République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Saad Dahleb Blida -1-

Faculté des Sciences

Département d'Informatique

MEMOIRE DE FIN D'ETUDES EN VUE DE

L'OBTENTION DU DIPLÔME DE MASTER EN

INFORMATIQUE

Option: Traitement Automatique de Langue (TAL)

Extraction de motifs basée sur word2vec

Réalisé par :

- **Mr. Slimane Yacine**
- **Mr. Tahar djoudi Salim**

Promotrice :

- **M^me^. Zahra Fatma Zohra**

Devant le jury composé de :

- **M^me^ ARKAM**
- **M^me^ Charfa**

Blida le : 15/12/2020

Remerciement

Nous tenons à remercier.

En premier lieu le bon Dieu de nous avoir donné la force et le courage pour réaliser à terme ce travail.

Nous remercierons notre promotrice Mme Zahra Fatma Zohra pour nous avoir Proposé ce thème, nous lui sont très reconnaissants pour ses remarques et conseils.

En fin nos remerciements s'adressent aussi aux membres du jury pour nous avoir fait l'honneur d'examiner notre travail.

Résumé

L’extraction de motifs est une technique des techniques importante dans le domaine de la fouille de données. Elle a de nombreuses applications telles que l’analyse du comportement des consommateurs, bio-informatique, sécurité, etc.

Il existe de nombreux algorithmes permettant l’extraction des motifs. Ces algorithmes présentent plusieurs problèmes tels que la pertinence des motifs extraits. En effet, dans ce travail, nous utilisant la technique de word2vec pour l'extraction de motifs à partir de données réelles.Afin d’évaluer de trouver les motifs nous avons utilisé les mesures de similarité cosinus et la distance euclidienne. Le premier algorithme se base sur skip gram, le deuxième se base sur le principe de négative sampling et le troisième utilise une fonction de probabilité jointe afin de trouver les motifs de toutes les tailles.

Mots-clés : word2vec, extraction des motifs, motifs fréquents, skip gram, négative sampling, wordembeddings.

Abstract

Pattern extraction is an important technique in the field of data mining. It has many applications such as analysis of consumer behavior, bioinformatics, security, etc.

There are many algorithms for extracting patterns. These algorithms present several problems such as the relevance of the extracted patterns. Indeed, in this work, we are using the technique of word2vec for the extraction of patterns from real data. In order to evaluate to find the patterns we used the cosine similarity measures and the Euclidean distance. The first algorithm is based on skip gram, the second is based on the principle of negative sampling and the third uses a joint probability function to find patterns of all sizes.

Keywords: word2vec, pattern extraction, frequent patterns, skip gram, negative sampling, word embeddings.

ملخص

يعد استخراج الأنماط تقنية مهمة في مجال التنقيب عن البيانات. لها العديد من التطبيقات مثل تحليل سلوك المستهلك ، والمعلوماتية الحيوية ، والأمن ، إلخ.

هناك العديد من الخوارزميات لاستخراج الأنماط. تقدم هذه الخوارزميات العديد من المشاكل مثل أهمية الأنماط المستخرجة. في الواقع ، في هذا العمل ، نستخدم تقنية word2vec لاستخراج الأنماط من البيانات الحقيقية. للتقييم لإيجاد الأنماط ، استخدمنا مقاييس تشابه جيب التمام والمسافة الإقليدية. تعتمد الخوارزمية الأولى على تخطي الجرام ، والثانية تعتمد على مبدأ أخذ العينات السلبية ، والثالثة تستخدم دالة احتمالية مشتركة للعثور على أنماط من جميع الأحجام.

الكلمات المفتاحية:word2vec ، استخلاص الأنماط ، الأنماط المتكررة ، تخطي الجرام ، أخذ العينات السلبية ، تضمين الكلمة.

Table de matières

Introduction générale 1

1.1 Introduction 4

1.2 Intelligence Artificielle 4

1.3 Machine Learning 4

1.3.1 Apprentissage supervisé 5

1.3.2 Apprentissage non supervisé 6

1.3.3 Apprentissage par renforcement 6

1.4 Deeplearning 6

1.4.1 Pour quoi le choix deep Learning 7

1.5 Reseau de noronnes 7
1.6 Les architectures du deeplearning 9
1.6.1 Réseaux à une couche (feed-forward) 9
1.6.2 Réseaux multi-couches (feed-forward) 10
1.6.3 Réseaux récurrents (feed-back) 10
1.7 Les architectures deDeep Learning 11
1.7.1 Perceptron multi-couches (MLP) 11
1.7.2 Réseaux de neurones profonds 12
1.7.3 Réseau de neurones récurrents 12
1.7.4 Les réseaux de neurones convolutifs 12
1.8 Exemples d'application de Deep Learning 13
1.9 Conclusion 13
2 Chapitre2 : Word Embeddings 14
2.1 Introduction 14
2.2 Traitement Automatique du Langage Naturel 14
2.3 La relation entre Word Embeddings et TALN 14
2.4 Les Word embeddings 15
2.5 Les méthodes de Word embeddings 16
2.5.1 GloVe(Global Vectors for Word Representation) 16
2.5.2 Objectif de GloVe 16
2.5.3 Word2vec 17
2.5.4 Architectures de Word2Vec 19
2.5.5 Les différences entre GloVe et word2vec 23
2.5.6 Utilisation des « Word Embeddings » 24
2.6 Conclusion 25
3 Chapitre3 : Extractions des motifs fréquents 26
3.1 Introduction 26
3.2 Extractions des motifs 26
3.2.1 Concepts de base du processus d'extraction des motifs fréquents 26
3.2.2 Algorithme d'extraction des motifs 28
3.2.3 Discussion 37
3.3 Extractions des motifs séquentiels 38
3.3.1 Motifs séquentiels 38

3.3.2 Concepts généraux 39
3.3.3 ALGORITHMES D'EXTRACTION DE MOTIFS SEQUENTIELS 40
3.4 Conclusion 48
4 Chapitre 4 : Solution proposée 49
4.1 Introduction 49
4.2 Algorithme de item2vec_1 49
4.2.1 Pseudo code 50
4.3 Algorithme Item2vec_2 56
4.4 Algorithme item2vec_3 59
4.4.1 Une fonction d'apprentissage des vecteurs d'éléments d'incorporation 59
4.5 Conclusion 60
5 Chapitre 5 : Tests et évaluations 61
5.1 Introduction 61
5.2 Environment de development 61
5.2.1 Environment materiel 61
5.2.2 Environment logiciel 62
5.3 Implémentation 63
5.3.1 Algorithme item2vec_1 63
5.3.2 Algorithme item2vec_2 65
5.3.3 Projection 2d 67
5.4 Expérimentation 67
5.4.1 Description de la base de donnée (ToyData.txt) 67
5.4.2 Evaluation des résultats 69
5.4.3 Temps d'exécutions 71
5.4.4 Consommation de la mémoire 72
5.4.5 Comparaison entre les résultats obtenus 73
5.5 Conclusion 74
Conclusion générale et perspectives 75
Bibliographie 76

Liste des figures

Figure 1.1 - Intelligence artificielle, machine learning et deep learning 7
Figure 1.2 – Représentations d'un réseau de neurones artificiels (ANN) sous la forme d'un
L'information se propage en suivant ces liens entre les entrées et les sorties. [08] 8
Figure 1.3 – Structure interne d'un nœud de calcul dans le cas d'un neurone artificiel sommateur.
..... 9
Figure 1.4 – Réseaux à une couche [04]. 9
Figure 1.5 – Réseaux Multi-couche [04]. 10
Figure 1.6 – Réseaux récurrents [04]. 10

Figure 1.7 – Réseau de neurones dense, acyclique et structuré en couches communément appelé perceptron multi-couches (MLP). [08] 11
Figure 2.1 – Exemples des architectures Skip-Gram et CBOW de Word2Vec. [28] 18
Figure 2.2 – Architecture du modèle CBOW. [29] 20
Figure 2.3 – Exemples de l'architecture Skip-Gram de Word2Vec. [29] ..... **Erreur ! Signet non défini.**
Figure 3.1 : Algorithme Extraction des itemsets avec Apriori.[34] 29
Figure 3.2 : Algorithme extraction des itemsets avec Eclat. [38] 31
Figure 3.3 : arbre de 1ère transaction. 35
Figure 3.4:arbre de 2eme transaction. 35
Figure 3.5 : arbre de 3eme transaction. 36
Figure 3.6 : arbre de 4eme transaction. 36
Figure 3.7 :arbre de 5eme transaction.[41] 36
Figure 3.8 : Jointure de séquences dans GSP 41
Figure 3.9 : Algorithme GSP.[50] 42
Figure 3.10 : classes d'équivalences générer par l'algorithme SPADE.[49] 46
Figure 3.11 - Le pseudo code de l'algorithme SPADE.[51] 47
Figure 4.1 : Architecture de item2vec_1 50
Figure 4.2 : calcule le hidden et output layer et la fonction softmax. 52
Figure 4.3 : calcule erreur. 54
Figure 4.4 : calcule delta pour w1 et w2. 55
Figure 4.5 : Exemple de distribution de probabilité 56
Figure 4.6 La différence d'entrainement dans les deux solutions proposes.[55] 57
Figure 4.7 : Exemple de distribution positifs et distribution negatifs. 58
Figure 4.8 : Exemple de distribution positifs et distribution negatifs. 58
Figure 5.1 : la fonction du prétraitement. 64
Figure 5.2 : la fonction d'entrainement. 64
Figure 5.3 : fonction pour lire la base de donnée 65
Figure 5.4 : fonction pour générer les vocabs. 66
Figure 5.5 : La fonction d'enregistrer l'entrainement. 66
Figure 5.6 : Projection 2d du modèle extrait en utilisant item2vec_1. 69

Figure 5.7 : Projection 2d du modèle extrait en utilisant item2vec_2. 69

Liste des tableaux

Tableau 3.1: Base detickets de caisse.[32] 26
Tableau 3.2:représentation canonique sous forme de codes binaires. [32] 28
Tableau 3.3 : Exemple de base de transactions. [41]..... 32
Tableau 3.4 : Items associés à leur support [41]..... 33
Tableau 3.5 : les items qui ont un support supérieur ou égal à 3..... 34
Tableau 3.6 : les itemsets de type ordonnée. 34

Tableau 3.7: Fouille du FP-Tree. [41] 37
Tableau 3.8: Fouille du FP-Tree. [42] 37
Tableau 3.9-Exemple base de données de séquences. [45] 39
Tableau 3.10 : représentation des données horizontalement. [52] 43
Tableau 3.11 - Calculer les candidats. [52] 44
Tableau 5.1 -La base de données ToyData. 68
Tableau 5.2 : Similarités cosine et la distance réalisée par item2vec_1, En bas à gauche la distance. En haut à droite, les similarités. 70
Tableau 5.3 : Similarités cosine et la distance réalisée par item2vec_2, En bas à gauche la distance. En haut à droite, les similarités. 70
Tableau 5.4 - Temps de réponse des algorithmes. 71
Tableau 5.5 - Consommation de la mémoire des algorithmes 72
Tableau 5.6 - Les motifs extrais par les trois algorithmes. 73
Tableau 5.7 - Le pourcentage d'être les résultats justes 73

Introduction générale

Les algorithmes d'apprentissage automatique fonctionnant sur des données structurées nécessitent des représentations de données souvent symboliques sous forme des vecteurs numériques. Ces représentations vectorielles de données sont, soit calculées par des algorithmes d'embedding dédiés ou implicitement apprises par des architectures d'apprentissage comme les réseaux de neurones. Les performances des méthodes d'apprentissage automatique dépendent de manière cruciale de la qualité des représentations vectorielles.

La communauté de recherche en traitement du langage naturel (TLN) a une longue tradition en utilisation de bag-of-words à vecteur unique, où sa dimension est égale à la taille du vocabulaire. Récemment, il y a eu une explosion de l'utilisation de wordembeddings comme entrée dans les algorithmes d'apprentissage automatique, en particulier dans la communauté de l'apprentissage profond. Les mots incorporés sont des vecteurs de nombres réels qui sont des représentations distribuées de mots.

Une technique d'entraînement populaire pour wordembeddings, word2vec [1], consiste à utiliser un réseau de neurones à deux couches qui est formé sur le mot courant et les mots de contexte qui l'entourent. Cette reconstruction du contexte des mots est vaguement inspirée du concept linguistique de qui stipule que les mots qui apparaissent dans le même contexte ont une signification similaire [2].

Les algorithmes d'apprentissage profond appliqués au wordembeddings ont connu des améliorations spectaculaires dans plusieurs domaines tels que, la traduction automatique, l'analyse des sentiments, les résumés automatiques …etc. Récemment, une multitude de recherches proposent un large éventail des méthodes d'incorporation de vecteurs pour diverses applications.

Problématique

L'extraction de motifs est un domaine essentiel de Data Mining qui consiste à découvrir des motifs intéressants, inattendus et utiles dans un ensemble de données. Traditionnellement, les chercheurs de la communauté de l'extraction de motifs se préoccupaient de l'amélioration des performances des algorithmes proposés.

A cause de développement rapide des applications dans les domaines de la santé, de la bio-informatique et du commerce électronique, des masses de données très importantes sont de plus en plus collectées. Par conséquent, plusieurs défis sont apparus dans le domaine de l'extraction de motifs :

- Le problème de l'efficience des algorithmes d'extraction de motifs est encore posé.
- L'imperfection des données collectées.
- Le problème de la mauvaise qualité des motifs extraits.
- L'efficacité des algorithmes proposés est une autre difficulté à traiter.
- Le temps de réponse est très élevé.

Objectif

Plusieurs travaux ont été proposés dans la littérature qu'ils se sont inspirés de wordembeddings, plus précisément la technique word2vec pour proposer des représentions vectorielles aux différentes structures de données tels que : node2vec [3], graph2vec [4], doc2vec [5], dna2vec [6].

Effet, le but de ce travail s'intègre dans ce contexte. Il s'agit de proposer des méthodes qui permettent d'extraire et d'analyser des motifs simples. Ces méthodes se basent sur les représentations vectorielles générées en s'inspirant de la technique de word2vec.

En premier lieu,on va se focaliser sur la faisabilité et l'utilité de l'utilisation du principe de wordembeddings pour l'extraction des itemsets (motifs simples). Cette méthode est censée améliorer beaucoup plus la qualité des motifs extraits dans un premier lieu, grâce à la représentation contextuelle et sémantique des items.

Organisation du mémoire

A part l'introduction le mémoire se reparti en cinq chapitres :

Chapitre 1 : « Apprentissage profond »

Ce chapitre est consacré à la présentation d'apprentissage profond.

Chapitre 2 : « Word Embeddings »

Le deuxième chapitre présente le wordembeddings, ces méthodes, et sa relation avec TALN.

Chapitre 3 : « Extraction des motifs fréquents »

Ce chapitre est consacré à la présentation des différents types d'algorithme pour l'extraction des motifs fondamentaux.

Chapitre 4 : « Conception »

Ce chapitre, sera réservé pour exposer nos solutions proposées.

Chapitre 5 : « Tests et évaluations »

Le dernier chapitre concrétise et valide la méthode adoptée.

1 Chapitre1 : Apprentissage Automatique

1.1 Introduction

l**e** Machine Learning, aussi appelée apprentissage automatique en français, C'est une science moderne utilisée pour trouver des répétitions (pattern) dans un ou plusieurs flux de données et faire des prédictions basées sur des informations statistiques.

