatives et positives qui sont présentées sous forme de texte. L'extraction des émotions, d'autre part, est liée à la différenciation entre les différentes émotions (ex. heureux, en colère et triste). La tâche se concentre sur l'analyse des sentiments en fonction de l'exploration d'opinion. L'objectif principal de l'analyse des sentiments est d'examiner les émotions des personnes, leurs opinions, leurs comportements, leurs sentiments et leurs jugements au sujet d'entités comme les problèmes, les occasions, les personnes, les entreprises et les sujets**.[9]**

Bien que les outils du PNL arabe aident à fournir des caractéristiques linguistiques et grammaticales du texte, les ressources sur le sentiment en arabe qui ont été développées

pour fournir une compréhension plus approfondie du sentiment et des aspects sémantiques du texte sont : Sentiment Lexica et Sentiment Corpora. Extraire d'intéressant connaissances du twiter par exemple en appliquant les méthodes de fouille de textes, parmi les plus utilisé on peut trouver : Hybrid classifier, Lexical-based classifier, Feature extraction, Support Vector Machines classifier,Naive Bayes, et K-nearest neighbor classifiers. Bien que de bons progrès et des résultats obtenus pour l'analyse des sentiments arabes, il reste beaucoup à faire pour le dialecte arabe. Chaque pays arabe a son propre dialecte, mais en général, nous nous référons à des dialectes par région comme, par exemple, le Levantin, le Golfe, l'égyptien et le nord-africain. Chaque région possède des caractéristiques différentes, ce qui rend la tâche d'analyse des sentiments encore plus difficile. Par exemple, l'expression يعطيك العافية est un complément positif dans le dialecte levantin signifiant « Qu'Allah vous accorde la santé », alors qu'il a une connotation négative dans les dialectes marocain et tunisien, « brûler dans le feu ».De plus, Les données de Twitter comprennent de nombreux néologismes, abréviations, émojis, élongations, négations, intensificateurs, changements de polarité et sarcasmes**. [9]**

2.2.3.3 Text Mining dans les documents Web

La quantité d'information disponible sur le Web (Sites, Pages Facebook, Article web..) a été considérablement augmentée à l'heure actuelle. Alors qu'une grande partie de l'information textuelle existe sur le web, il est facile pour l'utilisateur de récupérer les informations de manière appropriée uniquement si l'utilisateur comprend la langue qui est utilisé dans le document Web. Il est significatif dans la situation où les utilisateurs essaient de trouver certaines informations qui sont composées de scripts arabes et les mots mentionnés sont les mêmes dans différentes langues. En raison de cela, plusieurs chercheurs ont effectué des études afin de déterminer l'élément d'information présentes sur le document web. **[9]**

Les techniques d'extraction de l'information ont été appliqué aux documents en texte clair; mais extraction informations des pages web "HTML" présente un domaine de problème différent. La raison est que, les documents HTML contiennent de nombreuses balises (markup tags) qui peuvent trouver des informations utiles sur le Web. D'autre part, les pages Web sont également relativement peu structurée. Au lieu d'un document composé de paragraphes, une page Web peut être un document composé d'un navigation, tableaux avec données textuelles et numériques, phrases majuscules et mots répétitifs. Parmi les methodes de text mining les plus utilisées dans le web: Decision Tree classifier, K-Nearest Neighbor, NaïveBayes, et Arabic Text Classifier (ATC),etc. **[9]**

2.3 Travaux de fouille de texte en langue Arabe

2.3.1 ASTD : Arabic Sentiment Tweets Dataset

Lien : http://www.mohamedaly.info/datasets/astd

Description

Il s'agit d'un ensemble de tweets en arabe contenant plus de 10000 entrées. Ils sont classés dans l'une des quatre catégories suivantes : objectif, subjectif négatif, subjectif positif et subjectif mixte. Ce travail est réalisé conjointement avec Mahmoud Nabil et Amir Atiya, avec des données collectées par le premier auteur. Mahmoud Nabil, Mohamed Aly et Amir Atiya. ASTD : Arabic Sentiment Tweets Dataset, EMNLP, 2015. La figure suivante montre un exemple du dataset ASTD :

	Tweet	Translation	Rate
1	اكثر شعور يوجع ! ^#لما تجوع في بيت مو بيتكم ☺**	Feeling that hurts ^ ! #To starve ina house not yours	Negative
2	محبين البرنامج بيزيدوا :)	Fans of El-Bernameg are increasing :)	Positive
3	#كفاية اسفاف	#stop smallness	Negative
4	الطاقة البشرية اذا ما احسن استغلالها هى رصيدا و ليست عبئا قوتنا فى عددنا	Human energy if properly exploited is an asset and not a burden our strength in our numbers	Positive
5	احيي الشيخ حسن عبد البصير امام مسجد سيدى جابر الذي رفض تعليمات الأوقاف بنفاق مرسي في خطبة الجمعة تعلموا الاستقامة أيها #الاخوان الكاذبون	I greet Sheikh Hassan AbdelBassir Imam Sidi Gaber mosque,who refused the instructions of the endowments to hypocrite Morsi in his Friday sermon learn the integrity liars brotherhood	Mixed
6	هل يُتوج ايتكو مدريد بلقب الليجا الأحد القادم؟ #برشلونة	Is Atletico Madrid going to be crowned La Liga next Sunday? # Barcelona	Objective

Figure 7 : un exemple de classification d'une partie du dataset ASTD

2.3.2 AP-Data Set sur la poésie arabe

Lien : https://www.kaggle.com/fahd09/arabic-poetry-dataset-478-2017

Description

La poésie arabe est la forme la plus ancienne et la plus importante de l'arabe littérature d'aujourd'hui. La poésie arabe ancienne est probablement la vie sociale, politique et intellectuelle dans le monde arabe. la poésie moderne a connu des changements majeurs et des changements tant dans la forme que dans les sujets.

Contenu

L'ensemble de données contient plus de 58K poèmes qui s'étendent du 6ème siècle a nos jours. Avec chaque poème, le nom du poète, le poème et sa catégorie. Les données ont été gratté sur adab.com.

Citation

Ce jeux de données étais citer dans 2 article :

1-**Arabic Poetry: Classification based on Era** : dans cet article, ils ont proposés un modèle d'apprentissage profond basé sur CNN qui classifie les poèmes arabes en fonction de son époque, ce qui n'est jamais rapporté auparavant. Pour construire ce modèle, la construction d'un ensemble de données est la première étape, ils ont utilisés donc un ensemble de données actualisé sur la poésie arabe (2020). En utilisant des intégrations de mots FastText, basées sur le corpus complet de poèmes (non étiquetés). Deux classificateurs ont été formés, à savoir un classificateur d'apprentissage profond supervisé et un classificateur basé sur FastText. Ils ont effectués plusieurs expériences. Tout d'abord, la mise en œuvre d'un classificateur de polarité des poèmes aux époques modernes et non modernes, qui a atteint la plus grande précision et F1-score de 0,913 et 0,914, respectivement, en utilisant un modèle d'apprentissage profond sans termes fréquents. Dans la deuxième expérience, ils ont classé les poèmes en trois époques.

2-**Classification of Arabic Poems: from the 5th to the 15th Century :** cet article décrit un système de classification des poèmes arabes selon les époques où ils ont été écrits. Ils ont utilisés des techniques d'apprentissage automatique où ils ont appliqué un tas de filtres et de classificateurs. Les meilleurs résultats ont été obtenus en utilisant l'algorithme Multinomial Naïve Bayes (MNB), avec une précision égale à 70,21%, un F1-Score de 68,8% et un Kappa égal à 0,398, sans filtrer les mots stop. Ils ont observés que les mots d'arrêt peuvent avoir un impact positif sur l'exactitude, mais aussi un impact négatif s'ils sont utilisés avec le word tokenizer precessing.

2.3.3 ABMC Arabic in Business and Management Corpora Dataset

Lien :https://www.lancaster.ac.uk/~elhaj/corpora.html.

Description

Cet ensemble de données a été créé par El-Haj et al. à 2016, l'arabe dans L'ensemble de données des Sociétés de gestion (CGAA)

Contenu

1) **Corpus Management :** 400 articles des présidents et des PDG de l' entreprises arabe.
2) **Economie News :** 400 articles de différentes langues arabes des journaux en ligne.
3) **Actualités boursières** : 400 articles collectés sur investing.com.

Le corps principal contient 1200 articles. Les articles ont été étiquetés en utilisant Stanford arabe part of Speech Tagger. La ressource est accessible au public et a été utilisée dans le Corpus arabe Livre linguistique de l'Université King Saud.

2.3.4 Tashkeela Arabic diacritized text dataset

Auteur : Hamza Abbad.

Contenu

Une version de l'ensemble de données de texte arabe diacritisé de Tashkeela a été nettoyée à partir du contenu non rabique et du texte non acritisé, puis divisée en ensembles de formation, de développement et de test. Le processus de nettoyage comprend la suppression des balises XML et des symboles étranges, ainsi que la correction des erreurs diacritiques et l'unification des conventions contradictoires. Après cela, la tokenization est effectuée tout en se concentrant sur l'extraction des mots arabes. Le résultat est un fichier de jetons séparés par des espaces, où les mots et les nombres sont séparés, mais pas les séquences de ponctuation (c'est-à-dire une parenthèse de fin suivie d'un point). La segmentation de la phrase se fait à des ponctuations habituelles telles que des points, des virgules, des points d'interrogation/d'exclamation et la fin des lignes.

Le processus de partition se fait en mélangeant des groupes de phrases puis en divisant chaque groupe en trois parties (Train/Val/Test) et en distribuant le contenu sur de petits fichiers texte bruts de tailles allant de 1,5 Mo à 1,6 Mo pour faciliter le chargement.

Caractéristiques

- Textes bruts en arabe entièrement diacritisés.
- Plus de 3 millions de phrases avec un nombre différent de mots.

- Arabe pour la plupart classique.
- Space-separated tokens.
- 90 % de formation, 5 % de validation et 5 % de données d'essai.

2.3.5 Direct Arabic products, opinions data set for opinion mining

Lien : https://search.datacite.org/works/10.7910/dvn/ytswj4
Auteur : Sarah Saad
Description
Les opinions des produits dans l'ensemble de données Arabe sentiment sont collectées manuellement à partir de différentes ressources de produits sociaux pour l'exploration d'opinion, l'extraction de motifs et les tâches d'analyse de sentiments. Les opinions recueillies comprenaient différents types d'opinions directes qui comprennent au moins une caractéristique du produit, qu'elle soit explicite ou implicite. L'ensemble de données contient vingt catégories de produits différentes comme la maison, bébé, différents types de produits logiciels et d'autres types de produits. De plus, les caractéristiques des produits sont identifiées manuellement à partir des opinions des clients et de la description du produit. Les produits sont classés en fonction de chaque type de produit et il y a une recherche spécifique liée à chaque type.

Pour chaque produit, le nom du produit et une brève description des capacités du produit sont enregistrés dans le fichier d'information du produit et classés en fonction des types de produits spécifiques avec une requête initiale spécifique pour chaque type. Les données collectées contiennent des avis sur une vingtaine de catégories de produits différentes. Ces opinions sont choisies en fonction de la taille du texte et du nombre de caractéristiques qui apparaissent dans le texte avec des opinions divergentes. Pour chaque opinion, nous gardons une trace du texte d'opinion et de la note de sentiment entrée par les clients. La cote représente la polarité globale de l'examinateur à l'égard des produits dans l'une des deux catégories : sentiment positif ou négatif. Les principaux attributs de l'ensemble de données comprennent le nombre total d'opinions dirigées utilisées dans l'ensemble de données qui devraient comprendre au moins une caractéristique de produit explicite, le nombre d'opinions ayant un score de sentiment positif est de 1459 et le score de polarité de sentiment négatif est de 516.

2.3.6 Narrateurs d'El Hadith

Lien : https://www.kaggle.com/fahd09/hadith-narrators

Description

Hadith (un mot arabe) se réfère aux paroles et les actions du prophète Mahomet. Ces collections de hadiths ont été transmises à travers des générations de savants musulmans jusqu'à ce qu'elles aient été recueillies et écrites dans de grandes collections. La chaîne des narrateurs est un domaine d'étude principal dans la bourse islamique parce qu'un seul hadith peut avoir plusieurs chaînes de narrateurs (qui peut ou ne peut pas se chevaucher). Cependant, il est principalement resté un domaine qualitatif où les savants de Hadith tentent de déterminer l'authenticité des hadiths en étudiant et en validant les chaînes de narrateurs qui ont transmis un hadith donné.

Contenu

Cet ensemble de données sans précédent contient plus de 24000 chercheurs et narrateurs ainsi que leurs enseignants/élèves (et d'autres métadonnées également) qui fourniront un aperçu macroscopique de comment et où les hadiths ont été préservés au début de l'islam. L'ensemble de données peut également répondre à de nombreuses autres questions sur la question de savoir si certaines écoles de bourses sont plus prolifiques dans la préservation des hadiths que d'autres. La figure suivante montre un exemple de classification d'une partie d'El hadith :

24028 unique values	Comp.(RA) 28%	NA 88%	NA 96%
	Succ. (Taba' Tabi') 15%	'Abd al-Muttalib b. ... 0%	Umm Walad 0%
	Other (14015) 58%	Other (3003) 12%	Other (1003) 4%
عمر بن الخطاب (رضي الله عنه نفيل	Generation]	Makhzumi	bint 'Amr (Jarwal) [186] , Qaraybah bint Abi Umayyah al-Sugra...
'Uthman ibn 'Affaan (رضي عثمان بن عفان الله عنه	Comp.(RA) [1st Generation]	'Affan ibn Abi al-'As / Arwa bint Kuraiz bin Rabi'a [426]	Ruqayyah bint Muhammad [102] , Umm Kulthum bint Muhammad [94] , Fakhitah bint Ghazwan [576] , Ramlah...
Ali ibn Abi Talib (علي بن أبي طالب بن عبد المطلب (رضي الله عنه	Comp.(RA) [1st Generation]	Abu Talib ibn 'Abd al-Muttalib [9992] / Fatima bint Asad [77]	Fatima bint Muhammad [63] , Khawlah bint Ja'far [565] , [566] , Asma bint 'Umays [69] , Umm Walad(s...
Talha ibn 'Ubaidullah (طلحة بن	Comp.(RA) [1st Generation]	'Ubaidullah b. 'Uthman b. 'Amr /	Hamnah bint Jahsh [72] , Khawla bint

Figure 8 : un exemple de classification d'une partie de narrateurs El hadith DB

Remarque : Ils ont conçu aussi l'ensemble de données qui contient la translation du Quran en plusieurs langues, pour qu'il soit facilement utilisable dans des applications de traitement en langage naturel dans le but de faciliter le développement d'outils d'extraction de connaissances pour ces langues. En particulier, le Coran sémantique est compatible avec le format d'échange de langage naturel et contient des informations morpho-syntaxiques explicites sur les termes utilisés. Ainsi qu'ils ont appliqué l'extraction de motifs séquentiels sur des article Wikipédia qui contient des question/réponses arabe nous allons les détailler dans le travail qui suit.