Dans ce chapitre, nous définirons brièvement l'intelligence artificielle et l'apprentissage automatique, puis nous présenterons l'apprentissage profond en détail.

1.2 Intelligence Artificielle

L'intelligence artificielle (IA) se définie : «un ensemble de techniques et d'algorithmes mises en œuvre pour produire des machines capables de simuler l'intelligence». [1]

l'IA est une spécialité scientifique relative au traitement des connaissances, Le but de cette spécialité est de permettre à une machine d'exécuter des fonctions qui sont associées à l'intelligence humaine (compréhension, raisonnement, dialogue, adaptation, apprentissage, etc.), Alors que le machine learning et le deeplearning sont des sous-domaines qui se réfèrent à des techniques et des algorithmes sur des ensembles de données, afin d'avoir des bonnes résultats grâce à l'expérience et l'apprentissage au fil du temps. [02]

1.3 Machine Learning

L'apprentissage automatique (Machine Learning) est l'un des principaux domaines dans l'intelligence artificielle qui traite des méthodes d'identification et des algorithmes par lesquels un ordinateur peut apprendre, ce domaine est associé à l'intelligence artificielle et plus spécifiquement intelligence computationnelle.

L'intelligence computationnelle est une méthode d'analyse de données qui pointe vers la création automatique de modelés analytiques. Autrement dit, permettant à un ordinateur d'élaborer des concepts, d'évaluer, prendre des décisions et prévoir les options futures. [03]

Le machine Learning nécessite deux ensembles de données :

- **Ensemble de données pour l'entraînement :** c'est la base de connaissance utilisée pour entrainer, notre l'algorithme d'apprentissage, pendant cette phase, les paramètres du modèle peuvent être réglés (ajustés) en fonction des performances obtenues.
- **Ensemble de données pour le test :** cela est utilisé juste pour évaluer les performances du modèle sur les données non-vues.

La théorie de l'apprentissage utilise des outils mathématiques dérivés de la théorie des probabilités et de la théorie de l'information, Cela vous permet d'évaluer l'optimalité de certaines méthodes par rapport aux autres. [04]

On peut citer trois types d'algorithme d'apprentissage automatique :

- Apprentissage supervisé.
- Apprentissage non supervisé.
- Apprentissage par renforcement.

1.3.1 Apprentissage supervisé

L'apprentissage supervisé est la tâche d'apprentissage automatique la plus simple et la plus connue. Il est basé sur un certain nombre d'exemples pré classifiés, dans lesquels est connu à priori la catégorie à laquelle appartient chacune des entrées utilisées comme exemples. Dans ce cas, la question cruciale est le problème de généralisation, après l'analyse d'un échantillon d'exemples, le système devrait produire un modèle qui devrait fonctionner pour toutes les entrées possibles. [04]

L'ensemble de données pour l'entrainement, est constitué de données étiquetées, c'est-à-dire d'objets et de leurs classes associées. Cet ensemble d'exemples étiquetés constitue donc l'ensemble d'apprentissage.

Afin de mieux comprendre ce concept, prenons un exemple : un utilisateur reçoit chaque jour un grand nombre d'e-mails, certains sont des e-mails d'entreprises importants et d'autres sont des e-mails indésirables non sollicités ou des spam.

Un algorithme supervisé sera présenté avec un grand nombre d'e-mails qui ont déjà été étiquetés par l'utilisateur comme spam ou non spam. L'algorithme fonctionnera sur toutes les données étiquetées, faire des prédictions sur l'e-mail et voir si c'est un spam ou non. Cela signifie

que l'algorithme examinera chaque exemple et fera une prédiction pour chacun pour savoir si l'e-mail est un spam ou pas. [05]

1.3.2 Apprentissage non supervisé

La deuxième classe d'algorithmes d'apprentissage automatique est appelée apprentissage non supervisé, dans ce cas, nous n'étiquetons pas les données au préalable, nous laissons plutôt l'algorithme arriver à sa conclusion.

Les algorithmes d'apprentissage non supervisé sont particulièrement utilisés dans les problèmes de clustering, dans lesquels, étant donné une collection d'objets, nous voulons être en mesure de comprendre et de montrer leurs relations. Une approche standard consiste à définir une mesure de similarité entre deux objets, puis à rechercher tout groupe d'objets plus similaires les uns aux autres, par rapport aux objets des autres clusters. Par exemple, dans le cas précédent des e-mails spam/ non spam, l'algorithme peut être capable de trouver des éléments communs à tous les spam (par exemple, la présence de mots mal orthographiés). Bien que cela puisse fournir une classification meilleure qu'aléatoire, il n'est pas clair que les spam/non spam puissent être facilement séparés. [06]

1.3.3 Apprentissage par renforcement

L'apprentissage par renforcement est une approche de l'IA qui met l'accent sur l'apprentissage du système à travers ses interactions avec l'environnement.

Avec l'apprentissage par renforcement, le système adapte ses paramètres en fonction des réactions reçues de l'environnement, qui fournit ensuite un retour d'information sur les décisions prises. Par exemple, un système qui modélise un joueur d'échecs qui utilise le résultat des étapes précédentes pour améliorer ses performances, est un système qui apprend avec le renforcement. La recherche actuelle sur l'apprentissage avec renforcement est hautement interdisciplinaire et comprend des chercheurs spécialisés dans les algorithmes génétiques, les réseaux de neurones, la psychologie et les techniques de contrôle. [04]

1.4 Deeplearning

Le Deep Learning est basé sur l’idée des réseaux de neurones artificiels et il est taillé pour gérer de larges quantités de données en ajoutant des couches au réseau. Un modèle de deeplearning a la capacité d’extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires

et apprendre sur ces caractéristiques petit à petit à travers chaque couche avec une intervention humaine minime. [07]

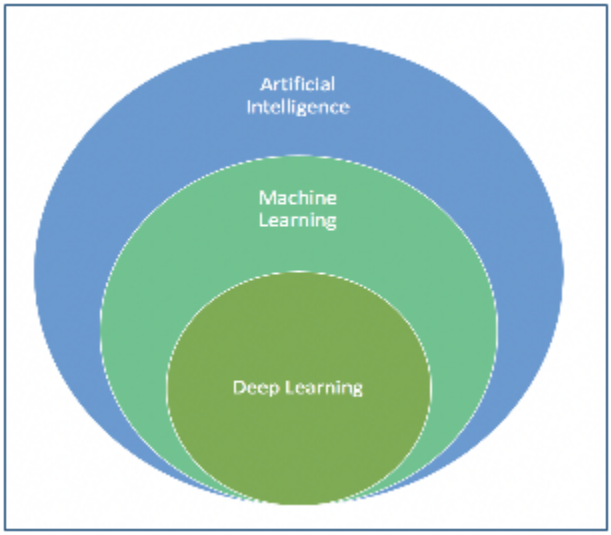


Figure 1.1- Intelligence artificielle, machine learning et deep learning

1.4.1 Pour quoi le choix deep Learning

Tout d'abord les différents algorithmes du deepLeaming ne sont apparus qu'à l'échec de l'apprentissage automatique tentant de résoudre une grande variété de problèmes de l'intelligence artificielle (l'IA) [07] :

- Afin d'améliorer le développement des algorithmes traditionnels dans de telles tâches de l'IA.
- De développer une grande quantité de données telle que les big, data.
- De s'adapter à n'importe quel type de problème.
- D'extraire les caractéristiques de façon automatique.

1.5 Reseau de noronnes

Le réseau de neurones artificiels (Artificial Neural Network - ANN) fut introduit comme un modèle rudimentaire du traitement de l'information dans le cerveau humain. Ainsi, la structure élémentaire d'un ANN est un réseau de petits nœuds de calcul reliés entre eux par des liens dirigés et pondérés (Fig. 1.2). Les nœuds représentent les neurones et les liens pondérés représentent la force des connections synaptiques reliant les neurones entre eux.

Le neurone peut alors être un sommateur des potentiels des signaux synaptiques qui lui parviennent, et qui transmet à son tour une information basée sur cette somme via une fonction de transfert de préférence non linéaire (Fig. 1.3).

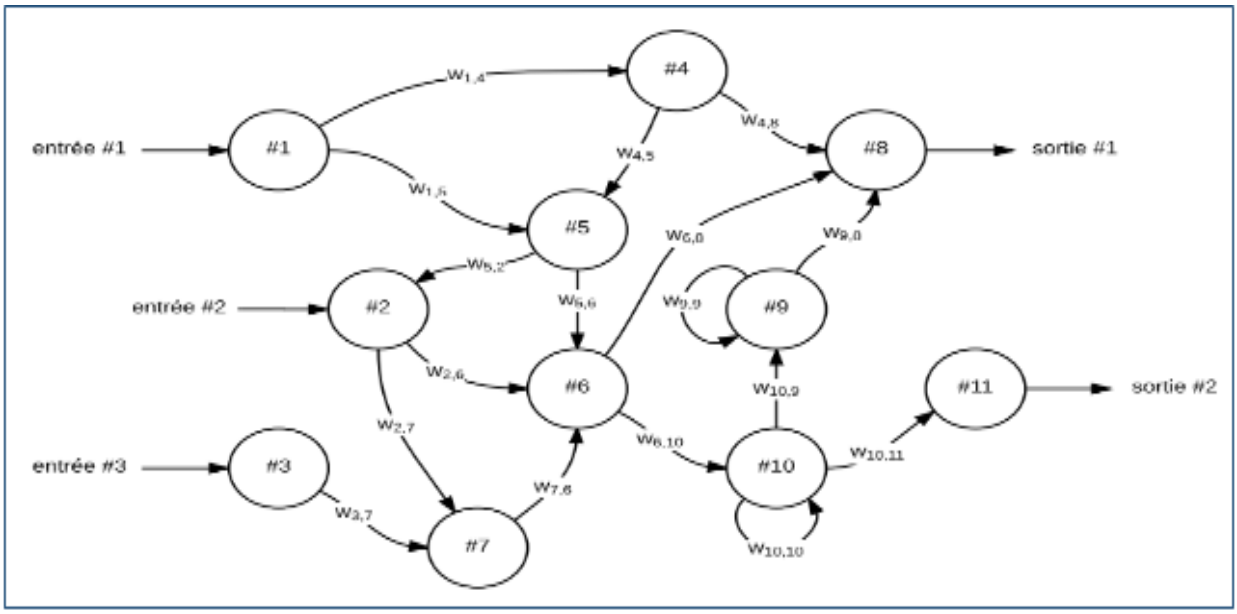


Figure 1.2–Représentations d'un réseau de neurones artificiels (ANN**),** Réseau de nœuds de calculs reliés par des liens dirigés et pondérés par les coefficients Wi,j .[08]

Un ANN est activée en injectant des données au niveau de tout ou partie des nœuds puis en propageant l'information en suivant les liens pondérés. Une fois l'information propagée, on peut collecter les niveaux d'activation de tout ou partie des nœuds et les utiliser comme commande dLa sortie est alors communiquée aux autres nœuds de calculs. [08]

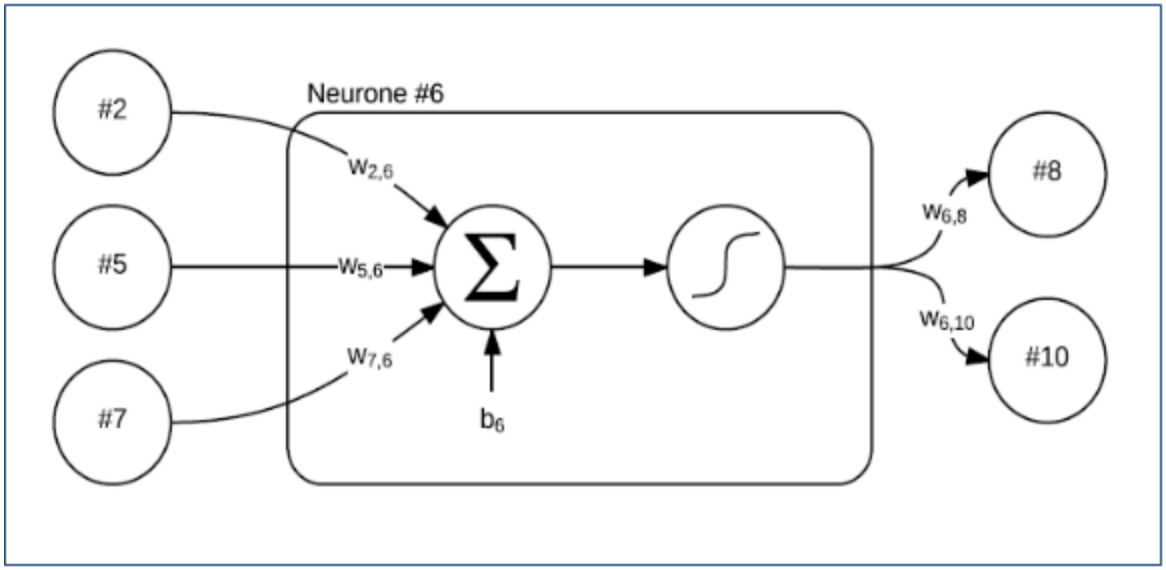


Figure 1.3– Structure interne d'un nœud de calcul dans le cas d'un neurone artificiel sommateur.[8]

1.6 Les architectures du Deeplearning

La structure d'un réseau dépend de l'algorithme d'apprentissage que vous avez l'intention d'utiliser. En général, nous pouvons identifier 3 classes de réseaux :

1.6.1 Réseaux à une couche (feed-forward)

Dans cette forme simple de réseau en couches, nous avons des noeuds d'entrée et une couche de neurones (couche de sortie). Le signal se propage dans le réseau de manière linéaire, en commençant par la couche d'entrée et en se terminant par la couche de sortie. Il n'y a pas de connexions qui reviennent et pas de connexions transversales dans la couche de sortie. [08]

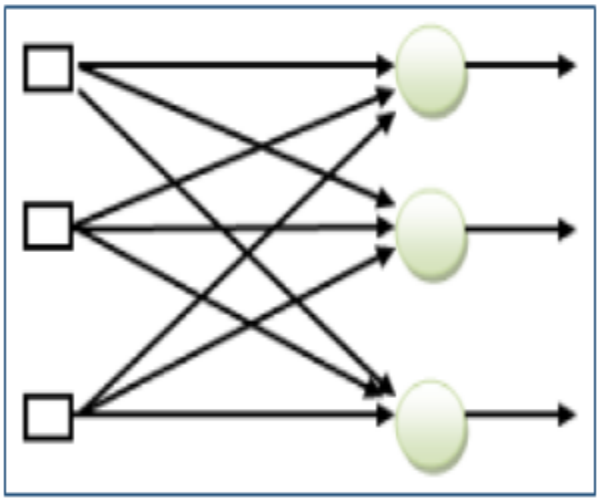


Figure 1.4 – Réseaux à une couche. [04]

1.6.2 Réseaux multi-couches (feed-forward)

Cette classe de réseaux feed-forward diffère de la précédente car elle a une ou plusieurs couches de neurones cachés (couches cachées) entre les couches d'entrée et de sortie. Chaque couche a des connexions entrantes de la couche précédente et sortantes dans la suivante, donc la propagation du signal se fait de façon linéaire sans cycles et sans connexions transversales. Ce type d'architecture fournit au réseau une perspective globale car il augmente les interactions entre les neurones.

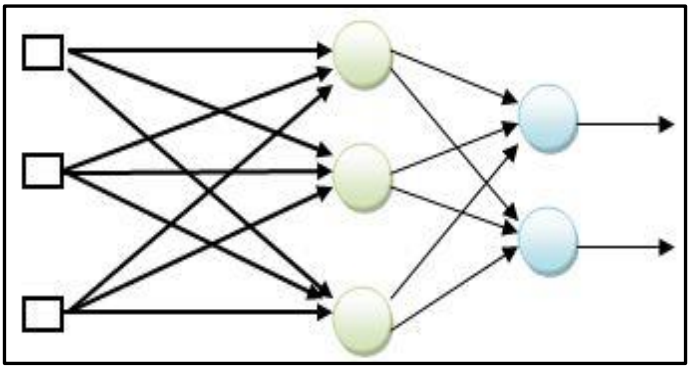


Figure 1.5– Réseaux Multi-couche. [04]

1.6.3 Réseaux récurrents (feed-back)

Le réseau cyclique est différent du réseau précédent car il est périodique. L'existence de cycles a un impact profond sur la capacité d'apprentissage et les performances du réseau, en particulier en rendant le système dynamique.

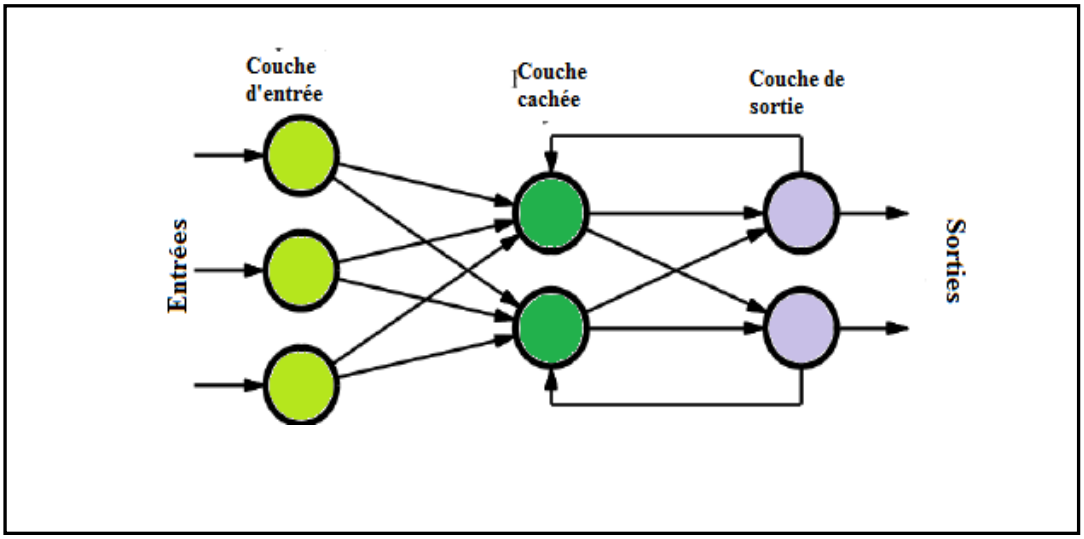


Figure 1.6– Réseaux récurrents.[04]

1.7 Les types de Deep Learning

Le deeplearning est un domaine en évolution rapide, avec de nouvelles architectures, variantes ou algorithmes apparaissant chaque semaine. Cependant, il n'est pas toujours possible de comparer les performances de ces algorithmes car ils ne sont pas évalués sur le même ensemble de données.

1.7.1 Perceptron multi-couches (MLP)

Nous commençons donc par présenter le plus classique des réseaux de neurones : le perceptron multi-couches (MultiLayer Perceptron en anglais ou MLP) qui est un réseau de neurones acyclique (Feed-Forward Neural Network - FFNN) structuré en couches. Par conséquent, un perceptron multicouche est constitué d'une couche d'entrée, d'une ou plusieurs couches intermédiaires dites cachées et d'une couche de sortie. Pour chaque couche, tous ses nœuds sont connectés à tous les nœuds de la couche précédente. La (**Figure 1.7)** donne une représentation d'un perceptron multi-couches avec deux couches cachées. [08]

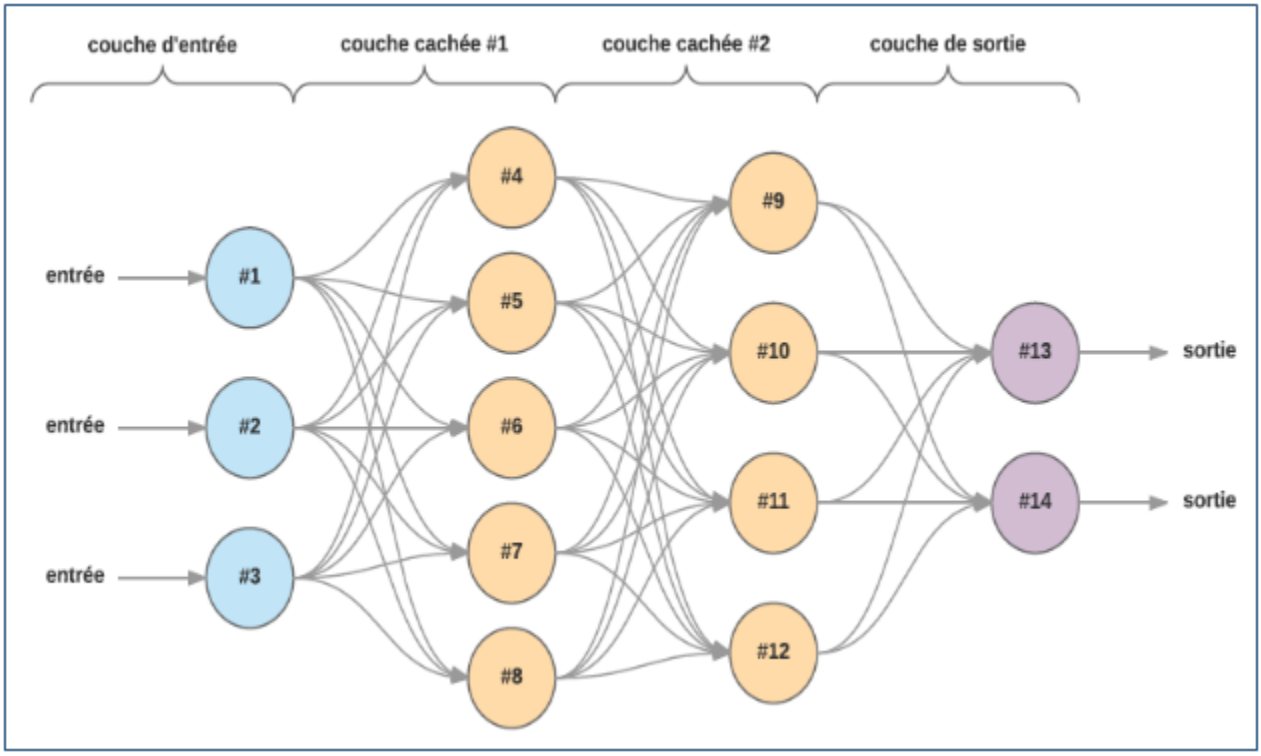


Figure 1.7–Réseau de neurones dense, acyclique et structuré en couches communément appelé perceptron multi-couches (MLP). [08]

1.7.2 **Réseaux de neurones profonds**

Les réseaux de neurones dits "profonds" (Deep Neural Networks en anglais) sont des perceptron multi-couches avec un nombre de couches supérieur à trois. Pendant longtemps, la méthode d'apprentissage de ce type de réseau de neurones acyclique ne permet pas la convergence vers un réseau de neurones puissant. Des avancées significatives dans les méthodes d'apprentissage et le choix de la fonction de transfert d'unités linéaires rectifiées (ReLU) minimisent l'effet de la dilution du gradient dans les couches inférieures du réseau, ce qui permet d'utiliser plus de réseaux de neurones. [07]

1.7.3 **Réseau de neurones récurrents**

L'idée derrière RNN est d'utiliser des informations séquentielles. Dans les réseaux de neurones traditionnels, nous supposons que toutes les entrées (et sorties) sont indépendantes les unes des autres. Mais pour de nombreuses tâches, c'est une très mauvaise idée. Si on veut prédire le prochain mot dans une phrase, il faut connaître les mots qui sont venus avant. Les RNN sont appelés récurrents, car ils exécutent la même tâche pour chaque élément d'une séquence, la sortie étant dépendante des calculs précédents. [07]

Une autre façon de penser les RNN est qu'ils ont une « mémoire » qui capture l'information sur ce qui a été calculé jusqu'ici. En théorie, les RNN peuvent utiliser des informations dans des séquences arbitrairement longues, mais dans la pratique, on les limite à regarder seulement quelques étapes en arrière. [09] [10] [11]. Il est utilisé pour :

- La modélisation du langage et génération de texte
- La traduction automatique
- La reconnaissance vocale
- La description des images

1.7.4 **Les réseaux de neurones convolutifs**

Convolutional Neural Network (CNN) (réseaux de neurones convolutifs) est un réseau neuronal spécialisé utilisé pour traiter les données avec une topologie en forme de grille.Dans les domaines de la reconnaissance et de la classification d'images et de vidéos, ces méthodes se sont avérées très efficaces. CNN a réussi à identifier les visages, les objets, panneaux de circulation et auto-conduite des voitures [12]. Récemment, les CNN ont été efficaces dans plusieurs tâches de traitement du langage naturel (telles que la classification des phrases) [13][14] [15].

Un réseau convolutif est un type de réseau de neurones feed-forward [16], Cette classe de réseaux à une ou plusieurs couches de neurones cachés (couches cachées) entre les couches d'entrée et de sortie. Chaque couche a une connexion depuis la couche supérieure et une sortie sur la couche suivante, de sorte que la propagation du signal se fait de manière linéaire, sans cycles et sans interconnexions. Ce type d'architecture fournit au réseau une perspective globale car il augmente les interactions entre les neurones [17] [18] [19].

1.8 **Exemples d'application de Deep Learning**

Les applications du Deep Learning sont utilisées dans divers secteurs, de la conduite automatisée aux dispositifs médicaux.

Grace au deep Learning nous pouvons maintenant

- Ajouter des sons à des films silencieux.
- Faire de la traduction automatique.
- Faire de classification des objets en photographies.
- Générer d'écriture automatique.
- Génération de légende d'image.
- Jeu automatique.

1.9 Conclusion

Dans ce chapitre, nous avons défini l'intelligence artificielle, Machine learning et Deeplearning et la différence entre eux. Nous avons montré les déférents types d'apprentissage automatique, ensuite nous avons expliqué que les réseaux de neurones artificiels sont calculatoires et sont des systèmes inspirés par les processus biologiques qui se produisent dans le cerveau humain, et on a montré ces architectures et les déférents types des réseaux de neurones. Et enfin on a cité quelquesexemples d'application de Deep Learning.

2 Chapitre2 : Word Embeddings

2.1 Introduction

Dans l'application Traitement Automatique du langage Naturel (TALn), nous devons travailler avec des données textuelles. Eh bien, nous ne pouvons pas directement alimenter nos données textuelles pour l'entraînement dans nos modèles l'apprentissage automatique, d'apprentissage profond, etc. Que ce soit la régression, la classification ou toute autre tâche TALn, nous devons convertir nos données textuelles en une forme numérique qui peut être alimentée en modèles pour un traitement ultérieur.

Word Embedding convertit les données textuelles en données numériques d'une certaine forme. En général, l'incorporation de mots convertit un mot en une sorte de représentation vectorielle.

2.2 Traitement Automatique du Langage Naturel

Le Traitement Automatique du Langage Naturel est un ensemble de méthodes et de programmes permettant aux ordinateurs de traiter des données linguistiques, Il s'agit également d'une série d'opérations ou de calculs que la machine doit effectuer. Son objectif est de traiter des données linguistiques (textes) exprimées en langage dit ”naturel”. En plus la conception de programmes capables de traiter automatiquement des données linguistiques de type textes écrits, dialogues écrits ou oraux et des unîtes linguistiques (mots, phrases, énonces, ...). [20]

2.3 La relation entre Word Embeddings et TALN

Le TALN est un sous-domaine de l'apprentissage automatique (ML) qui traite du langage naturel, généralement sous la forme de texte, lui-même composé d'unités plus petites comme des mots et des caractères. Le traitement des données textuelles est problématique, car nos ordinateurs, scripts et modèles d'apprentissage automatique ne peuvent pas lire et comprendre le texte dans un sens humain. [21]

Le Word Embeddings (WE) est l'approche dominante de ce problème et est si répandue que leur utilisation est pratiquement supposée dans tout projet de TALN. Quand vous commenciez un projet de classification de texte, d'analyse des sentiments ou de traduction automatique, il y a de fortes chances que vous commenciez par télécharger des Embeddingspré-calculées ou par

réfléchir à la méthode à utiliser pour calculer votre propres Word embeddings à partir de votre ensemble de données. [21]

2.4 Les Word embeddings

Le Word Embedding [22] spécifie un ensemble de techniques d'apprentissage automatique qui visent à représenter les mots ou les phrases d'un texte par des vecteurs de nombres réels. Le caractère de cette nouvelle représentation est que les mots qui apparaissent dans des contextes similaires ont des vecteurs correspondants relativement proches. Par exemple si on présente les mots « chien » et «chat» par des vecteurs dans un espace vectorielon trouve qu'ils sont relativement proches dans. Cette technique est basée sur une hypothèse (appelée « de Harris » ou « distributionalhypothesis ») que les mots qui apparaissent dans des contextes similaires ont des significations connexes.

Par rapport aux modèles vectoriels, la technologie d'incorporation de mots, par exemple, réduit la dimensionnalité de la représentation des mots, simplifiant ainsi les tâches d'apprentissage impliquant ces mots, car ils sont moins perturbés par la dimensionnalité.

Les WE caractérisent chaque mot par un ou plusieurs vecteurs denses, de faible dimension ayant des éléments réels, capturant les spécificités latente (de contexte) du mot et les propriétés syntaxiques et sémantiques utiles.

Un format Word Embedding essaie généralement de mapper un mot à l'aide d'un dictionnaire sur un vecteur. Décomposons cette phrase en détails plus fins pour avoir une vue claire. Prenonsl'exemple [23]

Phrase = « Word Embeddings are Word converted into numbers **».**

Un mot dans cette **Phrase**peut être 'Embeddings ou 'numbers etc.

Un dictionnaire peut être la liste des mots uniques de la **Phrase,** un dictionnaire peut ressembler à ['Word','Embeddings','are','Converted','into','numbers'].

Une représentation vectorielle d'un motparle« one-hotvector »c'est-à-dire le 1 représente la position ou le mot existe et 0 partout ailleurs. Alors, la représentation vectorielle du mot'numbers' dans ce format selon le dictionnaire ci-dessus est [0,0,0,0,0,0,1] et la représentation du mot 'Converted' est [0,0,0,1,0,0].