2.4 Conclusion

Dans ce chapitre on a présenté les différentes utilisations de fouille de texte dans la Langue arabe, ainsi que les travaux réalisés jusqu'à maintenant, nous avons remarqué qu'il y a trop peu de recherches dédiée à l'extraction d'information à partir de texte arabe, concrètement extraction de motifs. Dans ce qui suit nous allons présenter une description de jeu de données que nous allons utiliser ainsi que les algorithmes appliquées.

Chapitre 3

Présentation de jeu de données et d'algorithmes utilisées

3.1 Introduction

Dans la suite de ce chapitre nous allons présenter le jeu de données que nous allons utiliser, ainsi que les algorithmes appliqués sur ce dernier.

3.2 Présentation de jeu de données

Corpus coranique arabe, une ressource linguistique annotée qui montre la grammaire, la syntaxe et la morphologie arabes de chaque mot du Saint Coran. Le corpus propose trois niveaux d'analyse : l'annotation morphologique, une treebank syntaxique et une ontologie sémantique. Une treebank est une ressource linguistique qui rassemble des arbres syntaxiques. Ce sont des analyses de phrases annotées manuellement qui peuvent être lues à la fois par les humains et les ordinateurs, avec des treebanks différents adoptant différentes théories de syntaxe. La plus récente recherche informatique en langue arabe se concentre sur l'arabe standard moderne, et l'arabe classique du Coran a été relativement inexploré. Presque aucune attention n'a été accordée à la grammaire arabe traditionnelle, malgré de nombreux volumes écrits sur le sujet au cours des siècles.**[10]**

La distribution Uthmani du projet **Tanzil** est utilisée (http://tanzil.info) et n'est pas modifiée. C'est une représentation exacte de la Madina Mushaf. Le texte est stocké sous forme de document XML Unicode, avec un élément XML pour chaque chapitre et verset du Coran.

Il n'est pas facile de trouver un système de POS disponible, robuste et précis pour traiter les textes coraniques en arabe, car il s'agit d'un texte sacré et d'une langue dont la structure morphologique est complexe. Pour décrire le corpus coranique pour tous les lecteurs, le corpus coranique arabe (Ducs et Habash, 2010) est une ressource linguistique intégrée et fiable qui se compose de 77430 mots d'arabe coranique, divisé en 114 documents. Chaque mot est étiqueté avec sa partie de la parole ainsi que de multiples caractéristiques morphologiques qui sont

basées sur le traditionnel Grammaire arabe. En outre, il est stocké dans un fichier texte et est disponible gratuitement. Les données dans le corpus est écrit dans le schéma de translittération arabe de Buckwalter.**[10]**

3.2.1 Buckwalter

La translittération de Buckwalter utilise des caractères ASCII pour représenter l'orthographe arabe. Comme il y a une correspondance un-à-un avec Unicode, le schéma d'encodage est réversible. JQuranTree utilise un superset de translittération Buckwalter pour permettre une translittération réversible de Tanzil XML. JQuranTree est un ensemble d'API Java pour accéder et analyser le Coran, sous sa forme arabe authentique. La distribution Uthmani du projet Tanzil est utilisée et n'est pas modifiée.**[10]**

Translittération étendue de Buckwalter : Il y a 4 caractères non-arabiques dans la sorcière d'encodage originale ne se trouvent pas dans le texte coranique : P (peh), J (tcheh), V (veh) et G (gaf). Le caractère de combinaison alif + maddah (|) n'est pas non plus utilisé dans Tanzil XML. Ces caractères ne sont pas implémentés par l'encodeur JQuranTree Buckwalter. De même, 14 symboles coraniques ne figurent pas dans le schéma original. Dans le schéma étendu utilisé par JQuranTree, ceux-ci ont été attribués aux marques de ponctuation ASCII. Ce n'est pas ambigu, car la ponctuation moderne ne se produit pas dans le Coran :

- Maddah (^)
- Hamza Above (#)
- Small High Seen (:)
- Small High Rounded Z ero (@)
- Small HighU pright Rectangular Zero (")
- Small High Meem Isolated Form ([)
- Small Low Seen (;)
- Small Waw (,)
- Small Ya (.)
- Small High Noon (!)
- Empty Centre Low Stop (-)
- Empty Centre High Stop (+)
- Rounded High Stop With Filled Centre (%)
- Small Low Meem (])

Le schéma étendu de translittération de Buckwalter est illustré dans la figure 9. ci-dessous. Les sections surlignées en jaune indiquent les parties ajoutées par rapport à l'original :**[10]**

UNICODE			BUCKWALTER	
Decimal	Hex	Glyph	ASCII	Orthography
1619	U+0653	ٓ	^	Maddah
1620	U+0654	ٔ	#	HamzaAbove
1648	U+0670	ٰ	`	AlifKhanjareeya
1649	U+0671	ٱ	{	Alif + HamzatWasl
1756	U+06DC	ۜ	:	SmallHighSeen
1759	U+06DF	۟	@	SmallHighRoundedZero
1760	U+06E0	۠	"	SmallHighUprightRectangularZero
1762	U+06E2	ۡ	[	SmallHighMeemIsolatedForm
1763	U+06E3	ۣ	;	SmallLowSeen
1765	U+06E5	ۥ	,	SmallWaw
1766	U+06E6	ۦ	.	SmallYa
1768	U+06E8	ۨ	!	SmallHighNoon
1770	U+06EA	۪	-	EmptyCentreLowStop
1771	U+06EB	۫	+	EmptyCentreHighStop
1772	U+06EC	۬	%	RoundedHighStopWithFilledCentre
1773	U+06ED	ۭ	]	SmallLowMeem

Figure 9 : Les parties ajoutées dans le schéma étendu de translittération de Buckwalter

3.2.2 Composant de jeu de donnes

Le jeu de données est organisé en quatre colonnes comme suit :

1. LOCATION : comprend quatre parties :
   - 1.1.N° chapitre : c'est-à-dire en arabe (رقم السورة)
   - 1.2.N° verset : c'est-à-dire en arabe (رقم الاية)
   - 1.3. N° mot : c'est-à-dire la position d'un mot dans un verset par exemple :
     bi=1 , somi=1 , {ll~ahi=2.
   - 1.4. N°partie : c'est-à-dire le numéro des partie d'un mot par exemple :
     le Préfixe bi = 1 et la racine somi = 2.
2. FORM : comprend les parties principales du mot .**[10]**

3. TAG : comprend l'étiquette (POS) de la partie du discours pour chaque partie du mot
tels que Nom, Verbe, Adjectif, etc,

nous avons remarqué qu'il est écrit sous la forme suivante : POS : la valeur de Part of speech et cette valeur peux être l'un de ces tag comme montre le tableau suivant : **[10]**

	Tag	Arabic Name	Description
Nouns	N	اسم	Noun
	PN	اسم علم	Proper noun
Derived nominals	ADJ	صفة	Adjective
	IMPN	اسم فعل أمر	Imperative verbal noun
Pronouns	PRON	ضمير	Personal pronoun
	DEM	اسم اشارة	Demonstrative pronoun
	REL	اسم موصول	Relative pronoun
Adverbs	T	ظرف زمان	Time adverb
	LOC	ظرف مكان	Location adverb

Figure 10: part of speech de la partie de discours pour les nominaux.

4. FEATURES: décrit les caractéristiques morphologiques du mot, notamment : Racine, LMAs, sexe,etc. Nous pouvons mentionner quelques caractéristiques : **Préfixe** : En plus de part of speech de la parole, de multiples fonctions d'inflexion sont assignées à chaque segment morphologique. Par exemple, les caractéristiques pour la personne, le sexe et le nombre. Les caractéristiques pour les préfixes se terminent en + et sont montrées dans les figures 11 à 15 ci-dessous. En revanche, les suffixes commencent par +.**[10]**

Feature	Name	Segment part-of-speech / description
Al+	determiner (*al*)	DET – determiner prefix ("the")
bi+	preposition (*bi*)	P – preposition prefix ("by", "with", "in")
ka+	preposition (*ka*)	P – preposition prefix ("like" or "thus")
ta+	preposition (*ta*)	P – particle of oath prefix used as a preposition ("by Allah")
sa+	future particle (*sa*)	P – prefixed particle indicating the future ("they will")
ya+	vocative particle (*yā*)	VOC – a vocative prefix usually translated as "O"
ha+	vocative particle (*hā*)	VOC – a vocative prefix usually translated as "Lo!"

Figure 11 : Caractéristiques identifiant les segments préfixés.

Feature	Name	Segment part-of-speech / description
A:INTG+	interrogative particle (*alif*)	INTG – prefixed interrogative particle ("is?", "did?", "do?")
A:EQ+	equalization particle (*alif*)	EQ – prefixed equalization particle ("whether")

Figure 12 : Caractéristiques identifiant la particule alif comme préfixe.

Feature	Name	Segment part-of-speech / description
w:CONJ+	conjunction (*wa*)	CONJ – conjunction prefix ("and")
w:REM+	resumption (*wa*)	REM – resumption prefix ("then" or "so")
w:CIRC+	circumstantial (*wa*)	CIRC – circumstantial prefix ("while")
w:SUP+	supplemental (*wa*)	SUP – supplemental prefix ("then" or "so")
w:P+	preposition (*wa*)	P – particle of oath prefix used as a preposition ("by the pen")
w:COM+	comitative (*wa*)	COM – comitative prefix ("with")

Figure 13 : Caractéristiques identifiant la particule wāw comme préfixe.

Feature	Name	Segment part-of-speech / description
f:REM+	resumption (*fa*)	REM – resumption prefix ("then" or "so")
f:CONJ+	conjunction (*fa*)	CONJ – conjunction prefix ("and")
f:RSLT+	result (*fa*)	RSLT – result prefix ("then")
f:SUP+	supplemental (*fa*)	SUP – supplemental prefix ("then" or "so")
f:CAUS+	cause (*fa*)	CAUS – cause prefix ("then" or "so")

Figure 14 : Caractéristiques identifiant la particule fa comme préfixe

Feature	Name	Segment part-of-speech / description
l:P+	preposition (*lām*)	P – the letter *lām* as a prefixed preposition
l:EMPH+	emphasis (*lām*)	P – the letter *lām* as a prefixed particle used to give emphasis
l:PRP+	purpose (*lām*)	P – the letter *lām* as a prefixed particle used to indicate purpose
l:IMPV+	imperative (*lām*)	P – the letter *lām* as a prefixed particle used to form an imperative

Figure 15 : Caractéristiques identifiant la particule lām comme préfixe

Racine et LEMs : En arabe et d'autres langues sémitiques comme l'hébreu, des mots semblables peuvent être regroupés selon une racine. Il s'agit d'une séquence de 3 ou 4 consonnes (appelées radicaux) qui forment ensemble une racine trilitère ou quadrilatérale. A partir d'une seule racine, une grande variété de mots peuvent être formés, avec des significations distinctes mais connexes. Par exemple à partir de la racine trilittérale kāf tā bā (ك ت ب) le verbe "écrire" peut être formé, ainsi que ses dérivés en arabe y compris "écrire", "livre", "auteur", "bibliothèque" et "bureau". Le concept de lemme est également utilisé pour regrouper des mots similaires à un niveau de granularité plus fin qu'une racine. Les lemmes regroupent des formes-mots qui ne diffèrent que par leur morphologie ineffective (par opposition à dérivée) et ne varient pas dans leur signification. Contrairement à la racine, la lemme est un mot réel sélectionné pour représenter le groupe et est généralement le même mot que celui utilisé dans les titres du dictionnaire. Une troisième caractéristique utilisée pour regrouper les mots est la fonction SP (spéciale). Certains groupes de verbes et de particules ont des règles particulières en grammaire arabe en ce qui concerne les terminaisons de cas et les rôles syntaxiques. **[10]**

Feature	Name	Description
ROOT:	root	Indicates the (usually triliteral) root of a word, for example ROOT:ktb
LEM:	lemma	Specifies the common lemma for a group of words, for example LEM:kitaAb
SP:	special	Indicates that the word belongs to a special group, for example SP:<in~

Figure 16 : caractéristiques des racines et des LEMs

Personne, sexe et nombre : En arabe, les mots peuvent infléchir pour la personne, le sexe et le nombre. Contrairement aux mots anglais inflexion non seulement pour le pluriel et le singulier, mais aussi pour le double. Par exemple, il y a un mot-forme distinct pour représenter "deux livres". Dans le corpus coranique arabe, les caractéristiques de la personne, du sexe et du nombre sont combinées à l'aide d'une notation concaténative. Par exemple, 3MS représente la troisième personne, masculine, singulière. De même, 2D représente la deuxième personne, double. Le concept de genre dans la grammaire arabe peut faire référence au genre sémantique, morphinique ou grammatical (voir la grammaire du genre). **[10]**

Feature	Arabic Name	Values	Description
person	الاسناد	1, 2, 3	first person, second person, third person
gender	الجنس	M, F	masculine, feminine
number	العدد	S, D, P	singular, dual, plural

Figure 17: caractéristiques des Personne, sexe et nombre

D'où nous pouvons conclure que ils y a des caractéristiques différentes, séparé part '|' comme montre la figure suivante :

```
LOCATION        FORM    TAG     FEATURE
(112:1:1:1)     qulo    V
        STEM|POS:V|IMPV|LEM:qaAla|ROOT:qwl|2MS
(112:1:2:1)     huwa    PRON
        STEM|POS:PRON|3MS
(112:1:3:1)     {ll~ahu PN
        STEM|POS:PN|LEM:{ll~ah|ROOT:Alh|NOM
(112:1:4:1)     >aHadN  N
        STEM|POS:N|LEM:>aHad|ROOT:AHd|M|INDEF|N
OM
```

Figure18 : un verset d'Alikhlas en schéma de translittération de Buckwalter.