2.5 Les méthodes de Word embeddings

Le choix d'une bonne représentation des mots est un facteur très important pour optimiser les résultats des taches de traitement automatique de langage écrit ou oral. Il y a de différentes méthodes dans le sens de représentation des mots, ces méthodes connues sous le nom de « Word Embeddings » sont capables de représenter des mots par un vecteur de réels continus qui correspond à la position du mot dans un espace multidimensionnel. Ces méthodes sont utilisées pour des Modelés de Langage Neuronaux, ainsi que pour de nombreuses taches en traitement automatiques de langages. Ces méthodes de représentation sont intéressantes car non seulement elles permettent de diminuer l'espace de dimensionnalité, on ne stocke plus un dictionnaire en entier mais uniquement un espace de vecteurs continus, l'utilisation de vecteurs de mots produira des informations sémantiques. Nous allons passer en revue deux techniques les plus populaires qui peuvent être utilisées pour apprendre le Word embeddings à partir de données textuelles. [24]

2.5.1 GloVe(Global Vectors for Word Representation)

GloVe est un algorithme d'apprentissage non supervisé pour l'obtention de représentations vectorielles de mots. Il s'agit d'un projet open source de l'Université de Stanford. Il génère une "incorporation de mots" en agrégeant la matrice globale de cooccurrence mot-mot dans le corpus. Les plongements résultants montrent des sous-structures linéaires intéressantes de mots dans l'espace vectoriel. Elle combine les avantages des deux principales séries de modèles de la littérature, la factorisation matricielle globale et la méthode de la fenêtre contextuelle locale. Le modèle utilise efficacement les informations statistiques en entraînant uniquement les éléments non nuls dans la matrice de cooccurrence mot-mot [25].

GloVe à deux objectifs principaux:

- créer des vecteurs de mots qui capturent le sens dans un espace vectoriel.

- utiliser des statistiques de comptage globales au lieu de simplement des informations locales.

2.5.2 Objectif de GloVe

L'objectif de formation de GloVe est d'apprendre des vecteurs de mots tels que leur produit scalaire est égal au logarithme de la probabilité de cooccurrence des mots. Du fait que le logarithme d'un rapport est égal à la différence des logarithmes, cet objectif associe (le

logarithme de) des rapports de probabilités de cooccurrence avec des différences vectorielles dans l'espace vectoriel des mots. Étant donné que ces rapports peuvent coder une certaine forme de signification, ces informations sont également codées sous forme de différences vectorielles. Pour cette raison, les vecteurs de mots résultants fonctionnent très bien sur les tâches d'analogie de mots, telles que celles examinées dans le package word2vec.[22]

2.5.3 Word2vec

Le WORD2VEC est un algorithme de Word embeddings est parmi les algorithmes les plus connus, Il a été développé par l'équipe de recherche Google pour capturer le contexte des mots tout en fournissant une méthode très efficace de prétraitement des données textuelles brutes. Ce modèle prend en entrée un grand corpus de documents et génère un espace vectoriel de plusieurs centaines de dimensions. Chaque mot du corpus se voit attribuer un vecteur unique dans l'espace vectoriel. Le concept puissant de word2vec est que les mots représentés par des vecteurs de mots proches les uns des autres dans l'espace vectoriel ont non seulement la même signification, mais ont également le même contexte [27].

Word2vec est une méthode de représentation vectorielle de mots basée sur les réseaux de neurones artificiels [24]. Word2vec propose deux structures de réseaux de neurones, le modèle CBOW (ou Countinious Bag of Words) et le modèle Skip-Gram. Ces modèles sont entrainés à partir de « mots centraux » et du contexte dans lequel ces derniers apparaissent respectivement, les n mots qui les précèdent et les n mots qui les succèdent. Par exemple, dans la phrase « la grande maison est blanche », « maison » est le mot central, « la », « grande », « est », « blanche »est le contexte dumot« maison » pour une fenêtre de 2 mots [28].

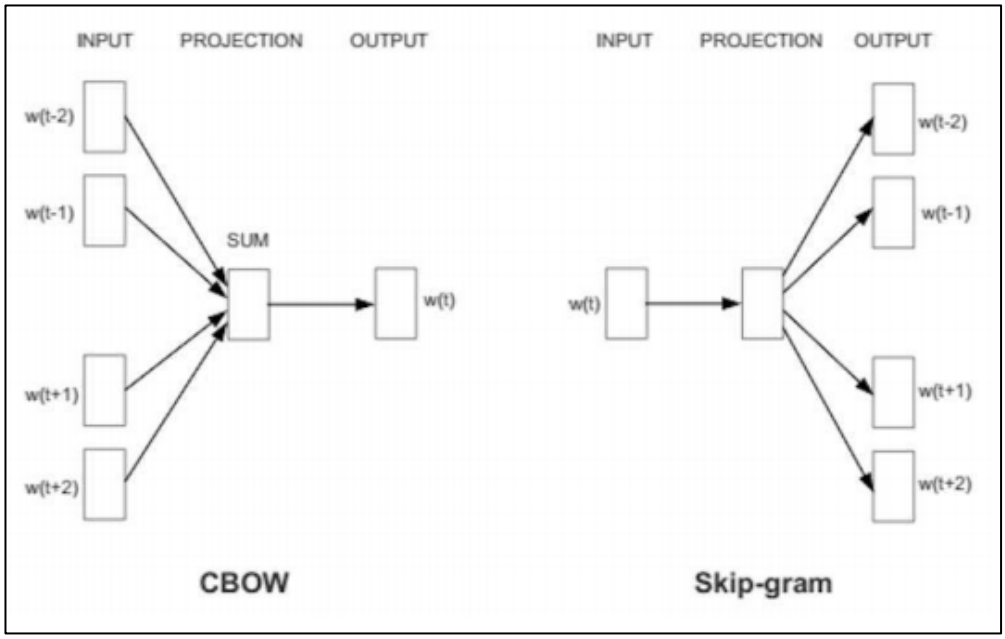


Figure 2.1– Exemples des architectures Skip-Gram et CBOW de Word2Vec. [28]

Ces réseaux de neurones sont entrainés de façon à prédire un mot central étant donné un contexte (CBOW) ou vice-versa (Skip-gram). Ces modèles sont structurés en trois couches

- **Couche d'entrée** contient le mot central pour le modèle Skip-Gram et un sac-de mots contenant la fenêtre de contexte du mot centrale pour le modèle CBOW.
- **Couche intermédiaire**correspond à la projection des mots d'entrée dans la matrice des poids.
- **Couche de sortie** correspond à la prédiction du modèle par l'utilisation de la fonction « Softmax », soit le mot central pour le modèle CBOW et un sacdemots contenant la fenêtre de contexte du mot centrale pour le modèle Skip-Gram.

Pour des raisons de complexité algorithmique liéeà l'utilisation de la fonction « Softmax » dans la couche de sortie, deux améliorations sont proposées dans [28] le « Softmax hiérarchique »en anglais (HierarchicalSoftmax) et l'« Echantillonnage négative » en englais (NegativeSampling). La combinaison de ces deux alternatives a permis un accroissement important de la vitesse de traitement et l'entrainement de modèles sur de grandes quantités de textes.

2.5.4 Architectures de Word2Vec

Deux architectures ont été proposées pour apprendre Word2vec, le modèle de sac continu de mots (CBOW continu bag of words) et le modèle skip-gramme.

2.5.4.1 Le modèleCBOW

L'architecture CBOW est un réseau de neurones simple et log-linéaire. A partir d'un contexte {m_{-n}, m_{-n+1}, ..., m_{n-1}, m_n} le modèle CBOW apprend à prédire un mot centrale m_c ou m_{-n} m_{-n+1} ..m_c .. m_{n-1} m_n est une phrase. [28]

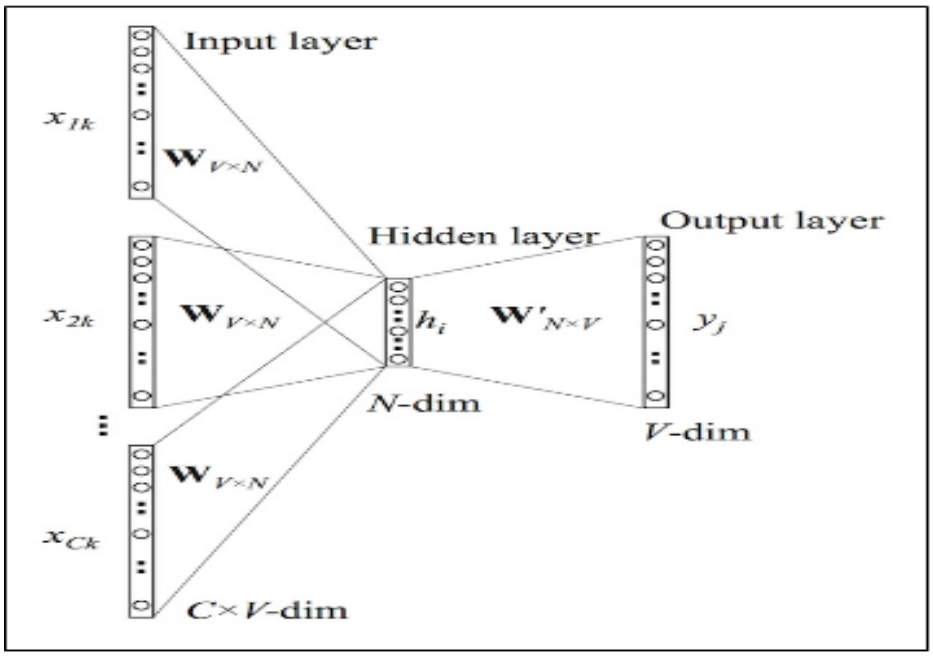


Figure 2.2– Architecture du modèle CBOW. [29]

Dans cette configuration chaque mot mi est représenté à la couche d'entrée par un vecteur Xi (x1 , x2 , ..., xk , ..., xv) où

- V est la taille du vocabulaire.
- k est le rang du mot mi dans le vocabulaire.
- x_k = 1 et $x_{k'}$ = 0 pour k ≠ k′ , un seul composant est égale à 1 les autres sont nulles.

Le passage à la couche intermédiaire se fait par le produit de la moyenne des vecteurs des mots de la fenêtre de contexte et de la matrice de pondération entre la couche d'entrée et la couche intermédiaire :

$$h = \frac{1}{C} \times W \,.\, (X_1 + X_2 + \cdots + X_{c-1} + X_c) \qquad (2.1)$$

Où

- h : Un vecteur de dimension N correspondant à la projection de la fenêtre de contexte sur la couche intermédiaire.
- C :La taille de la fenêtre de contexte.
- W : Matrice de taille V × N correspondant à la matrice de passage de la couche d'entrée vers la couche intermédiaire.
- $X1 + X2 + ... + XC-1 + XC$: La somme des vecteurs représentants les mots de la fenêtre de contexte.

La couche de sortie est calculée à partir de la fonction ≪Soft max≫ de la manière suivante

$$y_j = \frac{e^{(v'_{mj} \times h)}}{\sum_{k=1}^{V} e^{(v'_{mk} \times h)}} \qquad (2.2)$$

Où

- $y_j$est la j_{eme} composante du vecteur Y de taille V de la couche de sortie et qui correspond à la probabilité que le j_{eme} mot du vocabulaire appartienne au contexte de la couche d'entrée.
- v'_{mj}Est la j_{eme} ligne de la matrice W′ qui est la matrice de pondération entre la couche interm´ediaire et la couche de sortie.

La rétro-propagation est par la suite appliquée pour ajuster les composantes des matrices de pondération W et W′, de façon à optimiser la prédiction du réseau de neurones. L'architecture CBOW est plus efficace que son homologue Skip-gram et capture une meilleure représentation des mots fréquents [28].

Avantages de CBOW

- Il n'a pas besoin d'avoir d'énormes exigences de RAM comme celle de la matrice de co-occurrence où il doit stocker trois énormes matrices.
- Il est censé exécuter des méthodes supérieures aux méthodes déterministes (en général).

Inconvénients de CBOW

- CBOW prend la moyenne du contexte d'un mot (comme vu ci-dessus dans le calcul de l'activation cachée). Par exemple, Apple peut être à la fois un fruit et une entreprise, mais

CBOW prend une moyenne des deux contextes et la place entre un cluster pour les fruits et les entreprises.

- La formation d'un CBOW à partir de zéro peut prendre une éternité s'il n'est pas correctement optimisé.

2.5.4.2 Le modèle skip-gram

L'architecture Skip-Gram est l'image inverse de l'architecture CBOW. Contrairement au modèle CBOW, le modèle Skip-Gram apprend à partir d'un mot central mc à prédire son contexte {m_{-n}, m_{-n+1}, ..., m_{n-1}, m_n}

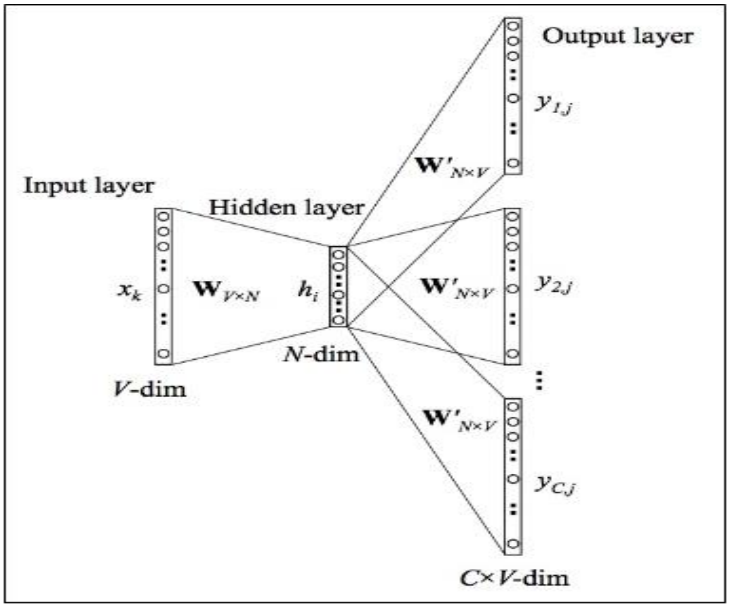


Figure 2.3– Exemples de l'architecture Skip-Gram de Word2Vec. **[29]**

Dans cette configuration la couche d'entrée est constituée d'un seul vecteur X_i (x_1, x_2, …, x_k, …, x_v) qui représente le mot central où $x_k = 1$ et $x_{k'} = 0$ pour $k \neq k'$, un seul composant est égale à 1 les autres sont nulles.

Le passage à la couche intermédiaire se fait par la somme pondérée de l'entrée par la matrice de pondération entre la couche d'entrée et la couche intermédiaire

$$h = W \times X \qquad (2.3)$$

Où

W matrice de taille W × V correspondant à la matrice de passage de la couche d'entrée vers la couche intermédiaire.

La couche de sortie est calculée en appliquant la fonction ≪Softmax≫ de la manière suivante

$$y_{c,j} = \frac{e^{(v'_{mj} \times h)}}{\sum_{k=1}^{V} e^{(v'_{mk} \times h)}} \qquad (2.4)$$

Où

- $y_{c,j}$ Le j_{eme} composant du ceme vecteur dans la fenêtre de contexte.
- v'_{mj} La j_{eme} ligne de la matrice W′ qui est la matrice de pondération entre la couche intermédiaire et la couche de sortie.

De la même façon que pour le modèle CBOW les composantes des matrices de pondération W et W' sont ajustées pour optimiser la capacité de prédiction du modèle en appliquant la retro-propagation.

Avantage du model Skip-Gram

1. Le modèle Skip-gram peut capturer deux sémantiques pour un seul mot. C'est-à-dire qu'il aura deux représentations vectorielles d'Apple. Un pour l'entreprise et un autre pour le fruit.
2. Skip-gram avec sous-échantillonnage négatif surpasse généralement toutes les autres méthodes.

2.5.5 Les différences entre GloVe et word2vec

GloVe et Word2vec sont tous deux des modèles non supervisés pour générer des vecteurs de mots. La différence entre eux est le mécanisme de génération de vecteurs de mots.

Les vecteurs de mots générés par l'un ou l'autre de ces modèles peuvent être utilisés pour une grande variété de tâches telles quetrouver des mots sémantiquement similaires à un mot, représentant un mot lorsqu'il est entré dans un modèle en aval. Une représentation incorporant un

mot d'un mot capture plus d'informations sur un mot qu'une simple représentation ponctuelle du mot, puisque la première capture la similitude sémantique de ce mot avec d'autres mots tandis que la seconde représentation du mot est équidistante de tous les autres mots [27].

2.5.6 Utilisation des « Word Embeddings »

Vous disposez de plusieurs options pour utiliser « Word Embeddings »dans votre projet de traitement du langage naturel.

- **Apprenez une intégration**

Vous pouvez choisir d'apprendre un mot incorporant pour votre problème, Cela nécessitera une grande quantité de données textuelles pour garantir l'apprentissage des incorporations utiles, telles que des millions ou des milliards de mots [30].

Vous avez deux options principales pour entraîner votre intégration de mots

- Apprenez-le en mode autonome, où un modèle est formé pour apprendre l'incorporation, qui est enregistrée et utilisée comme partie d'un autre modèle pour votre tâche ultérieure. C'est une bonne approche si vous souhaitez utiliser la même intégration dans plusieurs modèles.
- Apprendre ensemble, où l'intégration est apprise dans le cadre d'un grand modèle spécifique à une tâche. C'est une bonne approche si vous avez l'intention d'utiliser l'incorporation sur une seule tâche.

- **Réutiliser une incorporation**

Il est courant que les chercheurs mettent à disposition gratuitement des wordembeddingpré-formés, souvent sous une licence permissive, afin que vous puissiez les utiliser dans vos propres projets académiques ou commerciaux.

Par exemple, les intégrations de mots word2vec et GloVe sont disponibles en téléchargement gratuit.Ceux-ci peuvent être utilisés sur votre projet au lieu de former vos propres intégrations à partir de zéro [30].

Vous avez deux options principales en ce qui concerne l'utilisation des incorporations pré-formées

- Statique, où l'incorporation est maintenue statique et est utilisée comme un composant de votre modèle. Il s'agit d'une approche appropriée si l'incorporation convient bien à votre problème et donne de bons résultats.
- Mise à jour, où l'incorporation pré-entraînée est utilisée pour amorcer le modèle, mais l'incorporation est mise à jour conjointement pendant l'entraînement du modèle. Cela peut être une bonne option si vous cherchez à tirer le meilleur parti du modèle et à l'intégrer à votre tâche.

2.6 Conclusion

Le Word embeddings est une approche dominante qui vise à transformer des données textuelles en données numérique.

Dans ce chapitre on a défini le traitement automatique du langage naturel, après on a mentionné la relation entre le TALN et le Word embeddings. Ensuite on a défini le Word embeddings, Utilisation des Word embeddings et ses méthodes (GloVe et Word2vec), comme on a mentionné les objectifs de ces deux méthodes et la différence entre eux.

3 Chapitre3 : Extractions des motifs fréquents

3.1 Introduction

Data Mining également appelée exploration de données, fouille de données ou extraction de connaissances à partir de données, vise à extraire ou à obtenir des connaissances à partir d'une grande quantité de données grâce à des méthodes automatiques ou semi-automatiques. Il recommande d'utiliser un ensemble d'algorithmes de diverses disciplines scientifiques, telles que les statistiques, l'intelligence artificielle ou l'informatique, pour construire des modèles basés sur des données, c'est-à-dire trouver des structures ou des motifs intéressants basés sur des critères fixés au préalable et en extraire autant que possible connaissance. [31]

De toutes les étapes du processus, du point de vue de l'algorithme, l'étape d'exploration de données est la partie la plus compliquée. De nombreuses méthodes existent alliant statistiques, mathématiques et informatique, de la régression linéaire à l'extraction de motifs fréquents.

Sous ce vocable sont regroupées les méthodes mathématiques et les algorithmiques permettant d'effectuer les taches de fouille de donnée. On peut citer Extractions des motifs et Extractions des motifs séquentiels.

3.2 Extractions des motifs

La recherche des régularités dans les bases de données est l'idée principale du data mining. Ces régularités s'expriment sous différentes formes.Lors de l'analyse des paniers d'achat des consommateurs, l'extraction des itemsets comprend la mise en évidence de la cooccurrence entre les produits achetés, c'est-à-dire la détermination des produits (items) qui sont « souvent » achetés simultanément. On parle alors d'itemsets fréquents. [32]

3.2.1 Concepts de base du processus d'extraction des motifs fréquents

3.2.1.1 Cas du 'panier de la ménagère

Tableau 3.1: Base detickets de caisse.[32]

	Huile	Café	Tomate	Beure	Pattes
Ticket1	Huile, Tomate, Beure				

Ticket2	Café, Tomate, Pattes
Ticket3	Huile, Café, Tomate, Pattes
Ticket4	Café, Pattes
Ticket5	Huile, Café, Tomate, Pattes
Ticket6	Café, Tomate, Pattes

- **Item :** Un item correspond à un produit par exemple café.
- **Support :** Le support d'un item est égale au nombre de transaction dans lesquelles il apparait.

Exemple : support (tomate)=5/6.

- **Itemset :** Un itemsetest un ensemble d'item**.**Etant donné un ensemble de produits P={p1, p2, …, pN}, tout sous ensemble de P est dit Itemset.

Exemple :

Sur l'ensemble {Huile, Café, Tomate, Beure, Pattes}, des Itemset sont:

Itemset de taille 3 : {Café, Tomate, Pattes}.

Itemset de taille 1 : {Beure}.

Itemset de taille 0 : {}.

Remarque :

- L'ordre des éléments dans un Itemset ne compte pas. Donc {Café, Tomate} = {Tomate, Café}.

- Le Nombre d'itemsets possibles de R est: 2^|P|= 2Card(P) = Taille des partitions de P.

3.2.1.2 Représentation canonique

Afin de faciliter les calculs d'extraction des motifs, une représentation canonique (standard) sous forme de codes binaires des produits dans un itemset est utilisée. Ce principe est illustré pour l'exemple précédent dans la matrice suivante:

Tableau 3.2:représentation canonique sous forme de codes binaires. [32]

	Huile	Café	Tomate	Beure	Pattes
Ticket1	1	0	1	1	0
Ticket2	0	1	1	0	1
Ticket3	1	1	1	0	1
Ticket4	0	1	0	0	1
Ticket5	1	1	1	0	1
Ticket6	0	1	1	0	1

-Itemset fréquent : un itemset est dit fréquent si son support est supérieur à un seuil défini à l'avance, paramètre de l'algorithme de recherche. Support(P) $\geq \gamma$ [33][32][34].

3.2.2 Algorithme d'extraction des motifs

Afin de résoudre le problème de l'extraction de motifs, de nombreux algorithmes ont été développés. Les algorithmes de recherche de motifs couramment utilisés peuvent être divisés en deux catégories, à savoir la méthode de calcul du motif de grille sur la largeur et la méthode de calcul du motif de grille sur la profondeur. Historiquement, les premières méthodes qui sont apparues sont les méthodes en largeur.

3.2.2.1 Algorithme Apriori

Le premier algorithme à résoudre le problème de recherche de motifs fréquents est l'algorithme A-Priori. Le principe de l'algorithme A-Priori est de calculer les motifs du treillis en largeur d'abord. Les motifs sont calculés niveau par niveau. [32]

A-Description générale de l'Algorithme Apriori

L'idée de base d'Apriori est d'effectuer une extraction par niveaux comme suit:

- Recherche les motifs fréquents de longueur 1.
- Combinaison de ces motifs pour obtenir des motifs de longueur 2 et retenir seulement les plus fréquents.

- Combinaison de ces motifs pour obtenir des motifs de longueur 3 fréquents selon la même démarche.
- Répétez le processus de sélection de combinaison jusqu'à ce que la taille maximale soit atteinte.

L'analyse de cet algorithme montre qu'Apriori repose sur deux constations fondamentales du point de vue de la notion des motifs fréquents:

Tout sous-motif d'un motif fréquent est fréquent

Exemple : si le motif : Café, Tomate est fréquent:

Nous avons alors obligatoirement:

Support(café,Tomate) $\geq \sigma s \Rightarrow$ Support(café) $\geq \sigma s$ et Support(Tomate) $\geq \sigma s$.

Tout sur-motif d'un motif non fréquent est non fréquent

Exemple: si le motif: Café, Tomate est non-fréquent:

Nous avons alors obligatoirement: Support(café,Tomate) $< \sigma s \Rightarrow$ Support(café,X) $< \sigma s$, $\forall X \in P$ .Voici une version algorithmique Apriori:

Apriori

Entrées : (O,P,R) : une base de données formelle $\sigma_s \in [0 ;1]$

Sortie : l'ensemble des motifs frequents de la base, relativement au seuil σ_s.

Debut

$i \leftarrow 1$

$C_1 \leftarrow$ ensemble des motifs de taille 1

tant que $C_i \neq \emptyset$ **faire**

Calculer le support de chaque motif $m \in C_i$

$F_i \leftarrow \{m \in C_i \mid support(m) \geq \sigma_s\}$

$C_{i+1} \leftarrow$ generer-candidats(F_i)

$i \leftarrow i + 1$

fin-tant que

retourner $\bigcup_{i \geq 1} F_i$

Fin

Figure 3.1: Algorithme Extraction des itemsets avec Apriori.[34]

Exemple d'exécution d'Apriori :

L'exécution de Apriori sur la base formelle avec σs = 2/6 se passe est ainsi: 3, 5, 5, 1, 5.

Etape 1 : Génération des candidats de taille 1 :

– C1 = { a , b , c , d, e }.

– Supports : { 3/6 , 5/6, 5/6, 1/6, 5/6 }.

D'où F1 = { a , b , c , e }.

Premier résultat extrait: D n'est pas fréquent ⇒ aucun motif fréquent ne contiendra D.

Etape 2 : Génération de candidats de taille 2 , Combinaison des candidats de taille 1 de F1, 2 à 2 :

– C2 = {ab ,ac , ae , bc, be , ce }.

– Supports : {2/6 , 3/6 , 2/6, 4/6, 5/6, 4/6}.

Résultats: F2 = C2.

Etape 3 : Génération des candidats de taille 3 : Combinaison des des candidats de taille 2 de F2 et F1 :

– C3 = {abc ,abe , ace , bce }.

– Supports : { 2/6 , 2/6 , 2/6 , 4/6}.

Résultats: F3 = C3.

Etape 4 : Génération des candidats de taille 4 :

– C4 = {abce}.

– Supports : { 2/6}.

Etape 5 : Génération de candidats de taille 5 :

C5 = ∅

Donc : F5 = ∅.

Etape 6 : Résultat final: F1 ∪ F2 ∪ F3 ∪ F4.

Remarque : Apriori parcoure au fait le treillis des parties de P ordonné.

B-Avantages et inconvénients de Apriori

Avantages :

- Bonne performance par la réduction itérative des itemsets candidats.
- Facile à comprendre.

Inconvénients :

- Nombre d'itemsets (sous graphes) possibles peut-être très grand (ex : pour n = 100 on a 2^100 − 1 ≈ 10^30 itemsetspossibles).
- Nombre de parcours de la base de transactions égal à la taille du plus long itemset fréquent trouvé.
- Beaucoup d'itemsets fréquents ⇒ des règles redondantes [37].

3.2.2.2 Algorithme Eclat

Cet algorithme utilise le format vertical de la base de données, ou pour chaque itemset on dispose de son tidset, i.e. Contient toutes les transactions de cet ensemble d'itemsets. L'avantage du format vertical est de faciliter le calcul du support, car dans ce cas, il s'agit d'un problème d'intersection tidset. De plus, étant donné que seules les transactions impliquant des ensembles d'itemsets sont utilisées pour l'intersection, cela réduira automatiquement la taille de la base de données. Eclat effectue d'abord une recherche approfondie des ensembles d'itemsets fréquents sur la base du concept de classes équivalentes.[37]

Voici une version algorithmique Eclat :

Input: F_k= {I1,I2,…,In} // cluster of frequent k-itemsets.
Output: Frequent l-itemsets, l>k.

Bottom-Up(F_k) {

1. For all $I_1 F_k$
2. $F_k + 1 = \varnothing$
3. For all $I_j \in F_k$, i< j
4. $N = I_i \cap I_j$;
5. If N sup ≥ min_sup then
6. $F_k + 1 = F_k$ 1 ∪ N;
7. end
8. end
9. end
10. if $F_k + 1 \neq \varnothing$ then
11. Bottom-Up($F_k + 1$);
12. end
13. }

Figure 3.2: Algorithme extraction des itemsets avec Eclat. [38]

3.2.2.3 Algorithme FP-Growth (Frequent-Pattern Growth)

Une structure de données compacte appelée «Frequent-Pattern tree» est utilisée, qui fournit une solution au problème de la fouille de motifs fréquents dans les grandes bases de données de transactions. En stockant tous les éléments fréquents de la base de transactions dans une structure compacte, on supprimer la nécessiter de devoir scanner de façon répétée la base de transactions. De plus, en triant les éléments dans la structure compacte, on accélère la recherche des motifs. [39]

Le processus d'extraction de patterns se déroule en deux grandes étapes :

- La structure d'un arbre appelé FP-Tree(une représentation condensée de la base de données).
- Générer des motifs fréquents à partir de cette structure FP-Tree.
  L'arbre FP-tree est composé de:
- Arbre : mise à part la racine nulle, chaque nœud de l'arbre contient trois informations : l'item que représente ce nœud, sa fréquence, ainsi que le nœud suivant dans l'arbre.
- Index : contient la liste des items fréquents. Chaque item est associé à un pointeur qui indique le premier nœud de l'arborescence contenant l'item.

L'avantage de cette représentation de données est qu'il suffit de suivre les liens entre les nœuds pour connaître toutes les associations fréquentes d'éléments fréquents [40].

Exemple :

Soit la base de données suivante :

Tableau 3.3 : Exemple de base de transactions. **[41]**

Transaction iD	Itemsets
01	f,a,c,d,g,i,m,p
02	a,b,c,f,i,m,o
03	b,f,h,j,o
04	b,c,k,s,p

05	a,f,c,e,i,p,m,n

Cet ensemble de données à deux attributs et cinq instances, le premier attribut est Transaction Id et le deuxième attribut est essentiellement itemsets.

Les étapes pour résoudre ce problème :

1. Trouvez le support minimum de chaque item.

2. Trier l'ensemble d'itemstes fréquents dans l'ordre décroissant.

3. dessinez un FP-tree.

4. Exploitation d'un motif fréquent à partir de FP-tree.

Etape 1 :

Tableau 3.4 : Items associés à leur support [41].