3.3 Les algorithmes d'extraction de motifs utilisées

Avant de présenter les algorithmes il faut définir quelque notions :

- **Déf.1-Itemset (ensemble d'éléments) :**aussi appelé transactions est un ensemble d'éléments est appelé un itemset. Si un itemset a k-items, il est appelé un k-itemset. Un itemset se compose de deux ou plusieurs éléments. Un itemset qui se produit fréquemment est appelé un itemset fréquent. Ainsi, l'extraction fréquente d'éléments est une technique d'exploration de données qui permet d'identifier les éléments qui se produisent souvent ensemble. **Par exemple:** Pain et beurre, Ordinateur portable et logiciel antivirus, etc.**[11]**
  Un exemple d'une transaction :

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Figure 19 : Exemple de Transactions (Itemset)

- **Déf.2-Fréquence (frequency ou bien count) :** c'est le nombre d'occurrence d'un élément ou bien ensemble d'éléments dans un documents ou des transactions. Si nous calculons la fréquence de chaque élément de l'exemple dans la figure 19, nous pouvons trouver le tableau suivant :

Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

Figure 20 : Exemple de fréquences

- **Déf.3-Support :** le support pour un ensemble d'éléments à identifier comme fréquents. c'est le nombre de transactions où un item ou bien une Itemset apparaitre devisé par le nombre totale de transaction par exemple si on prendre l'exemple de la figure 19,20 donc le support de I1 est égale a : **[11]**

$$\textbf{Support(I1)} = \frac{le\ nombre\ de\ transactions\ (I1)}{le\ nombre\ totale\ de\ transactions} = \frac{4}{6} = 0.6$$

D'où 0.6 est le support de I1

3.3.1 Algorithme de Prefix Span :

Une méthode de pattern-growth basée sur la projection est utilisée dans l'algorithme PrefixSpan pour trouver les motifs séquentiel fréquents. L'idée de base derrière cette méthode est, plutôt que de projeter des bases de données de séquence en évaluant les occurrences fréquentes de sous-séquences, la projection est faite sur préfixe fréquent. Cela permet de réduire le temps de traitement qui augmente finalement l'efficacité de l'algorithme. Voici quelques définitions de base qui sont nécessaires pour comprendre l'exécution de l'algorithme PrefixSpan :[12]

a) Jeu de données de séquence : C'est un ensemble de séquences où chaque séquence a une liste d'ensembles d'éléments.

b) Ensemble d'éléments(Item Set) : C'est un ensemble non ordonné d'éléments distincts.
c) Support d'un motif séquentiel : C'est le nombre de séquences qui sont calculées en divisant le motif se produit avec le nombre total de séquences dans la base de données.
d) Motif séquentiel fréquent : Le motif séquentiel ayant un support supérieur au paramètre minsup qui sera fourni par l'utilisateur.

Jian Pei et al **[12]**. ont proposé un nouvel algorithme appelé PrefixSpan (Prefix-projected Sequential Pattern Mining) qui fonctionne sur la projection de la base de données et la croissance du motifs séquentiel. La technique de division et d'espace de recherche est mise en œuvre par PrefixSpan.
Nous pouvons résumé les étapes de fonctionnement de ce dernier par :

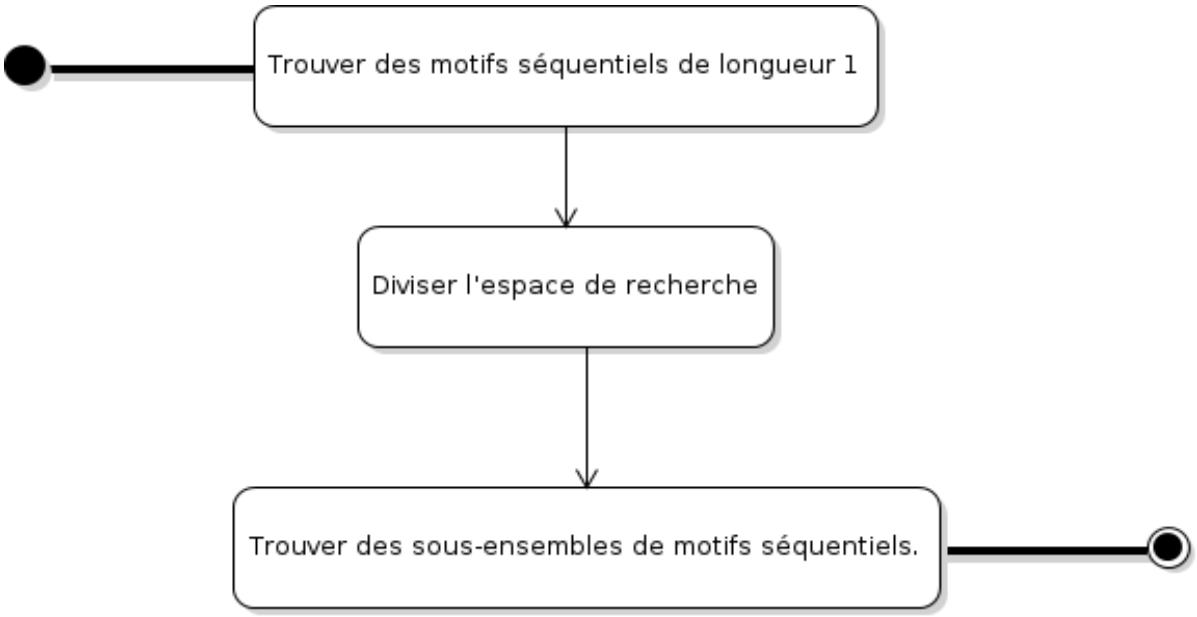

Figure 21: diagramme d'activité qui montre les étapes le fonctionnement de l'algorithme PrefixSpan

Explication de figure avec un exemple : à l'aide de la base de données séquentielle, S, du tableau suivant :

Sequence_ID	Sequence
1	$\langle a(abc)(ac)d(cf)\rangle$
2	$\langle(ad)c(bc)(ae)\rangle$
3	$\langle(ef)(ab)(df)cb\rangle$
4	$\langle eg(af)cbc\rangle$

avec un min-sup= 2 les motifs séquentiels dans S seront exploités par une méthode de projection préfixée dans les étapes suivantes : **[12]**

1. Trouver des motifs séquentiels de longueur 1. Balayer S une fois pour trouver tous les éléments fréquents dans les séquences. Chacun de ces éléments fréquents est un motif séquentiel de longueur-1. Ils sont : a : 4, b : 4, c : 4, d : 3, e : 3, et f : 3, où la notation « pattern : count » représente le motif et le nombre de supports associés.

2. Diviser l'espace de recherche. L'ensemble complet des motifs séquentiels peut être partitionné dans les six sous-ensembles suivants selon les six préfixes : (1) ceux avec préfixe a , (2) ceux avec préfixe b, ..., et (6) ceux avec préfixe f.

3. Trouver des sous-ensembles de motifs séquentiels. Les sous-ensembles de motifs séquentiels mentionnés à l'étape 2 peut être exploité en construisant des bases de données
en exploitation récursive. Les bases de données projetées, ainsi que les motifs séquentiels
sont énumérés dans le tableau suivant :

prefix	projected database	sequential patterns
$\langle a \rangle$	$\langle (abc)(ac)d(cf) \rangle$, $\langle (\_d)c(bc)(ae) \rangle$, $\langle (\_b)(df)eb \rangle$, $\langle (\_f)cbc \rangle$	$\langle a \rangle$, $\langle aa \rangle$, $\langle ab \rangle$, $\langle a(bc) \rangle$, $\langle a(bc)a \rangle$, $\langle aba \rangle$, $\langle abc \rangle$, $\langle (ab) \rangle$, $\langle (ab)c \rangle$, $\langle (ab)d \rangle$, $\langle (ab)f \rangle$, $\langle (ab)dc \rangle$, $\langle ac \rangle$, $\langle aca \rangle$, $\langle acb \rangle$, $\langle acc \rangle$, $\langle ad \rangle$, $\langle adc \rangle$, $\langle af \rangle$
$\langle b \rangle$	$\langle (\_c)(ac)d(cf) \rangle$, $\langle (\_c)(ae) \rangle$, $\langle (df)cb \rangle$, $\langle c \rangle$	$\langle b \rangle$, $\langle ba \rangle$, $\langle bc \rangle$, $\langle (bc) \rangle$, $\langle (bc)a \rangle$, $\langle bd \rangle$, $\langle bdc \rangle$, $\langle bf \rangle$
$\langle c \rangle$	$\langle (ac)d(cf) \rangle$, $\langle (bc)(ae) \rangle$, $\langle b \rangle$, $\langle bc \rangle$	$\langle c \rangle$, $\langle ca \rangle$, $\langle cb \rangle$, $\langle cc \rangle$
$\langle d \rangle$	$\langle (cf) \rangle$, $\langle c(bc)(ae) \rangle$, $\langle (\_f)cb \rangle$	$\langle d \rangle$, $\langle db \rangle$, $\langle dc \rangle$, $\langle dcb \rangle$
$\langle e \rangle$	$\langle (\_f)(ab)(df)cb \rangle$, $\langle (af)cbc \rangle$	$\langle e \rangle$, $\langle ea \rangle$, $\langle eab \rangle$, $\langle eac \rangle$, $\langle eacb \rangle$, $\langle eb \rangle$, $\langle ebc \rangle$, $\langle ec \rangle$, $\langle ecb \rangle$, $\langle ef \rangle$, $\langle efb \rangle$, $\langle efc \rangle$, $\langle efcb \rangle$.
$\langle f \rangle$	$\langle (ab)(df)cb \rangle$, $\langle cbc \rangle$	$\langle f \rangle$, $\langle fb \rangle$, $\langle fbc \rangle$, $\langle fc \rangle$, $\langle fcb \rangle$

Figure 22 : les motifs séquentiels fréquents et les base de données projetées

Tandis que le processus minier est expliqué comme suit :

A-Trouver des motifs séquentiels avec le préfixe [a]. Seules les séquences contenant [a] dois être collectées. En outre, dans une séquence contenant [a], seule la sous-séquence préfixée avec la première occurrence de [a] devrait être considéré. Par exemple, dans l'ordre [(e f)(ab)(d f)cb], Seules les séquences [(b)(d f)cb] devraient être prises en compte pour les motifs séquentiels d'exploitation minière préfixés [a]. Notez que [_b] signifie que le dernier événement dans le préfixe, qui est un (a), avec b, forment un événement. Les séquences dans S contenant (a) sont projetées par rapport à a pour former la base de données (a)-projected, qui se compose de quatre séquences de suffixes : [(abc)(ac)d(c f)],[(_d)c(bc)(ae)], [(_b)(d f)cb],
et [(_f)cbc].

En scannant une fois la base de données [a]-projetée, ses éléments localement fréquents sont a : 2, b : 4, b : 2, c : 4, d : 2, et f : 2. Ainsi, tous les motifs séquentiels de longueur-2 préfixés [a] sont trouvés, et ils sont : [aa] : 2, [ab] : 4, [(ab)] : 2, [ac] : 4, [ad] : 2 et [af] : 2.
Récursivement, tous les motifs séquentiels avec le préfixe [a] peuvent être partitionnés en six sous-ensembles : (1) ceux préfixés [aa], (2) ceux avec [ab], ..., et enfin, (6) ceux avec [af]. Ces sous-ensembles peuvent être exploités en construisant des bases de données et l'exploitation minière de façon récursive, comme suit :

I. La base de données haai-projetée se compose de deux sous-séquences non vides (suffixe) préfixé avec [aa] : {[(bc)(ac)d(c f)]}, {[(_e)]}. Parce qu'il n'y a aucun espoir de la génération de sous-séquences fréquentes à partir de cette base de données projetée met fin au traitement de la base de données [aa]-projetée.
II. La base de données habi-projected se compose de trois séquences de suffixes : [(c)(ac)d(c f)], [(c)a], et [c]. Extraction récursive de la base de données habi-projetée renvoie quatre schémas séquentiels : h(c)i, h(c)ai, hai et hci (c.-à-d. [a(bc)], [a(bc)a], [aba], et [abc].) Ils forment l'ensemble complet des modèles séquentiels préfixés avec [ab].
III. La base de données [(ab)]-projected ne contient que deux séquences : [(c)(ac) d(c f)] et [(d f)cb], ce qui conduit à trouver les modèles séquentiels suivants préfixé avec [(ab)] : [c], [d], [_f] et [dc].
IV. Les bases de données projetées peuvent être construites et exploitées de manière similaire. Les motifs séquentiels trouvés sont présentés dans le Tableau de la figure .

B- Trouvez des motifs séquentiels avec le préfixe [b], [c], [d], [e], et [f], respectivement. Ceci peut être réalisé en construisant les bases de données [b]-, [c]-, [d]-, [e]-, et [f]-projetées et de les exploiter respectivement. Les bases de données Le tableau de la figure **22** présente également les tendances observées.

4. L'ensemble des motifs séquentiels est la collection des motifs trouvés dans le récursif ci-dessus procédé minier.

Les Avantages de PrefixSpan

l'algorithme PrefixSpan est basé sur schéma de croissance séquentielle (sequence growth pattern). Avec ses avantages en termes de performance et d'efficacité, l'algorithme PrefixSpan est généralement préféré dans le domaine de l'extraction des motifs fréquents. Il présente les avantages suivants :

a) Aucune génération de candidats
b) La fréquence des articles locaux seulement dénombrable
c) La méthode de recherche Divide-and-conquer est utilisée
d) Il est supérieur à GSP ainsi qu'à FreeSpan**[12]**

3.3.2 L'Algorithme d'Apriori

L'algorithme d'Apriori a été le premier algorithme proposé pour l'exploitation fréquente d'itemset. Il a ensuite été amélioré par R Agarwal et R Srikant et devenu connu sous le nom d'Apriori. Cet algorithme utilise deux étapes « joindre » et « tailler » pour réduire l'espace de recherche. Il s'agit d'une approche itérative pour découvrir les éléments les plus fréquents. **[13]**

Les étapes d'apriori :

1. Étape de jointure : Cette étape génère (K+1) des itemset à partir de K-itemsets en joignant chaque élément avec lui-même.
2. Étape de taille : Cette étape scanne le nombre de chaque élément dans la base de données. Si l'élément candidat ne répond pas au minimum de support, il est considéré comme peu fréquent et donc supprimé. Cette étape est effectuée pour réduire la taille des éléments candidats. [13]

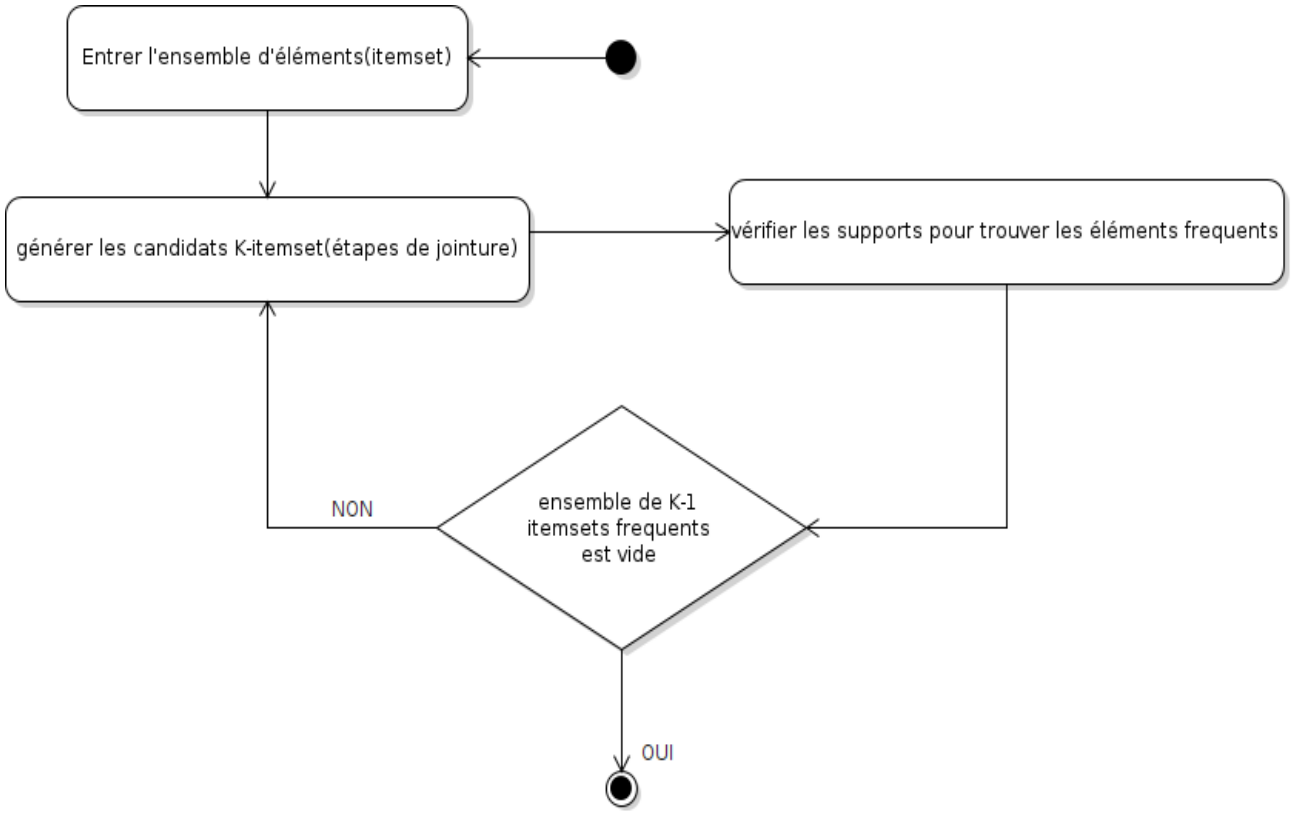

Figure 23 : diagramme d'activité qui montre les étapes le fonctionnement d'Apriori.

Explication des étapes :

1. La première étape est de scanner la base de données pour trouver les occurrences des itemsets dans la base de données
2. Seuls les candidats qui comptent plus ou égal à min_sup sont devancés pour la prochaine itération et les autres sont élagués.
3. Pour cette étapes dans l'étape de jointure, le 2-itemset est généré en formant un groupe de 2 en combinant des éléments avec lui-même.
4. Les candidats à deux éléments sont élagués en utilisant la valeur de seuil min-sup. Maintenant, le tableau aura deux éléments avec min-sup seulement.
5. La prochaine itération formera 3 – itemsets en utilisant les étapes : "join" et "prune" . Cette itération suivra la propriété anti monotone où les sous-ensembles de 3-itemsets, c'est-à-dire les sous-ensembles de 2-itemset de chaque groupe tombent dans min_sup. Si tous les sous-ensembles de deux éléments sont fréquents, alors le sous-ensemble sera fréquent, sinon il sera élagué.
6. L'étape suivante suivra la fabrication de 4-itemset en joignant 3-itemset avec lui-même et en élaguant si son sous-ensemble ne répond pas aux critères min_sup. L'algorithme est arrêté lorsque l'ensemble d'éléments le plus fréquent est atteint.[13]

Les Avantages d'Apriori

1. facile a comprendre le fonctionnement de l'algorithme.
2. Les étapes Join et Prune sont faciles à mettre en œuvre sur de grands itemsets dans de grandes bases de données.

Désavantages d'Apriori

1. Il nécessite un calcul élevé si les itemsets sont très grands et le support minimum est maintenu très bas.
2. Toute la base de données doit être scannée.

Exemple d'application d'apriori [13]

Nous continuons avec l'exemple mentionner dans la figure 19, 20 :

A-supposant que l'utilisateur a fixé le minimum support a : min_sup = 3, d'où nous remarquant que **I5** sera supprimer car elle a un support de **2,** du cout le tableau va devenir comme montre la figure suivante :

Item	Count
I1	4
I2	5
I3	4
I4	4

Figure 24 : les fréquences après la suppression de transaction qui a un support moins que min sup.

B-étape de Jointure : formant un 2-itemset de la figure trouvez l'occurrence de 2-itemset

Item	Count
I1,I2	4
I1,I3	3
I1,I4	2
I2,I3	4
I2,I4	3
I3,I4	2

Figure 25: étape de jointure (trouver l'occurrence de 2-itemset)

C-L'étape précédente montre que {I1, I4} et {I3, I4} ne respecte pas le minimum support car les fréquence de ces dernier sont égales a 2 et le min_sup=3 d'où nous allons les supprimer.

Item	Count
I1,I2	4
I1,I3	3
I2,I3	4
I2,I4	3

Figure 26 : suppression des item qui ont un support < 3

D-Étape de jointure et de prune : Formez 3-itemset. À partir du TABLEAU 1 de la figure 19, découvrez les occurrences de 3-itemset. À partir de la TABLEAU 5 de la figure 26, découvrez les sous-ensembles 2-itemset qui supportent min_sup. Nous pouvons voir que pour les sous-ensembles itemset {I1, I2, I3}, {I1, I2}, {I1, I3}, {I2, I3} se produisent dans le TABLEAU 5, de sorte que {I1, I2, I3} est fréquent.
Nous pouvons voir aussi que pour les sous-ensembles itemset {I1, I2, I4}, {I1, I2}, {I1, I4}, {I2, I4}, {I1, I4} ne sont pas fréquents, car il ne se produit pas dans le TABLEAU-5, donc {I1, I2, I4} n'est pas fréquent, donc il est supprimé.

Item
I1,I2,I3
I1,I2,I4
I1,I3,I4
I2,I3,I4

Figure 27 : étape de jointure et prune (taille) : trouver l'occurrence de 3-itemset

E-Résultat finale : D'où seulement **{I1, I2, I3} est fréquent**

3.3.3 l'algorithme FP-Growth

Fp Growth Algorithm (Frequent pattern growth). Le FPG est une amélioration de l'algorithme apriori. Il est utilisé pour trouver des itemset fréquents dans une base de données de transactions sans génération de candidats.
L'algorithme de FPG représente la base de données sous la forme d'une arbre appelé arbre de motif fréquent ou arbre FP(FP tree en anglais). Sa structure arborescente maintiendra l'association entre les itemsets. La base de données est fragmentée en utilisant un élément fréquent. Cette partie fragmentée est appelée « fragment de motif ». Les détails de ces motifs fragmentés sont analysés. Ainsi, avec cette méthode, la recherche d'itemsets fréquents est réduite comparativement. [14]

FP-Tree

Frequent Pattern Tree est une structure arborescente qui est faite avec les itemsets initiaux de la base de données. Le but de l'arbre FP est d'extraire le motif le plus fréquent. Chaque nœud de l'arbre FP représente un élément de l'ensemble d'éléments.

Le noeud racine représente null tandis que les noeuds inférieurs représentent les itemsets. L'association des nœuds avec les nœuds inférieurs qui sont les itemsets avec les autres itemsets est maintenue tout en formant l'arbre.**[14]**

Étapes de l'algorithme FPG

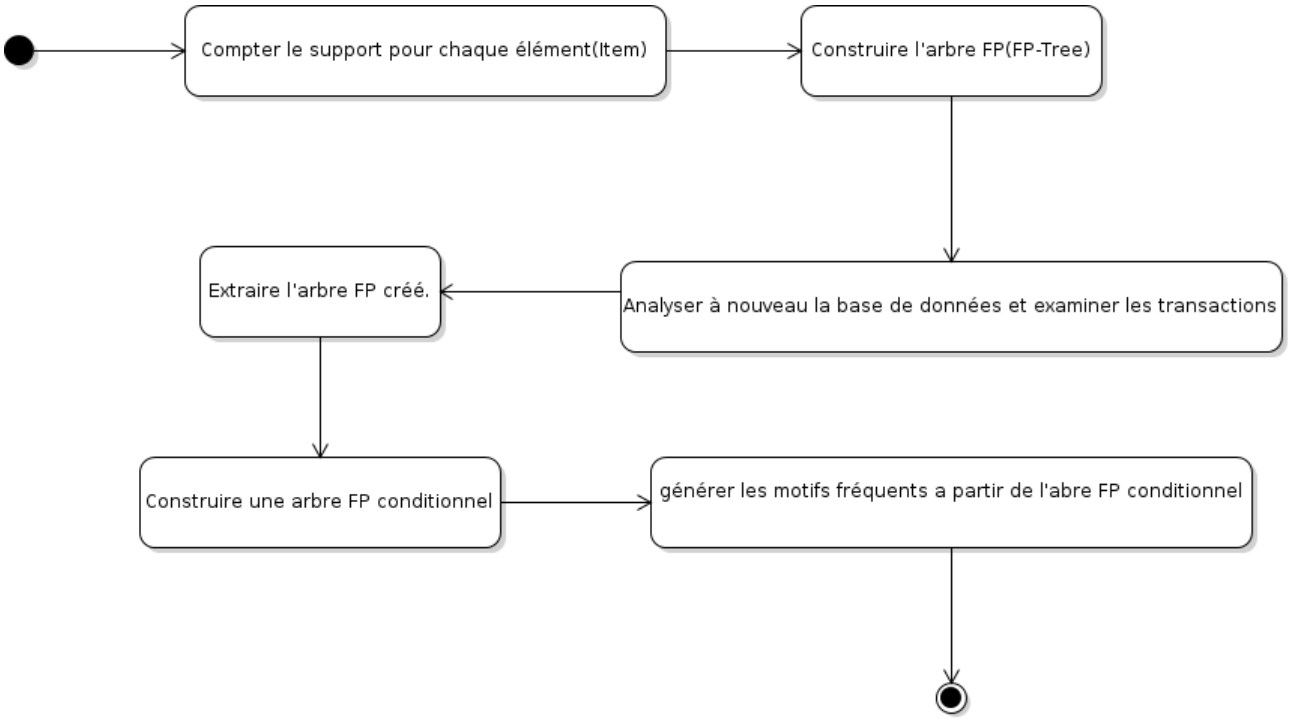

Figure 28 : diagrame d'activité qui montre les etapes le fonctionement de FP-growth

1. La première étape est de scanner la base de données pour trouver les occurrences des itemsets dans la base de données. Cette étape est la même que la première étape d'Apriori. Le nombre de 1-itemsets dans la base de données est appelé nombre de support ou fréquence de 1-itemset.
2. La deuxième étape est de construire l'arbre FP. Pour cela, créez la racine de l'arbre. La racine est représentée par null.
3. L'étape suivante consiste à analyser à nouveau la base de données et à examiner les transactions. Examinez la première transaction et découvrez les détails qui s'y trouvent. Les points avec le compte maximum sont pris en haut, les points suivants avec le compte inférieur et ainsi de suite. Cela signifie que la branche de l'arbre est construite avec des itemsets de transaction dans l'ordre décroissant de comptage.
4. La prochaine transaction dans la base de données est examinée. Les détails sont classés par ordre décroissant de nombre. Si un ensemble d'éléments de cette transaction est déjà présent dans une autre branche (par exemple dans la 1ère transaction), alors cette branche de transaction partagerait un préfixe commun à la racine.