Item	Support
A	3
B	3
C	4
D	1
E	1
F	4
G	1
H	1
I	1
J	1
K	1
L	2

M	3
N	1
O	2
P	3
S	1

Le Support minimum=3. Alors, supprimer les items qui ont un support inférieur à 3 :

Tableau 3.5 : les items qui ont un support supérieur ou égal à 3.

Item	Support
A	3
B	3
C	4
F	4
M	3
P	3

Etape 2 :

f = 4, c= 4, a= 3, b= 3, m= 3, p=3

Transaction ID	Itemset acheté	Itemset ordonnée
1	f,a,c,d,g,i,m,p	f,c,a,m,p
2	a,b,c,f,i,m,o	f,c,a,b,m
3	b,f,h,j,o	f,b
4	b,c,k,s,p	c,b,p
5	a,f,c,e,i,p,m,n	f,c,a,m,p

Tableau 3.6 : les itemsets de type ordonnée.

Etape 3 :

Insérez les transactions sous forme d'un arbre

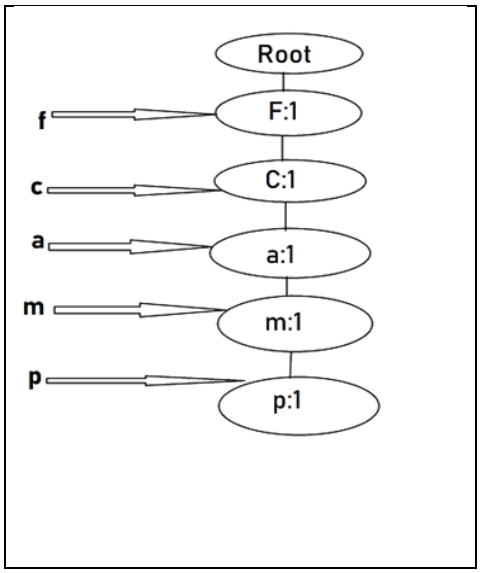


Figure 3.3: arbre de 1ère transaction.[41]

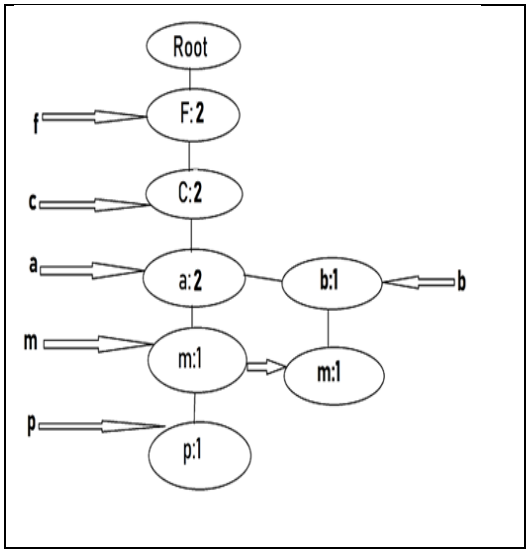


Figure 3.4:arbre de 2eme transaction.[41]

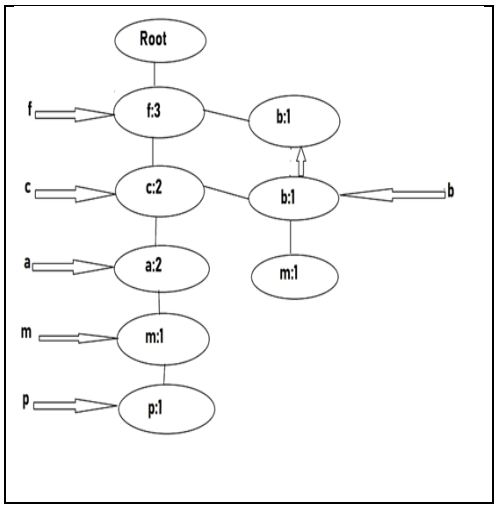


Figure 3.5:arbre de 3eme transaction.[41]

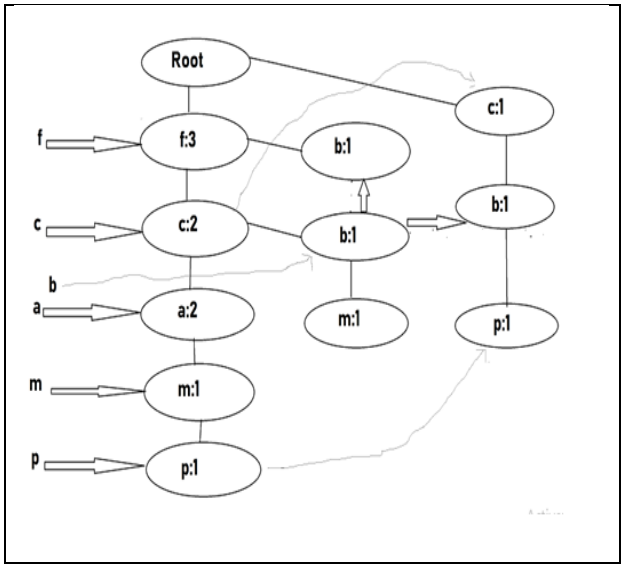


Figure 3.6: arbre de 4eme transaction.[41]

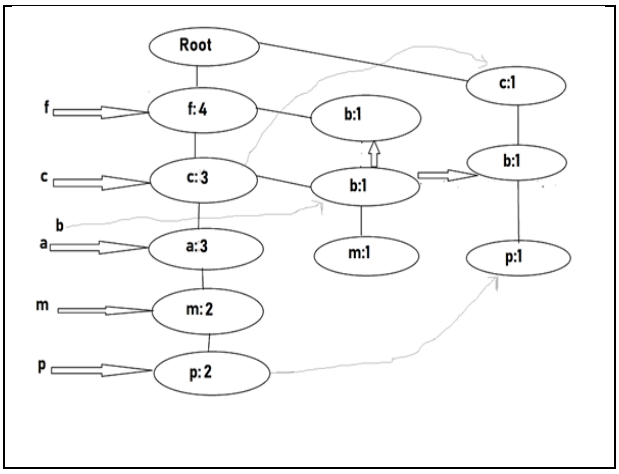


Figure 3.7:arbre de 5eme transaction.[41]

Etape 4 :

Tableau 3.7: Fouille du FP-Tree. [41]

item	Motifs conditionnels	FP-Tree conditionnels	Motifs fréquents
P	(fcam : 2)(cb : 1)	(c : 3) / p	(cp :3)
m	(fca : 2)(fcab : 1)	(f:3, c : 3. a:3) / m	(fcam :3)
B	(fca : 1)(f : 1)(c:1)	Empty	**Empty**
A	(fc : 3)	(f:3, c:3) / a	(fca :3)
C	(f : 3)	(f : 3) / c	(fc :3)
F	Empty	Empty	Empty

3.2.3 Discussion

Dans cette sous-section, nous discuterons de différents algorithmes pour l'extraction de motifs fréquente. Nous résumons les caractéristiques des différentes méthodes étudiées dans le tableau. La comparaison couvre les aspects suivants :

- Stratégie d'exploration :Cette propriété décrit la stratégie d'exploration des motifs fréquents.
- Caractéristiques : Cette propriété décrit les caractéristiques de l'approche en question. [42]

Tableau 3.8: Fouille du FP-Tree. **[42]**

Algorithme	Stratégie d'exploration	Caractéristiques
Apriori	Largeur d'abord	-Complet : Pas de perte d'informations. - Nombre considérable de l'accès à la base des transactions. - Sa performance est proportionnelle à son nombre

		de motifs candidats.
Eclat	Profondeur d'abord	-Deux balayages à la base des transactions. -Rapide :compte les supports des items avec les TIDs.
FPGrowth	Profondeur d'abord	-Deux balayages à la base des transactions. - Complet. - A chaque élément fréquent correspond un chemin dans l'arbre. -Structure compacte.

3.3 Extractions des motifs séquentiels

3.3.1 Motifs séquentiels

Le problème de fouille de motifs séquentiels a été proposé pour la première fois par Agrawal et al. (1995)

Les motifs séquentiels sont extraits à partir de séquences d'événements ordonnées et souvent sauvegardés dans des bases de données transactionnelles (voir l'exemple de la Table 3.9). Les motifs séquentiels ont pour objectif la découverte d'enchaînements fréquents dans les bases de données, avec des contraintes temporelles et l'identification des événements d'individus afin de pouvoir suivre leurs comportements séquentiels au cours du temps. [43]

3.3.1.1 Extraction de motifs séquentiels versus extraction d'itemsets

- La différence théorique entre les itemsets et les motifs séquentiels réside dans les contraintes temporelles. Alors que {pain, lait, figues} représente un itemset, {pain suivi par l'achat de lait et des figues dans les troisjours suivants} exprime un motif séquentiel.
- Mais il y a une différence pratique très importante qui complique l'adaptation d'un algorithme d'extraction d'itemsets à l'extraction de motifs séquentiels : si dans un itemset les items sont uniques, dans le cadre d'un motif séquentiel on peut avoir des répétitions d'items. [44]

3.3.2 Concepts généraux

Cette section présente les définitions et les principes généraux dans l'extraction de motifs séquentiels

3.3.2.1 Définitions

- **Une séquence**

Est une structure de données qui permet d'organiser un ensembled'éléments grâce à une relation d'ordre entre ces éléments [45], C'est une suite de transactions chronologiquement ordonnées et se rapportant à un même sujet.

Exemple : si un client achète des produits a, b, c, d, e et selon la séquence S = « (a) (bc) (d) (e) » , cela signifie qu'il a d'abord achète le produits a, puis les produits b et c ensemble, ensuite le produit d et finalement le produit e.

- **Base de données transactionnelles**

Une base de données de séquences D basée sur les items de I est un ensemble fini de paires (SID, T), appelées transactions, avec SID ∈ {1, 2, . . .} un identifiant et T ∈ T (I) une séquence construite sur I.

Tableau 3.9-Exemple base de données de séquences. [45]

Client	Date	Items
C1	01/04/2020	{paincolat}
C1	02/04/2020	{chips pain }
C1	05/04/2020	{pain yaourt}
C1	09/04/2020	{pain}
C2	04/04/2020	{chips}
C2	11/04/2020	{chocolat}
C2	15/04/2020	{pain yaourt chocolat}
C3	03/04/2020	{chips pain }
C3	08/04/2020	{pain yaourt}

C4	07/04/2020	{chips}
C4	09/04/2020	{chips}
C4	14/04/2020	{yaourt}

- **Inclusion**

Une séquence S'=<s'1 s'2 …… s'n> est une sous-séquence de S=<s1 s2 …… sn> s'il existe des entiers a1<a2<…an tels que s'1 ⊆sa1, s'2 ⊆ sa2,....., s'm ⊆ sam. On dit que S'est incluse dans S [46].

- **Fréquence d'une séquence**

Une séquence est considérée fréquente, si le support de cette séquence est supérieur ou égal au support minimum. Celui-ci est introduit par le client afin de mesurer la pertinence d'une séquence [47].

- **Support d'une séquence :** est le pourcentage de clients qui supportent cette séquence.

3.3.3 ALGORITHMES D'EXTRACTION DE MOTIFS SEQUENTIELS

Reconnaître les événements personnels qui changent avec le temps est non seulement utile mais nécessaire pour pouvoir suivre leur comportement séquentiel.Les méthodes existantes différentes essentiellement sur la manière de parcourir l'espace de recherche (largeur d'abord ou profondeur d'abord) et sur les structures de données utilisées pour indexer la base de données et faciliter une énumération rapide, afin de réduire le temps d'exécution ainsi que les besoins en espace disque ou en mémoire vive[48]. On peut diviserLes algorithmes d'extraction des motifs séquentiels en deux catégories :

- Méthodes horizontales.
- Méthodes verticales.

3.3.3.1 Méthodes horizontales

3.3.3.1.1 Algorithme GSP (GeneralizedSequential Patterns)

(Srikant et Agrawal 1996) c'est le premier qui proposé cet algorithme pour résoudre le problème des motifs séquentiels [43], il utilise le principe de l'algorithme Apriori, utilisé pour

l'extraction d'itemsets fréquents, elle repose sur un parcours par niveau de l'espace de recherche s'appuyant sur le paradigme générer- élaguer.

GSP est un algorithme basé sur la méthode générer-élaguer, qui peut minimiser le nombre de transferts sur la base de données. La technique généralement utilisée par les algorithmes de recherche de séquences est basée sur une création de candidats (par I-extension « ajout dans le dernier itemset » et S –extension « ajout d'une sous séquence ») suivie du test de ces candidats pour confirmer leur fréquence dans la base [49].

- **Génération de séquences candidatent**

GSP procédé à la génération des séquences candidates en deux phases : Phase d'légale et phase de jointure. Lors de la phase d'légale. GSP aboutit l'ensemble de1-séquence fréquentes qui servira comme pont de départ pour générer les 2- séquences candidates qui constitue l'amorce de la phase de jointure. GSP effectue un balayage de la base de donne et généré les 2-séquencecandidates potentielles ou fréquentes, tout en calculant leurs support servira d'amorce pour le prochain balayage, et ainsi de suite. Plus généralement CK ou les K-séquences candidates sont générées en joignant les k-séquence fréquentes (lk+1 avec luimême).

Soient est exactement la mémé séquences fréquentes peut être jointe a'a' laquelle le dernier item d'aura été rajouté. L'exemple suivant (de Agrawal et la 1993) explique la phase jointure de deux séquences Une séquence joindra une séquence S2si la sous séquence obtenue en retirant le premier item de est la même que celle obtenue en retirant le dernier item deLa séquence ? Générée notée S = x n'est autre que S1 à laquelle a été ajouté à la fin le dernier item de qui devient séparé dans S s'il l'était dans, sinon, il est ajouté au dernier itemset[50].

Exemple de jointer :

$$\begin{array}{l} \{A,B\}\{C\} \\ <\{B\}\{C,D\} \\ \hline <\{A,B\}\{C,D\}> \end{array} \qquad \begin{array}{l} <\{A,B\}\{C\}> \\ <\{B\}\{C\}\{E\} \\ \hline <\{A,B\}\{C\}\{E\}> \end{array}$$

Figure 3.8 : Jointure de séquences dans GSP.

- **Algorithme GSP (D,ps,I)**

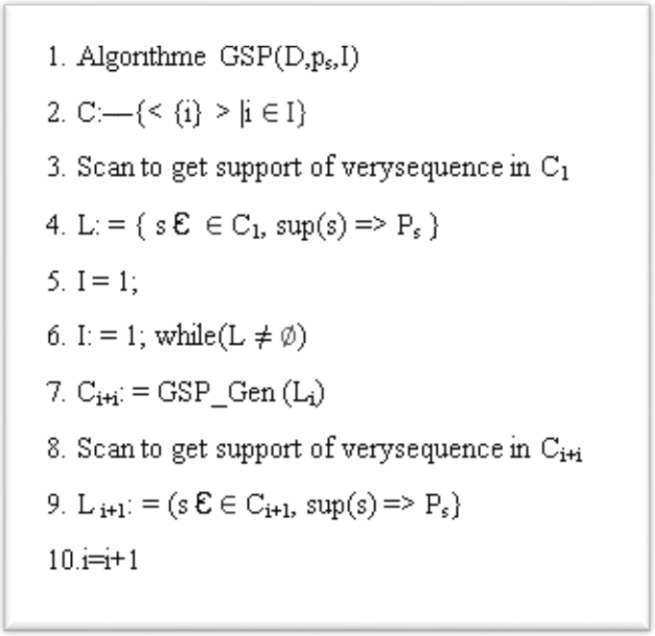


Figure 3.9 : Algorithme GSP.[**50]**

- **Calcul des supports**

L'algorithme GSP augmente de 1 le support de la séquence s pour chaque séquence Client contenant au moins une fois la séquence s en question. Dans le cas où aucune fenêtre d'événement ni max-gap min-gap ont été définis, l'algorithme ne fait que calculer le nombre de séquences clients qui contiennent les séquences candidates dont les supports doivent être déterminés. Par contre si un ou plusieurs de ces paramètres sont définis, l'algorithme doit s'assurer que les séquences respectent les contraintes imposées par les paramètres avant d'en augmenter le support. [51]

Exemple :

Candidats initiaux: toutes les séquences singleton < a>, <b >, <c >, <d >, <e > ,<f> ,<g >,<h> Scan base de données une fois, comptez le soutien pour les candidats[52].