5. De plus, le nombre d'itemset est incrémenté à mesure qu'il se produit dans les transactions. Le nombre de nœuds communs et de nouveaux nœuds est augmenté de 1 à mesure qu'ils sont créés et liés en fonction des transactions.
6. L'étape suivante consiste à extraire l'arbre FP créé. Pour cela, le noeud le plus bas est d'abord examiné avec les liens des noeuds les plus bas. Le nœud le plus bas représente la longueur du modèle de fréquence 1. À partir de ce nœud, parcourez le chemin dans l'arborescence FP. Ce chemin ou ces chemins sont appelés une base de modèle conditionnelle.
7. Construisez un arbre FP conditionnel, qui est formé par un nombre d'itemsets dans le chemin. Les éléments qui respectent le seuil de soutien sont pris en compte dans l'arborescence des PF conditionnels.**[15]**
8. Les motifs fréquents sont générés à partir de l'arbre FP conditionnel.

- **Exemple :** Si nous appliquons ces 8 étapes sur les transactions suivant :

Transaction ID	List of items in the transaction
T1	B , A , T
T2	A , C
T3	A , S
T4	B , A , C
T5	B , S
T6	A , S
T7	B , S
T8	B , A , S , T
T9	B , A , S

Figure 29 : base de données sous forme de transactions

Nous appliquons les même étapes qu'on a fait avec apriori (de la figure 20 a l'étape A),mais avec Fp-growth y'aura pas la générations de candidat, après la construction de l'arbre fp-tree et en suivant les 8 étapes mentionner avant, nous trouverons le résultat suivant :

Item	Conditional Pattern base	Conditional FP tree	Frequent Pattern Generation
Tomatoes (T)	{{A,B:1},{A,B,S:1}}	<A:2,B:2>	{A,T:2},{B,T:2},{A,B,T:2}
Corn (C)	{{A,B:1},{A:1}}	<A:2>	{A,C:2}
Squash (S)	{{A,B:2},{A:2},{B:2}}	<A:4,B:2>,<B:2>	{A,S:4},{B,S:4},{A,B,S:2}
Bean (B)	{{A:4}}	<A:4>	{A,B:4}

Figure 30 :Résultat d'application de FP-growth sur une transaction données

Avantages de FPG

1. Cet algorithme ne doit scanner la base de données que deux fois par rapport à Apriori qui scanne les transactions pour chaque itération.
2. L'appariement des éléments n'est pas fait dans cet algorithme et cela le rend plus rapide.
3. La base de données est stockée dans une version compacte en mémoire.
4. Il est efficace et évolutif pour l'exploitation à la fois des longs et courts motifs fréquents.[15]

Désavantages de FPG

1. FP Tree est plus lourd et difficile à construire qu'Apriori
2. couteuse.
3.Lorsque la base de données est grande, l'algorithme peut ne pas tenir dans la mémoire partagée.[15]

3.4 Conclusion

Dans ce chapitre on a présenté le jeu de données « quranic arabic corpus » utilisé, ainsi que les algorithmes que nous avons utilisée , nous avons remarqué qu'il y a. pas mal d'algorithmes d'extraction de motifs nous avons choisi ces 3 derniers : prefix span pour trouver les motifs séquentiels fréquents, Fp-Growth pour trouver les motifs fréquents et Apriori pour mettre une comparaison pour quoi le FPG est meilleure que la notion d'apriori .

Chapitre 4

Contributions

4.1 Introduction

Dans la suite de ce chapitre nous allons présenter le code que nous avons utilisé pour faire le prétraitement et le traitement avec les 3 algorithmes mentionner ci-dessus.

4.2 Implémentation

4.2.1 Environnement matériel et logiciel

La réalisation de notre système d'extraction de motifs fréquents, et les différentes expérimentations de nos approches, ont été faites dans l'environnement matériel et logiciel suivant :

- CPU : Intel® Core™ i5-2410M @ 2.30GHz.
- RAM : 8 GO.
- Système d'exploitation : Linux, distribution Ubuntu 18.04 LTS 64-bit.
- Outil de modélisation : Violet 2.1.
- Outils de développement : PyCharm 2021.1, Libre Office calculator.

Langages et outils de développement : Nous avons utilisé le langage de programmation Python 3, utilisant l'IDE PyCharm pour développer notre système d'extraction de motif fréquents

4.2.2 Processus de prétraitement

Nous avons utilisé le dataset Corpus coranique arabe version 0.4, qui est déjà décrit ci-dessus.

Pour réaliser notre travail, nous avions besoin les Lemmatisations de chaque mot d'où nous avons fait ce prétraitement :

Les imports :

```
import pandas as pd
```

- **Pandas :** est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de temporelles. Nous avons utilisé cette dernière pour manipuler les fichier CSV.

```
import numpy as np
```

- **NumPy** : est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

```
import csv
```

- **Le module `csv`** : implémente des classes pour lire et écrire des données tabulaires au format CSV. Il vous permet de dire « écris ces données dans le format préféré par Excel » ou « lis les données de ce fichier généré par Excel », sans connaître les détails précis du format CSV utilisé par Excel. Vous pouvez aussi décrire les formats CSV utilisés par d'autres application ou définir vos propres spécialisations.

```
import re
```

- **Le module *re* :** permet d'utiliser des expressions régulières avec Python. Les expressions régulières sont aussi appelées en anglais *regular expressions* ou en plus court *regex*

Etapes 1 : Dans cette étape nous avons utilisé le fichier texte original du dataset quranic arabic corpus pour découper la colonne FEATURE en sous colonnes en remplaçant le '|' par '\t' , en utilisant le code python suivant :

```
#*****************----------------------------------Function_1-----------------------------------------
--**************
#this function is used to splite the features columnin into multiple columns to make it easy
to take only the Lemmatization column
def Splitting_Features_column(file,output_file) :
   #the input file is the original dataset "quranic arabic corpus" Original-file.txt
   file_in = open(file, "rt")
   next(file_in)
   # create and open output file to write the result to
   file_out = open(output_file, "wt")
   # for each line in the input file(file_in)
   for line in file_in:
      # read, replace the string '|' with '\t' to split and write to output file
```

```
    file_out.write(line.replace('|', '\t'))
# close input and output files
file_in.close()
file_out.close()
#return the step 1 file that represent the original file with splited Features
return file_out
```

Le fichier original de dataset qui va être l'input de cette fonction :

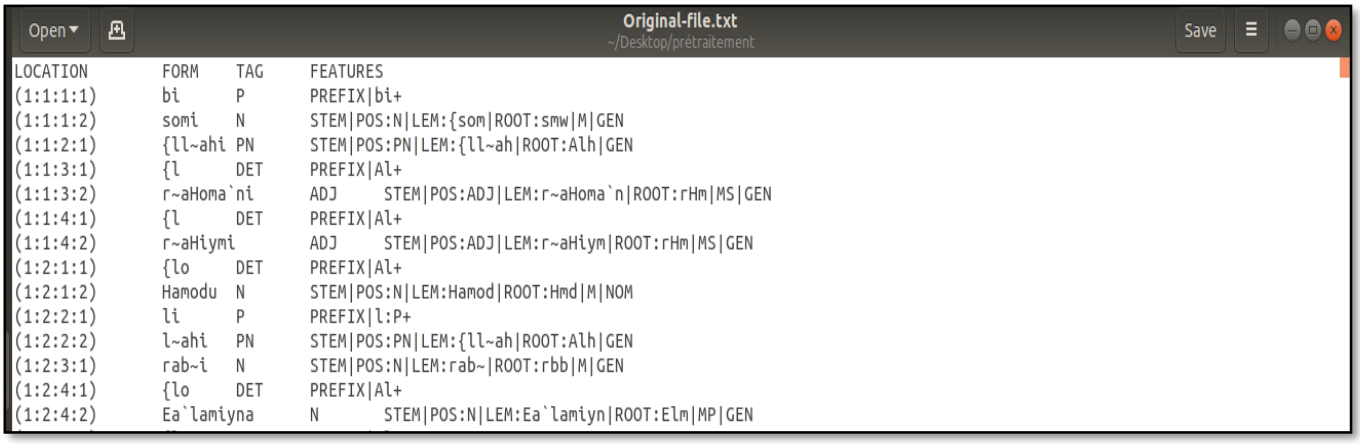

Figure 31 : Le fichier texte Original du dataset quranic arabic corpus

Résultat du code :

A	B	C	D	E	F	G	H
LOCATION	FORM	TAG	FEATURE				
(1:1:1:1)	bi	P	PREFIX	bi+			
(1:1:1:2)	somi	N	STEM	POS:N	LEM:{som	ROOT:smw	M
(1:1:2:1)	{ll~ahi	PN	STEM	POS:PN	LEM:{ll~ah	ROOT:Alh	GEN
(1:1:3:1)	{l	DET	PREFIX	Al+			
(1:1:3:2)	r~aHoma`ni	ADJ	STEM	POS:ADJ	LEM:r~aHoma`n	ROOT:rHm	MS
(1:1:4:1)	{l	DET	PREFIX	Al+			
(1:1:4:2)	r~aHiymi	ADJ	STEM	POS:ADJ	LEM:r~aHiym	ROOT:rHm	MS
(1:2:1:1)	{lo	DET	PREFIX	Al+			

Figure 32 : Résultat de l'étape 1 de prétraitement de données

Etapes 2 : nous avons utilisé le fichier csv original avec des features séparés pour renommer et remplacer les colonnes après la décomposition de la colonne FEARUES , par Faeature-1,Feature-2…..Feature-10, et nous avons choisi ce mot feature et pas des mots qui donnent un sens exact au colonne, car après la décomposition y'avais des mots par exemple qui ont 3 features et d'autre qui ont 4, d'où l'ordre change ici et les colonnes peuvent avoir des valeurs déférentes, par exemple de la colonne TAG et en même temps d'un autre de la colonne LEMs.

```
#-----------------------------------------------------Function_2-----------------------------------------------------
#this function is used to rename the splitted column of features from ---> (Feature) to---
->( Feature-1.....Feature-10)
def Renaming_columns(file,output_file) :
   #the input file is a the original file with splited Feature columns
   #reading the features splited  file  that is the result of "spliting_featiure_column" function
and in the same time renaming the columns
   df = pd.read_csv(file, encoding='utf-8', delimiter='\t',names=['LOCATION', 'FORM', 'TAG',
'Feature-1', 'Feature-2', 'Feature-3', 'Feature-4', 'Feature-5','Feature-6', 'Feature-7', 'Feature-
8', 'Feature-9', 'Feature-10'])
   #saving my dataframe as a csv file
   df.to_csv(output_file,index=False)
   return output_file
```

Résultat du code:

LOCATION	FORM	TAG	Feature-1	Feature-2	Feature-3	Feature-4	Feature-5
(1:1:1:1)	bi	P	PREFIX	bi+			
(1:1:1:2)	somi	N	STEM	POS:N	LEM:{som	ROOT:smw	M
(1:1:2:1)	{ll~ahi	PN	STEM	POS:PN	LEM:{ll~ah	ROOT:Alh	GEN
(1:1:3:1)	{l	DET	PREFIX	Al+			
(1:1:3:2)	r~aHoma`ni	ADJ	STEM	POS:ADJ	LEM:r~aHoma`n	ROOT:rHm	MS
(1:1:4:1)	{l	DET	PREFIX	Al+			
(1:1:4:2)	r~aHiymi	ADJ	STEM	POS:ADJ	LEM:r~aHiym	ROOT:rHm	MS

Figure 33 : Résultat de l'étape 2 de prétraitement de données

Etape 3: nous avons ici laissé que les ligne qui contient les LEMs, en supprimant les prefix,sufix...,en utilisant le fichier csv qui contient les features séparé avec un header renommer.

```
#*****************----------------------------------Function_3-------------------------------------
#this function keep only the rows that contain 'LEM:' because we want use only the
lemmatization of the words so here we take only the LEMAs and  delete all prefix suffix....
def keep_only_LEM_rows(file,output_f):
   #the source file is the original file after spliting feature column and renaming the header
   SOURCE_FILE =file
   result = []
   with open(SOURCE_FILE) as read_obj:
      csv_reader = csv.reader(read_obj)
      #searching for the cells that contain the expressions contain "LEM:"
      for row in csv_reader:
         for element in row:
            if 'LEM:' in element:
               #append the element to result table
               result.append(row)
   #change result to a pandas dataframe
   df = pd.DataFrame(result)
   #saving my dataframe as a csv file
   df.to_csv(output_f,index=False)
   return output_f
```

Résultat du code :

A	B	C	D	E	F	G	H
(1:1:1:2)	somi	N	STEM	POS:N	LEM:{som	ROOT:smw	M
(1:1:2:1)	{ll~ahi	PN	STEM	POS:PN	LEM:{ll~ah	ROOT:Alh	GEN
(1:1:3:2)	r~aHoma`ni	ADJ	STEM	POS:ADJ	LEM:r~aHoma`n	ROOT:rHm	MS
(1:1:4:2)	r~aHiymi	ADJ	STEM	POS:ADJ	LEM:r~aHiym	ROOT:rHm	MS

Figure 34 : Résultat de l'étape 3 de prétraitement de données

Etape 4: la fonction keep_only_LEM_elements(file) va extraire toutes les cellules qui contiennent LEM:'valeur', ou le paramètre « file » est un fichier qui continent que des lignes qui ont LEM :'valeur' dans l'un de ses cellules , et splitting_LEMAS_column va utiliser le résultat de la fonction dernière c'est-à-dire un fichier csv qui contient qu'une colonne avec LEM :'valeur' comme valeur de lignes, et prendre que la valeur le LEMs c'est-à-dire tous ce qui est après les 2 points. Le code python suivant est utilisé pour faire ça :

```
#****************----------------------------------Function_4-------------------------------------
#this function keep only the elements from each row that contains 'LEM:' and put them in a
csv file
def keep_only_LEM_elements(file,output_file):
   #this source file take the file which contain only the rows that have LEM:value in it that's
mean without prefix sufix...
   SOURCE_FILE = file
   result = []
   with open(SOURCE_FILE) as read_obj:
      csv_reader = csv.reader(read_obj)
      for row in csv_reader:
         for el in row:
            if 'LEM:' in el:
               result.append(el)
   df = pd.DataFrame(result)
   df=df.to_csv(output_file, index=False)
   return output_file
#****************----------------------------------Function_5-------------------------------------
#this function take the result of keep_only_LEM_rows that's mean as an input and as a
result take only the words that appear after LEM:
def Splitting_LEMAS_column(file,output_file):
   #the input file is the file that have only LEM elements in it thats mean a file having one
column containing LEM.value as rows
   with open(file, 'r') as f:
      my_csv_text = f.read()
   find_str = 'LEM:'
   replace_str = ''
   # substitute
   new_csv_str = re.sub(find_str, replace_str, my_csv_text)
   # open new file and save
   with open(output_file, 'w') as f:
      f.write(new_csv_str)
   return output_file
```

le resultat de keep_only_LEm_elements

LEM:{ll~ah	
LEM:r~aHoma`n	
LEM:r~aHiym	
LEM:Hamod	
LEM:{ll~ah	
LEM:rab~	
LEM:Ea`lamiyn	
LEM:r~aHoma`n	
LEM:r~aHiym	
LEM:ma`lik	
LEM:yawom	
LEM:diyn	

le resultat de splitting_LEMAs_clumn

{ll~ah	
r~aHoma`n	
r~aHiym	
Hamod	
{ll~ah	
rab~	
Ea`lamiyn	
r~aHoma`n	
r~aHiym	
ma`lik	
yawom	
diyn	

Figure 35 : Résultat de l'étape 4 de prétraitement de données.

Etape 5 : maintenant nous allons extraire le numéro de chapitre et de verset de la colonne LOCATION en utilisant le fichier csv qui contient que les lignes avec LEM :'valeur' dans l'un de cellules de ce dernier. L'objectif de cette fonction est que nous pouvons utiliser le résultat par la suite dans la fonction combine-file-location-with-LEMAs() pour concaténer les LEMs avec location.

```
#---------------------------------Function_7-----------------------------------------------------------------
#this function take the file that contains only the rows containing LEM:value in the cells of
that later to extract the location column and take only the chapter and verse than put them
in a new file LOCATION.csv
def extract_location(file,output_file):
    f = open(file, 'r', encoding='utf-8')
    result = []
    next(f, None)
    for line in f:
        re.search('(\:.+?\:)', line)
        l = re.split('[(:)]', line)
        if any(line.strip()):
            result.append(l[1:3])
    print(result)
    df = pd.DataFrame(result)
    df.rename(columns={0: 'N-chapter', 1: 'N-verse'}, inplace=True)
    df.to_csv(output_file,index=False)
    return (df)
```

Résultat du code :

	A	B	C
1	N-chapter	N-verse	
2	1	1	
3	1	1	
4	1	1	
5	1	1	
6	1	2	
7	1	2	
8	1	2	
9	1	2	
10	1	3	
11	1	3	

Figure 36 : Résultat de l'étape 5 de prétraitement de données

Etapes 6: nous avons ici nous allons concaténer les LEMs avec Location(N-chapitre, Nverset), et pour faire ça nous allons utiliser le fichier csv qui contient que les lignes avec LEM :'valeur' dans l'un de cellules de ces lignes, et pour faire ca il faut appeler les fonctions qui extraire location, les fonction qui aide a extraire que les LEMs mais en utilisant un seule fichier comme input.

```
#*****************----------------------------------Function_8-------------------------------------
#this function wil combile LOCATION file with LEMAs file that's mean using the file with rows having a LEM:value in cells than calling the functions that extract location and LEM values inside that function def combine..
def combine_Files_Location_with_LEMAs(file1,output_file,f_out_Location,f_out_LEM,f_out_Split_Lem):
    # reading the csv files Step-4.csv
    LOCATION=extract_location(file1,f_out_Location)
    LEMAs=keep_only_LEM_elements(file1,f_out_LEM)
    LEM=Splitting_LEMAS_column(LEMAs,f_out_Split_Lem)
    #converting to pandas dataframe
    data1 = pd.DataFrame(LOCATION)
    data2 = pd.read_csv(LEM)
    #concatinate the 2 dataframes data1 and data2
```

```
concatenated = pd.concat([data1, data2], axis=1)
print(concatenated)
#save to a dataframe
dataF = pd.DataFrame(concatenated)
dataF.rename(columns={'0': 'LEMAs'}, inplace=True)
dataF.to_csv(output_file,index=False)
```

Résultat du code:

	A	B	C	D
1	N-chapter	N-verse	LEMAs	
2	1	1	{som	
3	1	1	{ll~ah	
4	1	1	r~aHoma`n	
5	1	1	r~aHiym	
6	1	2	Hamod	
7	1	2	{ll~ah	
8	1	2	rab~	
9	1	2	Ea`lamiyn	
10	1	3	r~aHoma`n	

Figure 37 : Résultat de l'étape 6 de prétraitement de données

Etape 7 : ici on va supprimer les Stop words par exemple :min, man, maA…et pour faire ça nous avons besoin de connaitre la fréquence de chaque mot ensuit traduire le résultat de buckwalter vers l'arabe et la comparer avec celle traduit en arabe et les mettre dans un fichier StopWords.txt comme une liste séparé par des '|'.
Concernant la traduction nous avons utilisé buckwalter2unicode.py - Un script pour convertir l'arabe translittéré (en utilisant le système Buckwalter) vers Unicode (utf-8 ou 16).**[16]** **buckwalter2unicode** est programmé en Python.. Ce programme a été écrit et testé sur Python 3.
J'ai utilisé la ligne du code suivante dans le terminal pour traduire de buchwalter vers l'arabe

- sudo python buckwalter2unicode.py -i bucwalter_text.txt -o arabic_utf8_text.txt

Nous allons utiliser le code suivant dans cette étape :

```
def count_words_frequency(fname,output_file):
    from collections import Counter
    with open(fname) as f:
        next(f)
        freqs=Counter(f.read().split())
    print(freqs)
    df = pd.DataFrame.from_dict(freqs, orient="index")
    df1=df.rename(columns={' ': 'word', 0: 'frequency'})
    df1.to_csv(output_file)
```

```
    return df1
#****************----------------------------------Function_9-------------------------------------
#this function help us to remove the stop words like min, maA,... and save the result in a csv file
#in this function we take the file that having N-chapter,Nverse,Lemmatization as columns and remove the stop words that exist in the file like :min,man,maA.....and these  words are saved before as a list in a text file
def Removing_stop_words(file_in,SW_Ref,output_file):
  #opening the file that contain the list of stop words 'StopWord.txt'
  with open(SW_Ref) as file:
    list_of_stop_word = set(file.readlines()[0].split(','))
    data = pd.read_csv(file_in)
    df2 = data[~data['LEMAs'].isin(list_of_stop_word)]
    df2.to_csv(output_file,index=False)
```

Résultat du code :

word	frequency
min	3226
{ll~ah	2699
maA	2565
laA	1738
fiY	1701
<in~	1682
qaAla	1618
{l~a*iY	1464
EalaY`	1445
kaAna	1358
rab~	975
man	871
<ilaY`	742
<in	697
<il~aA	663
>an	625

	word	frequency
1	word	frequency
2	مِن	3226
3	ٱللَّه	2699
4	مَا	2565
5	لَا	1738
6	فِى	1701
7	إِنَّ	1682
8	قَالَ	1618
9	ٱلَّذِى	1464
10	عَلَىٰ	1445
11	كَانَ	1358
12	رَبَّ	975
13	مَن	871
14	إِلَىٰ	742
15	إِن	697
16	إِلَّا	663
17	أَن	625

Figure 38: Résultat de l'étape 7 de prétraitement de données

Open ▾ StopWords.txt ~/Desktop/prétraitement Save

```
min|man|maA|fiY|EalaY`|Kul~|{l~a*iY|laA|<in|<in~|KaAma|<ilaY`|<il~aA|>an|*a`lik|Ean|<i*aA|qad|>an~|kul~|lam|vum~|ha`*aA|>aw|bayon|qabol|<i*|>uwla`^ik|law|baEod|Eind|maE|lam~aA|>ay~uhaA|gayor|duwn|Hat~aY`|>am|bal|lan|*uw|hal|l~ayosa|lawolaA^|la`kin|>aY~|la`kin~|>am~aA|<i*on|kal~aA|<i*FA|ka>an~|<im~aA|*aA|>ayon|kaAna|kul~amaA|>an~aY`|
```

Figure 39 : Le fichier StopWords.txt

Étape 8: dans cette étape nous avons mettre chaque verset dans un ligne en utilisant le fichier csv qui contient dans chaque ligne le numéro de chapitre, numéro de verset et les LEMs sans les Stops Words.

```
#****************----------------------------------Function_10----------------------------------
----**************
#converting the file that having N-chapter,Nverse,Lemma  without Stop words  to a file
where each verse is in onerow
def Convert_to_verses(file,output_file):
    data = pd.read_csv(file)
    # Create DataFrame
    df = pd.DataFrame(data)
    df2 = df.groupby(['N-chapter', 'N-verse'])['LEMAs'].apply('|'.join).reset_index()
    df = pd.DataFrame(df2)
    df.to_csv(output_file,index=False)
```

Résultat du code :

A	B	
N-chapter	N-verse	LEMAs
1	1	{som|{ll~ah|r~aHoma`n|r~aHiym
1	2	Hamod|{ll~ah|rab~|Ea`lamiyn
1	3	r~aHoma`n|r~aHiym
1	4	ma`lik|yawom|diyn
1	5	<iy~aA|Eabada|<iy~aA|{sotaEiynu
1	6	hadaY|Sira`T|m~usotaqiym
1	7	Sira`T|>anoEama|gayor|magoDuwb|DaA^l~
2	2	kita`b|rayob|hudFY|mut~aqiyn
2	3	'aAmana|gayob|>aqaAma|Salaw`p|razaqa|>anfaqa
2	4	'aAmana|>anzala|>anzala|A^xir|yuwqinu
2	5	>uwla`^}ik|hudFY|rab~|>uwla`^}ik|mufoliHuwn

Figure 40 : Résultat de l'étape 8 de prétraitement de données

Étape 9: ensuit nous avons extraire que la liste des versets en utilisant le fichier csv qui contient les ligne ou chaque verset dans une ligne avec le numéro de chapitre et numéro verset:

```
#****************----------------------------------Function_11----------------------------------
----**************
#here we take only the liste of verses from the file that having each verse in one row
def take_only_verses_list(file,output_file):
    data1 = pd.read_csv(file)
    # Create DataFrame
    df = pd.DataFrame(data1)
    data2=df['LEMAs']
    df1 = pd.DataFrame(data2)
    df1.to_csv(output_file,index=False)
```

Résultat du code :

LEMAs
{som|{ll~ah|r~aHoma`n|r~aHiym
Hamod|{ll~ah|rab~|Ea`lamiyn
r~aHoma`n|r~aHiym
ma`lik|yawom|diyn
<iy~aA|Eabada|<iy~aA|{sotaEiynu
hadaY|Sira`T|m~usotaqiym

Figure 41 : Résultat de l'étape 9 de prétraitement de données.