Tableau – base de donnée séquentiel.

Seq.id	Séquence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Min_sup=2

Tableau – calculer le support

Cande	Sup
A	3
B	5
C	4
D	3
E	3
F	2
G	1
h	1

Tableau 3.10 : représentation des données horizontalement. **[52]**

	<a>	<b>	<c>	<d>	<e>	<f>
A	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
B	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
C	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
D	<da>	<db>	<dc>	<dd>	<de>	<df>
E	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
F	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

Tableau 3.11 - Calculer les candidats. [52]

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<ab>	<ac>	<ad>	<ae>	<af>
<b>			<bc>	<bd>	<be>	<bf>
<c>				<cd>	<ce>	<cf>
<d>					<de>	<df>
<e>						<ef>
<f>						

Calculer les candidats pour cet exemple : 8*8+8*7/2=92 candidates avec confi= 47 ,57% candidats.

Limites de l'algorithme GSP

- Une grande quantité de candidats peut être générée dans les grandes bases de donnée.
- Une grande quantité de balayages de la base de données est requise. Étant donné que la longueur de chaque séquence candidate grandit d'un item à chaque balayage, l'identification d'une 15-séquence requiert 15 balayages de la base de données.
- Les méthodes basées sur l'algorithme Apriori, comme c'est le cas pour le GSP, ont de la difficulté à découvrir de longues séquences. Ceci vient du fait que les longues séquences sont formées à partir d'un nombre important de séquences plus courtes et le nombre de candidats générés varie de manière exponentielle avec la longueur de ces derniers [47].

3.3.3.1.2 Algorithm PSP (PrefixTree for SequentialPattern)

Il utilise le même principe d'extraction que l'algorithme GSP. Cependant, il peut améliorer ses performances et établir différentes structures hiérarchiques pour représenter les objets candidats, et permettre la prise en compte des changements de temporalité entre les événements [53], l'arbre de hachage utilisé par l'algorithme GSP présente un défaut. En effet lors de la recherche des feuilles susceptibles de contenir des candidats inclus dans la séquence analysée, la structure utilisée ne tient pas compte des changements de date entre les items de la séquence qui servent à la navigation.

Par exemple, avec la séquence < (10 30) (20 40)>, L'algorithme va atteindre la feuille du sommet 30 (fils de 10), alors que cette feuille peut contenir deux types de candidats [54] :

- ceux qui commencent par < (10 30).
- et ceux qui commencent par < (10 30) ... de l'autre.

Ensuite, l'objectif est de construire une arborescence de préfixes pour gérer les candidats.

Le principe de cette structure est de décomposer la séquence candidate en fonction du préfixe de la séquence candidate.De plus, pour prendre en compte le changement d'itemset, l'arbre est doté de deux types de branches. Le premier type, entre deux items, signifie que les items sont dans le même itemset alors que le second signifie qu'il y a un changement d'itemset entre ces deux items. Ainsi, l'arbre de préfixes ne stocke plus les candidats dans les feuilles mais permet de retrouver les candidats de la façon suivante : tout chemin de la racine à une feuille représente un candidat et tout candidat est représenté par un chemin de la racine à une feuille. Cette nouvelle structure de données permet ainsi une nette amélioration des performances lors de l'extraction de motifs séquentiels. [53]

Limites de l'algorithme PSP

Le principal problème avec PSP est le nombre de passes dans la base de données. Pour une séquence de longueur k, effectuez k passes.Perte de temps lors du calcul du nombre de clients prenant en charge une séquence donnée.[47]

3.3.3.2 Méthode verticale

3.3.3.2.1 L'algorithme SPADE

SPADE (pour le terme anglais Sequential Pattern Discoveryusing Equivalence classes) ZAKI (2001), elle est le premier algorithme à proposer et utiliser une représentation verticale de la base de données pour extraire de manière plus efficace des motifs séquentiels [49]. Afin d'optimiser l'espace de mémoire utilisée, SPADE décompose l'espace de recherche dans des classes d'équivalences qui peuvent être traités indépendamment en mémoire [44], ces classes d'équivalences sont définies à partir d'une relation d'équivalence sur le préfixe. Ainsi, deux séquences de taille k sont dans la même classe d'équivalence si elles ont un préfixe commun de taille k – 1. [49]

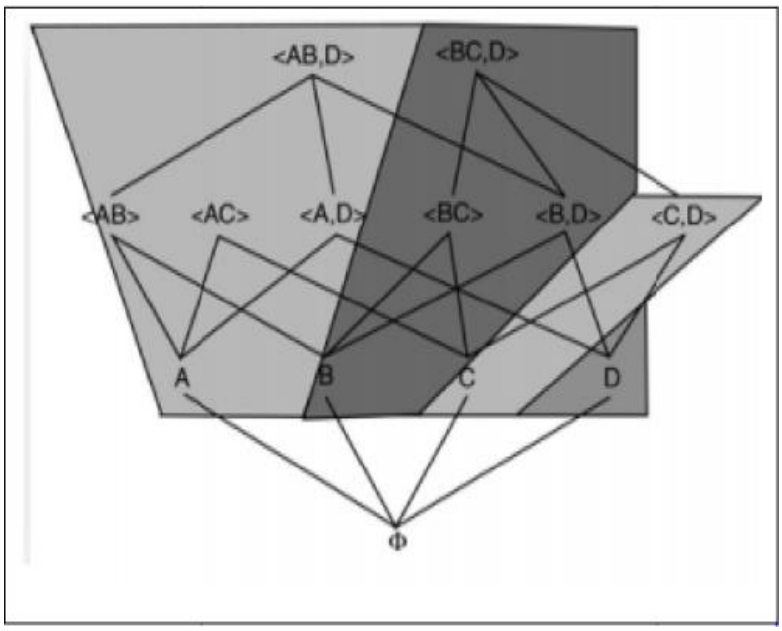


Figure 3.10: **classes d'équivalences générer par l'algorithme SPADE.[49]**

Dans les bases de données verticales, la base de données devient un ensemble de nuplets de la forme <itemset : (sequence_ID ,event_ID)>. L'ensemble des paires ID d'un itemset forme l'identifiant de la liste (ID_list) de l'itemset. Pour découvrir les K-séquences (séquences contenant k items), l'algorithme SPADE joint les ID_lists de deux éléments de l'ensemble des k − 1 séquences fréquentes.

SPADE a besoin de seulement deux balayages de la base de donné afin d'extraire les motifs séquentiels.

Le premier balayage vise à trouver les items fréquents F1. le deuxième à trouver les séquences fréquentes de longueur 2 F2 [48], Le calcul de F2 (les fréquents de taille 2) par SPADE, passe par une inversion de la base, qui la transforme d'un format vertical vers un format horizontal. Les auteurs considèrent que cette opération peut être simplifiée si la base peut-être chargée en mémoire vive [51].

Lors de la génération des séquences candidates, les jointures se font de la façon suivante :

- Élément contre élément : AB jointe avec AD en ABD.
- Élément contre séquence : AB jointe avec A → C en AB → C.

- Séquence contre séquence : A → B jointe avec A → C en A → BC , A → B→ C et A→ C

La procédure s'arrête quand aucune séquence fréquente ne peut être générée ou qu'aucune séquence ne peut être jointe. L'utilisation de bases de données verticales permet d'améliorer l'étape de vérification des séquences candidates.

```
Donnees en entrée : DB, minsup

nr. Produit la liste des sequencesfrequentes

Lire DB pour calculer F1, et les IdList(S) pour tout S
∈ F1

1.  K = 1
2.  WhileFk≠∅
3.  {
4.      Fk+1 = ∅
5.  forall P ∈Fk , Q ∈Fkif prefixe(P) = prefixe(Q)
6.      {
7.          Z = fusion(P,Q)
8.  ConstructIdList(Z)
9.          If freq(Z) >minsup Fk+1 = Fk+1 U {Z}
10.     }
11. k = k+1
12. }
13. Return F1 U F2 .... UFk.
```

Figure 3.11- Le pseudo code de l'algorithme SPADE.[51]

Limite de SPADE

La nécessité d'une très grande mémoire pour transformer et puis après stocker toute la base de données.

3.4 Conclusion

Dans ce chapitre, nous avons présenté certaines techniques d'extraction de motifs fréquent et séquentielle dans base des séquences. Une étude comparative des différents algorithmes dans ce domaine a été réalisés nous avons montré que l'application de ces algorithmes sur une même base de données avec les mêmes seuils de support donne les mêmes motifs. Par conséquent, la comparaison doit principalement porter sur les performances, c'est-à-dire le temps de réponse de ces algorithmes et l'espace de mémoire nécessaire aux tâches d'extraction.

4 Chapitre 4 : Solution proposée

4.1 Introduction

Dans ce chapitre, nous proposons trois (03) variantes de l'algorithme item2vec. La premièrevariante est item2vec_1 dans lequel nous avons utilisé le word2vec avec l'architecture de skip gram et la fonction softmax comme fonction objectif sur notre dataset qui contient des transactions pour prédire les items fréquents. La deuxième variante est l'algorithme item2vec_2 qui utilise une autre fonction objective qui s'appelle « NegativeSampling » qui peut améliorer les résultats et réduit le coût.

Le troisième algorithme utilise une fonction basée sur une probabilité conjointe de mapper des éléments à un faible espace vectoriel dimensionnel qui peut capturer la relation entre les éléments bien cooccurrence, et enfin on va comparer les résultats des solutions.

4.2 Algorithme de item2vec_1

Dans l'algorithme item2vec_1, l'entrée est l'itemcentral et les prédictions sont les transactions. Considérons une transaction T, si T(i) est l'entrée (item central), alors T(i-2), T(i-1), T(i + 1) et T(i + 2) sont les items de contexte, si la *taille de la fenêtre glissante* est de 2.

Définissonsquelques variables:

V: *Nombre de items uniques dans notre dataset (Vocabulaire)*

X: *Couche d'entrée (encodage One-Hot-Vector de notre entrée).*

N: *Nombre de neurones dans la couche cachée du réseau de neurons.*

W: *Poids entre la couche d'entrée et la couche cachée.*

W': *Poids entre la couche cachée et la couche de sortie.*

Y: *Une couche de sortie softmax ayant des probabilités de chaque item de notre vocabulaire.*

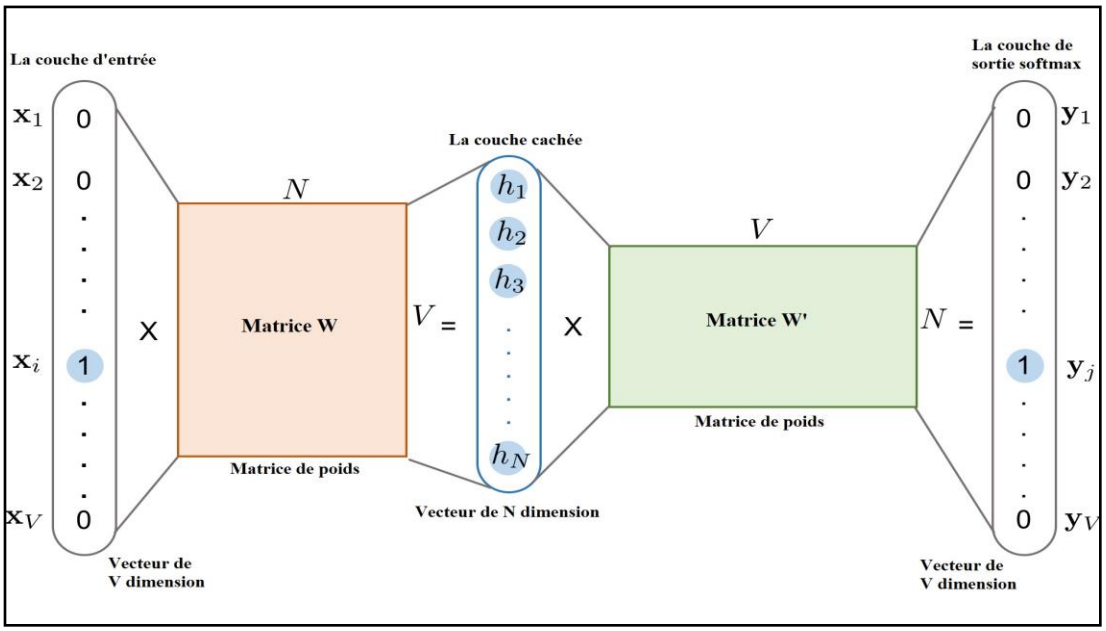


Figure 4.1 : Architecture de item2vec_1.

Notre architecture de réseau neuronal est définie dans les chapitres précédents, faisons maintenant quelques calculs pour dériver les équations nécessaires dans notre algorithme.

4.2.1 Pseudo code

Algorithme item2vec_1 : pseudo code de la première solution

1. **Entré :** ensemble de transactions
2. **Sortie** : ensemble d'itemsets fréquents
3. **Debut**
4. Préparation des données ; // fait le Vocabpour crée (one hot vector).
5. hyperparamètre : taux d'apprentissage, epochs, window size.
6. Générer des données d'entraînement ; // construire one hot vecteur pour nos items.
7. entrainement du modèle ; // Passer les items encodés à travers le feedforward,ajuster les poids à l'aide de la rétropropagation et calculer la perte.
8. Inférence ; // Obtenir un vecteur de l'item et trouvez des items similaires.
9. **Fin.**

4.2.1.1 hyperparamètres

Avant de nous lancer dans l'implémentation réelle, définissons certains des hyperparamètres dont nous aurons besoin plus tard**.**

-[window_size] : les mots de contexte sont des mots voisins du mot cible (target).en suppose que window_size égale à 2 ce qui signifie que les mots situés à 2 à gauche et à droite des mots cibles

-[epoche] : c'est le nombre d'entrainement a chaque epoche , nous parcourons tous les échantillons d'apprentissage.

-[taux d'apprentissage] : Le taux d'apprentissage contrôle la quantité d'ajustement apporté aux poids par rapport au gradient de perte.

4.2.1.2 Entrainement du modèle

4.2.1.2.1 Feedforward

Pour obtenir la matrice de couche cachée **h** (de taille N x 1) nous multiplier le One-hot-vector de l'item central (noté **x**) avec la première matrice de poids **W :**

$$h = W^T . x \qquad (1)$$

(Vx1) (NxV) (Vx1)

Maintenant, nous multiplions le vecteur de couche caché **h** par la deuxième matrice de poids **W'** pour obtenir une nouvelle matrice **u.**

$$u = W'^T . h \qquad (2)$$

(Vx1) (VxN) (Nx1)

Notons que nous devons appliquer une fonction de probabilité conditionnelle, dite softmax, à la couche u pour obtenir notre couche de sortie y.