Étape 10: la dernière étape est de séparer les versets qui sont coller par des "|" en utilisant le fichier csv qui contient que les verset ou chaque verset dans une ligne, et ça sera l'étape qui va nous donner le fichier finale prétraité :

```
# ****************----------------------------------Function_12----------------------------------
#this is the final step where we splite the file that contain the converted verses in one row
to multiple columns where each word is in a column
def spliting_verses_column(file,output_file) :
    # input file
    file_in = open(file, "rt")
    next(file_in)
    # output file to write the result to
    res=[]
    # for each line in the input file
    for line in file_in:
        line = line.strip()
        # read replace the string and write to output file
        line.join('/n')
        res.append((line.split('|')))
    # close input and output files
```

```
print(res)
with open(output_file, 'w') as f:
    # using csv.writer method from CSV package
    write = csv.writer(f)
    write.writerows(res)
file_in.close()
return output_file
```

Résultat du code :

	A	B	C	D	E	F	G
1	{som	{ll~ah	r~aHoma`n	r~aHiym			
2	Hamod	{ll~ah	rab~	Ea`lamiyn			
3	r~aHoma`n	r~aHiym					
4	ma`lik	yawom	diyn				
5	<iy~aA	Eabada	<iy~aA	{sotaEiynu			
6	hadaY	Sira`T	m~usotaqiym				
7	Sira`T	>anoEama	magoDuwb	DaA^l~			
8	kita`b	rayob	hudFY	mut~aqiyn			
9	'aAmana	gayob	>aqaAma	Salaw`p	razaqa	>anfaqa	
10	'aAmana	>anzala	>anzala	A^xir	yuwqinu		
11	>uwla`^}ik	hudFY	rab~	>uwla`^}ik	mufoliHuwn		
12	kafara	sawaA^'	>an*ara	>an*ara	'aAmana		
13	xatama	{ll~ah	qalob	samoE	baSar	gi$a`wap	Ea*aAb

Figure 42 : Résultat de l'étape 10 de prétraitement de données

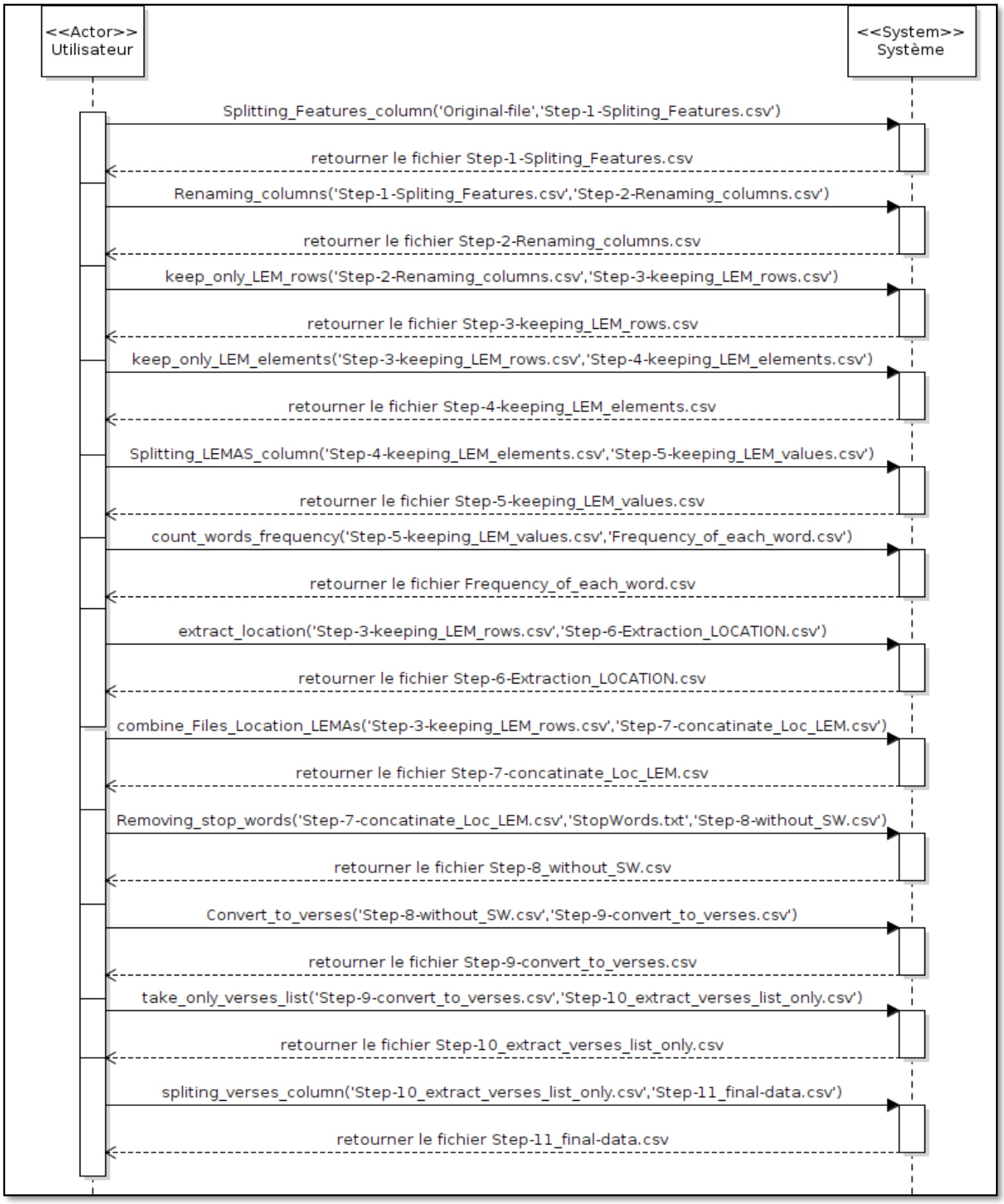

Figure 43 : diagramme de séquence d'enchaînement de prétraitement de données

Conclusion :

Nous avons maintenant terminer le prétraitement de jeu de données, par le diagramme de séquence qui explique l'enchainement de prétraitement , et par la suite nous allons traiter nos données en utilisant les 3 algorithmes d'extraction de motifs fréquents.

4.2.3 Traitement de données:

Dans cette étape nous allons tester notre jeu de données prétraitée sur les 3 algorithmes que nous avons présenté avant pour extraire les motifs fréquent à partir du quran.

les imports :

```
from prefixspan import PrefixSpan
```

- **prefixspan** c'est une bibliothèque pour implémenter l'algorithme prefix Span sous python

```
import pyfpgrowth
```

- **pyfpgrowth** c'est une bibliothèque pour implémenter l'algorithme fp-growth sous python

```
import pandas as pd
import numpy as np
import csv
```

- nous avons déjà expliquer ces 3 import dans la partie de prétraitement.

Algorithme de Prefix Span

Prefix span comme nous avons décrit avant, est une algorithme pour extraire les motifs séquentielle fréquents, le code suivant montre l'application de ce dernier sur notre dataset : ils prend comme paramètre :

- ✓ file_in :c'est le fichier csv prétraité de notre jeu de données.
- ✓ Min_support : c'est le minimum support dans cet algorithme le min support c'est la fréquence minimal d'itemset par exemple 30.
- ✓ Min_pattern_length : c'est le minimum length de motifs que nous voulons avoir par exemple si vous voulez voir que les motifs qui sont supérieur a 3 l'algorithme va afficher que les motifs de longueur >3.
  - Exemple : ['A','A','B']
- ✓ Max_pattern_length : c'est le maximun length de motifs que nous voulons avoir par exemple si vous voulez voir que

les motifs qui sont supérieur a 10 l'algorithme va afficher que les motifs de longueur <10.

- ✓ File out c'est le lien de fichier csv ou nous allons sauvegarder le résultat

Si on veut par exemple extraire les motifs séquentiels fermé nous pouvons simplement utiliser la ligne suivante :

```
ps=ps.frequent(min_support_threshold, closed=True)
```

Si on veut par exemple extraire les motifs séquentiels libre (generator) nous pouvons simplement utiliser la ligne suivante :

```
ps = ps.frequent(min_support_threshold, generator=True)
```

Le code que nous avons utilisé pour tester prefix Span:

```
# ****************----------------------------------Function_1_PrefixSpan_Algorithm----------
defPrefixSpan_Algo(file_in,min_support_threshold,min_pattern_length,max_pattern_lengt,f
ile_out) :
   with open(file_in) as file:
      reader = csv.reader(file)
      my_list = list(reader)
   print(my_list)
   ps = PrefixSpan(my_list)
   #set the minimum lenght of patterns
   ps.minlen = min_pattern_length
   #set the max length of a pattern
   ps.maxlen =max_pattern_length
   #set the threshold of min-pattern frequency to 50 or whatever i want
   ps=ps.frequent(min_support_threshold)
   #if you incomment the next line you will get the closed patterns
   #ps=ps.frequent(min_support_threshold, closed=True)
   #if you incomment the next line you will get  the generator patterns
   #ps = ps.frequent(min_support_threshold, generator=True)
   #both closed and generators
   #ps = ps.frequent(min_support_threshold,closed=True,generator=True)
   print(ps)
   df = pd.DataFrame(ps)
   #saving the result of prefixSpan in a csv file

   df.to_csv(file_out, index=False, header=True)
```

Résultat d'utilisation de ce code avec :

- min_support=30
- min_pattern_length=3
- max_pattern_length=10

Résultat du code :

32	['nafos', '{ll~ah', '{ll~ah']			
30	['>aroD', '{ll~ah', '{ll~ah']			
33	['Ealima', '{ll~ah', '{ll~ah']			
73	["samaA^'", '>aroD', '{ll~ah']			
30	["samaA^'", '>aroD', "$aYo'"]			
36	['xalaqa', "samaA^'", '>aroD']			
32	['{t~aqaY`', '{ll~ah', '{ll~ah']			
35	['jan~ap', 'jarayo', 'taHot']			
35	['jan~ap', 'jarayo', 'taHot', 'nahar']			

Figure 44 : Résultat du teste avec l'algorithme Prefix Span

- **Traitement de résultat de Prefix Span :**

Nous avons ajouté une fonction pour trouver les versets où l'algorithme à extraire les motifs séquentielles en utilisant l'expression régulière .*mot1.*mot2.*mot3.* ou : .*Mot.* : veut dire n'importe quelle mot avant ou après ce Mot.

Nous avons utiliser une petite fonction pour transformer le motifs de sa forme normal vers une expression régulière après nous l'utilisons ensuite dans une autre fonction qui va chercher cette expression dans un fichier csv qui contient les versets avec le numéro de chapitre et numéro de verset

La fonction : (Prefix_sapn_result_treatement) va prendre en argument le motif, le fichier qui contient les versets et un fichier pour sauvegarder le résultat

Le code suivant peut résumer ce que nous avons dit avant :

```
def change_to_a_RegEx(pattern):
    r=''
    #for each element in pattern add at the beginning of each item this '.*'
    for i in pattern:
        RegEx = r'.*' + i
        r=r+RegEx
    # next i added the '.*' at the end of the regular expression because in the for loop it put the '.*'
    # only in the beginning of the word so the result gonna be incomplete
    # that's why we should add '.*' at the end of RegEx to confirm that we can have a strings after the last word of my RegEx
    r = r + r'.*'
```

```
    return r
#print(change_to_a_RegEx(['jan~ap', 'jarayo', 'taHot']))
#this function gives us the verses from which the algorithm take the pattern
def Prefix_Span_result_treatement(pattern,verses,file_out):
    #this function will change the input pattern(['A','B','C','D']) to a regular expression like
this : .*A.*B.*C.*D.*
    RegEx=change_to_a_RegEx(pattern)
    print(RegEx)
    result=[]
    #the input file is a file where each verse in a row with the number of chapter and verse
    with open(verses) as read_obj:
        csv_reader = csv.reader(read_obj)
        for row in csv_reader:
            for el in row:
                if re.search(RegEx, el):
                    result.append(row)
    print(result)
    df = pd.DataFrame(result)
    df.rename(columns={0: 'N-chapter', 1: 'N-verse', 2: 'Verse'}, inplace=True)
    df.to_csv(file_out, index=False)
```

Résultat du code : si nous le testons avec le motifs : ['{ll~ah', 'rasuwl', '{ll~ah'] nous trouverons le résultat :

A	B	
N-chapter	N-verse	verse
2	98	Eaduw~|{ll~ah|malak|rasuwl|iboriyl|miykaY`l|{ll~ah|Eaduw~|ka`firuwn
3	32	qaAla|>aTaAEa|{ll~ah|rasuwl|awal~aY`|{ll~ah|>aHobabo|ka`firuwn
3	86	kayof|hadaY|{ll~ah|qawom|kafara|<iyma n|$anida|rasuwl|Haq~|jaA^'a|bay~inap|{ll~ah|hadaY|qawom|ZaAlim
3	101	kayof|kafara|talaY`|'aAyap|{ll~ah|rasuwl|{EotaSamu|{ll~ah|hadaY|Sira`T|m~usotaqiym

Figure 45 : les verset ou l'algorithme à trouver le motifs : ['{ll~ah', 'rasuwl', '{ll~ah']

Algorithme de FP-growth

FP-growth comme nous avons décrit avant, est une algorithme pour extraire les motifs fréquents, le code suivant montre l'application de ce dernier sur notre dataset : ils prend comme paramètre :

- ✓ file_in :c'est le fichier csv prétraité de notre jeu de données.
- ✓ Min_support : c'est le minimum support dans cet algorithme le min support c'est la fréquence minimal d'itemset par exemple 30.
- ✓ Min_pattern_length : c'est le minimum length de motifs que nous voulons avoir par exemple si vous voulez voir que les

motifs qui sont supérieur a 3 l'algorithme va afficher que les motifs de longueur >4. Par exemple : ['D','E','F','G']

- ✓ Max_pattern_length : c'est le maximun length de motifs que nous voulons avoir par exemple si vous voulez voir que les motifs qui sont supérieur a 10 l'algorithme va afficher que les motifs de longueur <10.
- ✓ File out c'est le lien de fichier csv ou nous allons sauvegarder le résultat

```
-----------------------------------------------------------------------------------------------------------
def FP_growth(file_in,support,N_transaction,min_pattern_length,max_pattern_length,file_out):
    import time
    # starting time
    start = time.time()
    #imports :
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import fpgrowth
    #openning the file that contains the final data pretreated and change it's type to a list than to pandas dataframe
before put it in our algorithm
    with open(file_in) as file:
        reader = csv.reader(file)
        my_list = list(reader)
    te = TransactionEncoder()
    te_ary = te.fit(my_list).transform(my_list)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    #use fpgrowth to find frequent patterns(itemsets)
    Frequent_itemsets = fpgrowth(df, min_support=(support/N_transaction), use_colnames=True)
    #change the values of the result to a list because the result is of pandas.core.dataframe type
    FP = Frequent_itemsets.values.tolist()
    print(type(FP))
    #change the list to a dictionary
    dictionary = dict()
    for v, k in FP:
        dictionary[k] = v
    result = {}  # create a new empty dictionary to put the result in
    for i in dictionary:
        #taking from the dictionary all pattern that satisfy the conditions bellow and put theme in a new dictionary
        if (len(i) >= min_pattern_length and len(i) <= max_pattern_length):
            result[i] = dictionary[i]
            result[i] = int(result[i] * N_transaction)
    # saving the result of Apriori in a csv file
    print(result)
    #change my dictionary 'result' to a pandas dataframe
    df = pd.DataFrame.from_dict(result, orient="index")
    #renaming columns
    df1 = df.rename(columns={'': 'pattern', 0: 'frequency'})
    #save the result to a csv file
    df1.to_csv(file_out)
    # end time   and total time taken
    end = time.time()
    print(f"Runtime of FP-growth algo program is {end - start}")
```

Si nous testons ce code avec avec :

- ❖ min_support=20.
- ❖ min_pattern_length=3.
- ❖ max_pattern_length=30.

Résultat de code :

A	B	C	D	E
word	frequency			
('>aHoyaA', '>amaAta', '{ll~ah')	27			
('>asolama', 'qaAla', '{ll~ah')	32			
('EiysaY', 'qaAla', '{ll~ah')	44			
('EiysaY', '{ll~ah', '{ll~ah')	23			
('rajul', '{ll~ah', '{ll~ah')	38			
('qaAla', 'qaAla', 'wa`Hid')	20			
('qaAla', 'qaAla', 'wa`Hid', '{ll~ah')	50			
('<ila`h', 'qaAla', 'wa`Hid')	31			
('<ila`h', 'qaAla', 'wa`Hid', '{ll~ah')	33			
('qaAla', 'wa`Hid', '{ll~ah')	43			
('qaAla', 'wa`Hid', '{ll~ah', '{ll~ah')	42			
('wa`Hid', '{ll~ah', '{ll~ah')	23			
('labiva', 'qaAla', 'yawom')	54			
('labiva', 'qaAla', 'qaAla')	41			
('>aqaAma', 'A^taY', 'Salaw`p', 'zakaw`p')	22			
('>aqaAma', 'A^taY', 'Salaw`p', 'zakaw`p', '{ll~ah')	33			

Figure 46 : Résultat de traitement avec l'algorithme de FP-growth.

Algorithme d'Apriori

Apriori comme nous avons décrit avant, est une algorithme pour extraire les motifs fréquents, le code suivant montre l'application de ce dernier sur notre dataset : ils prend comme paramètre :

- ✓ file_in :c'est le fichier csv prétraité de notre jeu de données.
- ✓ support : c'est le minimum support dans cet algorithme le min support c'est la fréquence minimal d'itemset par exemple 10.
- ✓ N-transaction : c'est le nombre totale de transaction,dans notre cas c'est : 6216,c'est-à-dire le nombre de versets.
  Min_pattern_length : c'est le minimum length de motifs que nous voulons avoir par exemple si vous voulez voir que les motifs qui sont supérieur a 3 l'algorithme va afficher que les motifs de longueur >4.

- Exemple : [‘D’,’E’,’F’,’G’]

✓ Max_pattern_length : c’est le maximun length de motifs que nous voulons avoir par exemple si vous voulez voir que les motifs qui sont supérieur a 10 l’algorithme va afficher que les motifs de longueur <10.

✓ File out c’est le lien de fichier csv ou nous allons sauvegarder le résultat

```
#this function change a Nested dictionary EX {1:{'A':1,'B',3},2:{('A','B'):5}... to one
dictionary{'A':1,'B':3,('A','B'):5}...
y{'A':1,'B':3,('A','B'):5}...
def Nested_dict_to_one_dict(dic):
    re = {}
    for k, v in dic.items():
        if isinstance(v, dict):
            re.update(v)
        else:
            re.update({k: v})
    return re
#Apriori Algoorithm
#the input file is transactional dataset and N_transation is the total number of transactions that we have and
support is the frequency of itemset
def apriori(file,support,N_transaction,min_pattern_length,max_pattern_length,file_out):
    with open(file) as file:
        reader = csv.reader(file)
        my_list = list(reader)
    from efficient_apriori import apriori
    itemsets, rules = apriori(my_list, min_support=(support/N_transaction), min_confidence=0)
    T = Nested_dict_to_one_dict(itemsets)  # this function change the result from nested dictinary to one normal
dictionary
    print(T)
    result = {}  # create a new empty dictionary to put the result in
    for i in T:
        if (len(i) >= min_pattern_length and len(i) <= max_pattern_length):
            result[i] = T[i]  # copy element by elemnet from patterns dictionary to result dictionary
    # saving the result of Apriori in a csv file
    df = pd.DataFrame.from_dict(result, orient="index")
    df1 = df.rename(columns={'': 'pattern', 0: 'frequency'})
    df1.to_csv(file_out
```

Si nous teston ce code avec

- ❖ support=50
- ❖ N_transaction =6216.
- ❖ min_pattern_length=3.
- ❖ max_pattern_length=20
- ❖ Le min_support dans l’algorithme sera calculé par 50/6216=0.008

- ❖ Le paramètre min_confidence (probabilité) : est un paramètre utilisé pour calculer et trouver les règles d'association mais notre but n'est pas d'extraire les règles d'où nous avons mis confidence=0.

('"aAmana", 'qaAla', '{ll~ah')	67					
('>aroD', "samaA^"', '{ll~ah')	131					
('"aAmana", 'Eamila', 'S~a`liHa`t')	51					
('Ealima', 'qaAla', '{ll~ah')	59					
('qaAla', 'rab~', '{ll~ah')	86					
('>aroD', 'qaAla', '{ll~ah')	53					
('qaAla', 'qawom', '{ll~ah')	50					
('gafuwr', 'r~aHiym', '{ll~ah')	57					
('"aAmana", 'rasuwl', '{ll~ah')	53					
('qaAla', 'rasuwl', '{ll~ah')	51					

Figure 47 : Résultat de traitement de notre jeu de données avec l'algorithme d'apriori

4.4. Conclusion :

Dans ce chapitre nous avons présenté le code python que nous avons utilisé pour faire le prétraitement et le traitement de données.et par la suite nous allons discuter les résultats que nous avons obtenu en testant avec déférents support, min_length, maxlength.

Chapitre 5

Discussion des résultats

5.1 introduction

Dans ce chapitre nous allons parler et faire un bilan sur ce qu'on a trouvé comme résultat, en testant les algorithmes avec des support , min-length, max_length déférents, pour voir le fonctionnement

5.2 Résultat l'algorithme de Prefix Span :

Min support threshold	Min pattern length	Max pattern length	Result
10	3	30	1003 motif
10	3	6	1002 motif
10	4	7	244 motif
20	4	10	23 motif
20	3	30	180 motif
20	3	4	179 motif
30	3	10	73 motif
30	4	10	9 motif
40	3	10	32 motif
50	3	10	15 motif
50	3	5	15 motif
50	2	10	116 motif
60	3	10	11 motif
60	2	10	91 motif
75	3	10	6 motif

Tableau 1 : tester le prefix span avec des valeurs d'arguments déférents

- Nous avons testé l'algorithmes d'extraction de motifs fréquents séquentielle prefix Span sur notre dataset prétraitée., en observant les résultat, nous pouvons conclure que à chaque fois le support augmente les motifs diminuera.
- Concernant la longueur minimal de motifs, nous pouvons dit que 3 c'est mieux que 2, et 1 car un motifs de longueur de 2 et 1 ne sera pas vraiment utile à utiliser par les expert de domaine.

- Ici nous testons avec les option closed et generator :

Min Support	Min patternlength	Option	Result
20	3	Closed	163 pattern
20	3	Generator	142 pattern

Tableau 2 : tester le prefix span avec les options closed et generator

5.3. Résultat d'algorithme de FP-growth et d'apriori

Min Support	Min pattern length	Max pattern length	Result
10	4	20	184 motif
10	5	20	51 motif
10	7	20	1 motif
20	4	30	17 motif
30	2	20	331 motif
30	3	10	49 motif
40	3	20	19 motif
50	3	10	10 motif
50	2	20	120 motif
60	2	10	13 motif

Tableau 3 : tester le FP-growth avec des valeurs d'arguments déférents

- Nous avons testé l'algorithmes d'extraction de motifs fréquents sur notre dataset prétraitée, en observant les résultat, nous pouvons conclure que à chaque fois le support augmente les motifs diminuera et les motifs que nous avons trouvé sont intéressantes avec un temps d'exécution très court
- Concernant la longueur minimal de motifs, nous pouvons dire aussi que 3 c'est mieux que 2, et 1 car un motifs de longueur de 2 et 1 ne sera pas vraiment utile à utiliser par les expert de domaine, ainsi nous avons tester avec les même paramètre l'algorithme d'apriori en comparant le temps d'exécution avec fp-growth sur tout

quand nous diminuons le support ,apriori prend beaucoup de temps d'où le FPG est plus rapide, et les résultats que nous avons obtenu sont intéressantes.

5.3 Conclusion :

Nous avons testé les algorithmes d'extraction de motifs fréquents sur notre dataset prétraitée. En observant les résultat, Nous avons remarqué que la suppression des Stop Words nous aide à trouver de bons résultats, car la fréquence de chaque SW est très élevé. Ainsi cela va donner des mauvaises résultats si nous ne les supprimons pas avant notre traitement.

Le performance de prefix Span est très rapide et efficace, ainsi que fp-growth, mais si nous comparons la vitesse de traitement et la complexité d'Apriori avec le Fp-growth nous pouvons conclure que le FPG est plus rapide qu'Apriori avec des bases de données pas très importantes en terme de volume.

Conclusion Générale

Nous avons abordé de nombreux sujets dans ce rapport. Tout d'abord, nous avons présenté un état de l'art général sur les techniques d'IA liée au fouille de texte, en expliquant le processus de text mining, plus particulièrement l'extraction d'information à partir du texte en se basant sur l'extraction de motifs.

Extraire de motifs à partir du texte c'est une étape très importante en TM, elle permet de trouver des patterns qui donnent la facilité aux experts à réaliser leur travail de recherche (classification, clustering…etc.), par exemple extraire des motifs grammaticaux dans une phrase, comme un sujet suivi d'un verbe suivi d'un complément. Cela s'applique à de nombreux domaines, par exemple la découverte de motifs, utilisés comme patron linguistique, montrant la relation entre gène et maladie.

Par conséquent, lors de la phase étude bibliographique, nous avons passé en revue des méthodes et algorithmes d'extraction de motifs séquentiels utilisés dans diverses langues (anglais, français…), pour montrer c'est quoi l'extraction de motifs et pour quoi on l'utilise. Par la suite nous, avons parlé du traitement de la langue arabe, à la fois plus difficile et très riche en forme et grammaticalement. Comme le montre le deuxième chapitre, très peu de travaux de recherche ont été menés sur l'extraction de motifs dans le texte arabe. Cela peut s'expliquer par les règles morphologiques uniques de cette langue. L'orthographe à côté des diacritiques tend à être plus phonétique et très clair en arabe.

Dans le cadre de ce travail nous avons réalisé un système générique d'extraction de motifs fréquents séquentielles et non séquentielles en utilisant les 3 algorithmes présenter dans le chapitre 3, ainsi que nous avons appliqué ces algorithmes sur un dataset arabe qui est intitulé par « quranic arabic corpus », contenant le sacré coran, aves des caractéristiques morphologiques de chaque mot composant ce dernier. Nous avons fait un prétraitement sur notre dataset pour que nous l'utilisons après dans la phase de traitement, en se basant sur l'extraction de la lemmatisation de chaque mot du coran, après avoir formé un fichier qui contient chaque verset dans une ligne en supprimant avant les SW, ainsi que prefix,sufix.

Nous avons réalisé ce système d'extraction de motifs fréquents générique, afin de rendre l'amélioration de ce système facile. Même si à travers nos expérimentations nous avons étudié plusieurs exemples d'extraction de motifs en utilisant les 3 algorithmes,

Bibliographie

[1] **Riedl MO**. Human-centered artificial intelligence and machine learning. Hum Behav & Emerg Tech. 2019;1:33–36.

[2] **Bastien L** https://www.lebigdata.fr/traitement-naturel-du-langage-nlp-definition.

[3] **Alexandra Twin**. https://www.investopedia.com/terms/d/datamining.asp.

[4] **Shivam Arora** l'article web : https://www.simplilearn.com/data-mining-vs machine-learning.

[5] **Miloš Radovanović, Mirjana Ivanović** "TEXT MINING:APPROACHES AND APPLICATIONS" Novi Sad J. Math.Vol. 38, No. 3, 227-234 (2008).

[6] Le site web : https://monkeylearn.com/text-mining/.

[7] **Pierre Holat.** Fouille de motifs et modélisation statistique pour l'extraction de connaissances textuelles. Modélisation et simulation. Université Sorbonne Paris 2018.

[8] **Noureddine Doumi.** Extraction d'information à partir d'un texte arabe. Intelligence artificielle [cs.AI]. Université Djillali Liabes de Sidi Bel Abbès, 2017

[9] **Said A. Salloum, Ahmad Qasim Alhamad, Mostafa Al-Emran, Khaled Shaalan,** A Survey of Arabic Text Mining.

[10] Dataset utilisé quranic Arabic corpus : https://corpus.quran.com/.

[11] **Prof. Fazal Rehman Shamil** Support, Confidence, Minimum support, Frequent itemset, K-itemset, absolute support in data mining

[12] **Jian Pei, Member, IEEE Computer Society, Jiawei Han, Senior Member, IEEE,Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Member, IEEE Computer Society, and Mei-Chun Hsu** Mining sequential Patterns by Pattern-Growth:The PrefixSpan Approach

[13] Article dans le site web : https://www.softwaretestinghelp.com/apriori-algorithm/.

[14] **Q. Ruan, Y. Li, and X. Liu,** "A hash table and linear based improved FP-Tree algorithm," Journal of Yangtze University (Natural Science Edition): Science & Engineering, vol. 1, pp. 76 79, 2010.

[15] Article web : https://www.softwaretestinghelp.com/fp-growth-algorithm-data-mining/

[16] **andyroberts** buckwalter2unicode.py - A script to convert transliterated Arabic (using the Buckwalter system) to Unicode.

[17] International Journal of Innovative Research in Computer Science & Technology(IJIRCST)ISSN: 2347-5552,Volume-3, Issue-3, May-2015.

Download violet UML : https://sourceforge.net/projects/violet/.
Download PyCharm IDE : https://www.jetbrains.com/pycharm/.