Soit $\mathbf{u}_j$ le $j^{ème}$ neurone de la couche **u.**
Soit $\mathbf{w}_j$ le $j^{ème}$ mot de notre vocabulaire où j est un indice quelconque.
Soit $\mathbf{V}_{\mathbf{w}_j}$ la $j^{ème}$ colonne de la matrice **W '** (colonne correspondant à un **mot** $\mathbf{w}_j$).

$$u_j = V_{wij}^T . h \qquad (3)$$

(1x1) (1xN) (Nx1)

y=softmax(u).

y_j =softmax(u_j).

y_j désigne la probabilité que w_j soit un mot de contexte.

$$P(w_j|w_i) = y_j = \frac{e^{u_j}}{\sum_{j'=1}^{u} e^{u_{j'}}} \qquad (4)$$

P (w_j | w_i) est la probabilité que w_j soit un item de contexte, étant donné que w_i est l'item d'entrée.

Exemple :

Dans le dataset qui contiens les items suivant :

[1 2 3 4 5 6 7 8 9] on suppose que l'item cible (target) c'est 1 et le contexte sont 2 et 3 avec window_size égale à 2. Alors la représentation de feedforwad et comme suit :

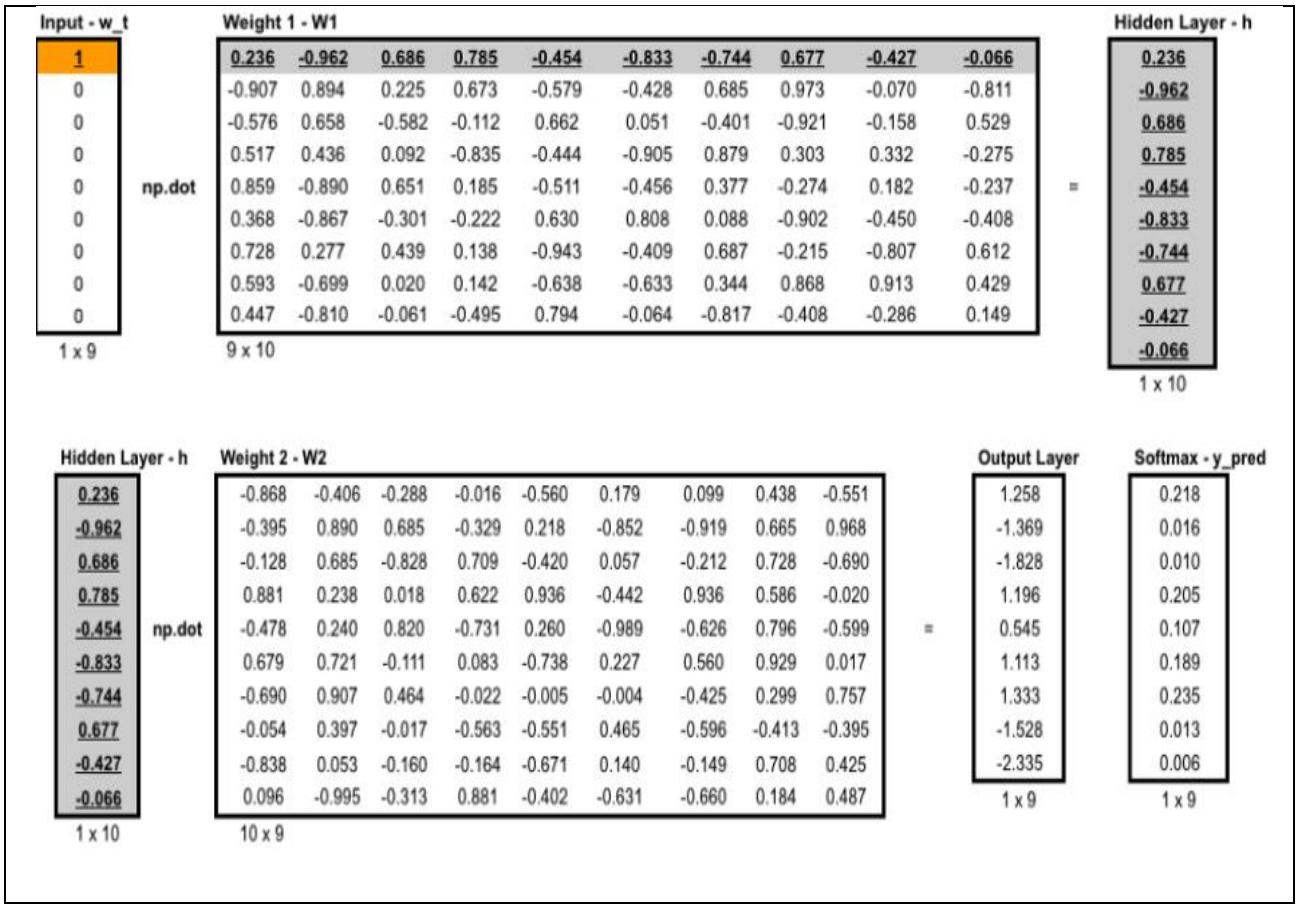


Figure 4.2: présentation de calculs fait au niveau de la couche cachée et le résultat de couche de sortie.

Maintenant on va expliquer comment la fonction softmax marche, on prend le vecteur précèdent (output layer) comme exemple :

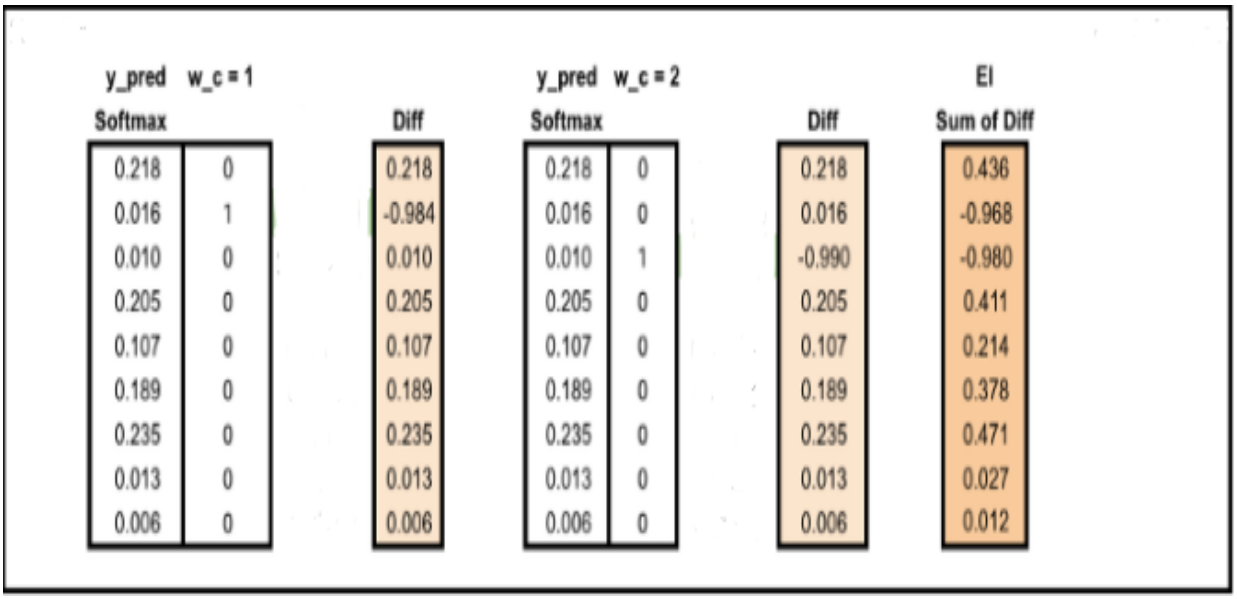
1,258		$e^{1.258}$
-1,369		$e^{-1,369}$
-1,828		$e^{-1,828}$
1,196	=	$e^{1,196}$
0,545		$e^{0,545}$
1,113		$e^{1,113}$
1.333		$e^{1,333}$
-1.528		$e^{-1,528}$
-2,335		$e^{-2,335}$

Maintenant on prend l'item de l'indice numéro 1 :

$y_i = (e^{1.258} \div (e^{1.258} + e^{-1,369} + e^{-1,828} + e^{1,196} + e^{0,545} + e^{1,113} + e^{1,333} + e^{-1,528} + e^{-2,335})) = 0,218.$

4.2.1.2.2 TrainErreurBack propagation et perte

- **Erreur :** avec y_pred , h et u on va calculer l'erreur pour cet ensemble particulier avec target et contextword ,ceci est fait en additionnant la différence entre y_pred et chacun des mots de contexte dans w_c.



y_pred Softmax	w_c = 1	Diff
0.218	0	0.218
0.016	1	-0.984
0.010	0	0.010
0.205	0	0.205
0.107	0	0.107
0.189	0	0.189
0.235	0	0.235
0.013	0	0.013
0.006	0	0.006

y_pred Softmax	w_c = 2	Diff	EI Sum of Diff
0.218	0	0.218	0.436
0.016	0	0.016	-0.968
0.010	1	-0.990	-0.980
0.205	0	0.205	0.411
0.107	0	0.107	0.214
0.189	0	0.189	0.378
0.235	0	0.235	0.471
0.013	0	0.013	0.027
0.006	0	0.006	0.012

Figure 4.3: calculer l’erreur.

- **Back propagation :** maintenant nous utilisons la fonction back probagation pour calculer la quantité d'ajustement, nous devons modifier les poids à l'aide de la fonction back_prop en passant l'erreur EI, la couche cachée h et le vecteur pour le targetw_t.

Pour mettre à jour les poids, nous multiplions les poids à ajuster (dl_dw1 et dl_dw2) par le taux d'apprentissage puis nous le soustrayons des poids actuels (w1 et w2).

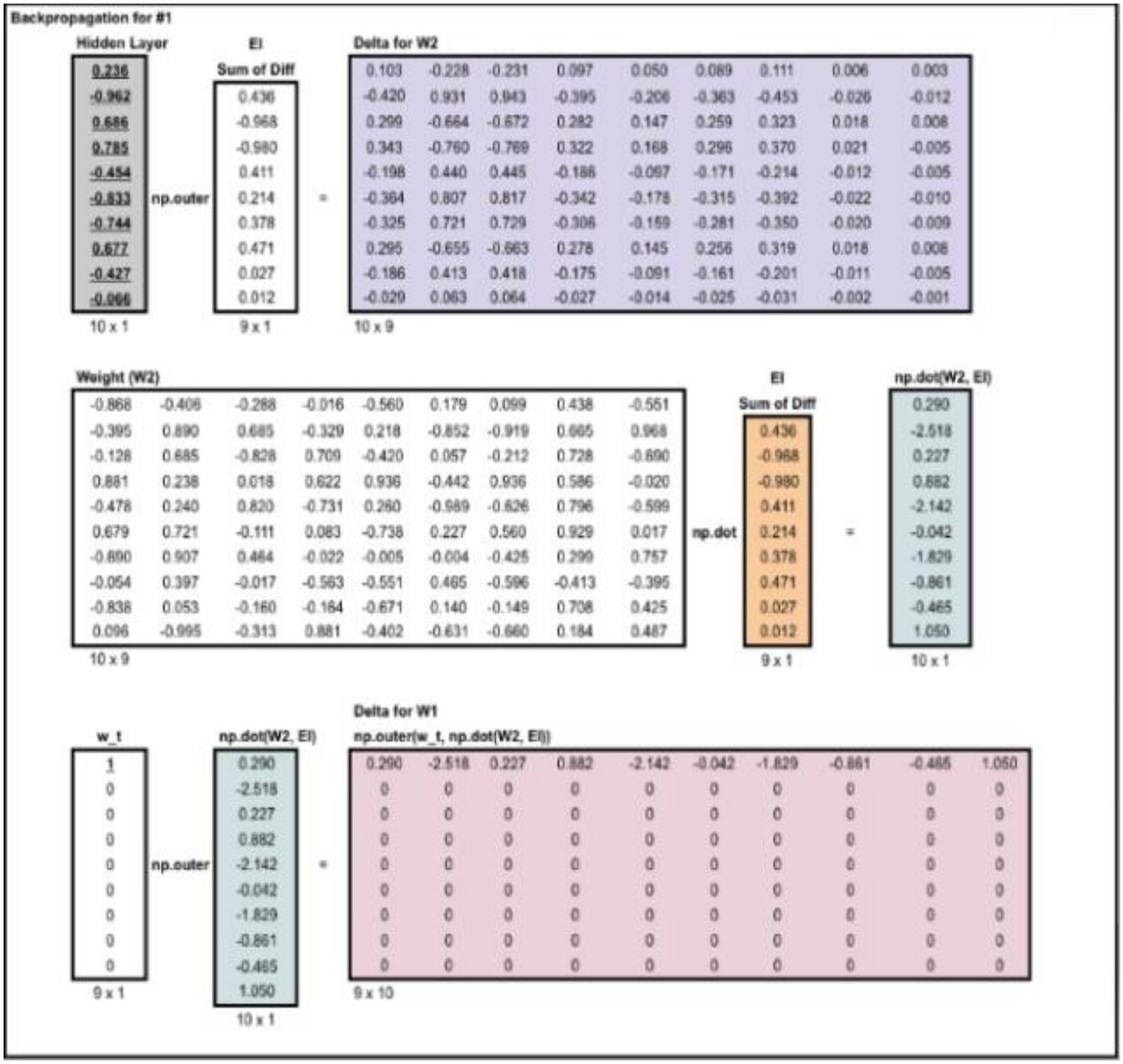


Figure 4.4: calcule delta pour w1 et w2.

- **Perte** : Enfin, nous calculons la perte globale après avoir terminé chaque échantillon d'apprentissage en fonction de la fonction de perte.

  Notez que la fonction de perte comprend 2 parties. La première partie est le négatif de la somme de tous les éléments de la couche de sortie (avant softmax). La deuxième partie prend le nombre d'items de contexte et le multiplie le log de la somme pour tous les items dans la couche de sortie.(Voir la fonction 6)

$$E = -log\left\{\prod_{c=1}^{C}\frac{e^{u_{jc*}}}{\sum_{j'=1}^{u} e^{u_{j'}}}\right\}$$, Pour que E est la Fonction de perte. (5)

$$= -\sum_{c=1}^{C} u_{jc} * + C.\log\left(\sum_{j'=1}^{u} e^{u_{j'}}\right)$$ (6

4.3 Algorithme Item2vec_2

Dans cette solution, Nous avons Travaille avec le même pseudocode de l'algorithme Item2vec_1 mais la fonction de la sortie est différente, nous avons utilisé word2vec avec échantillonnage négatif (NegativeSampling) au lieu softmaxpour réduire les coûts de calcul et pour améliorer les résultats.

La fonction de coût pour l'algorithme item2vec_1 et l'algorithme item2vec_1 ressemble à ceci:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log\, p\left(w_{t+j}\middle|w_t;\ \theta\right) \ldots\ldots (7)$$

Notez que T c'est le nombre de tous les vocabulaires. C'est équivalent à V. En d'autres termes, T = V.
La distribution de probabilité p(w_t+j|w_t) dans SG est calculée pour tous les V vocabulaires du corpus avec :

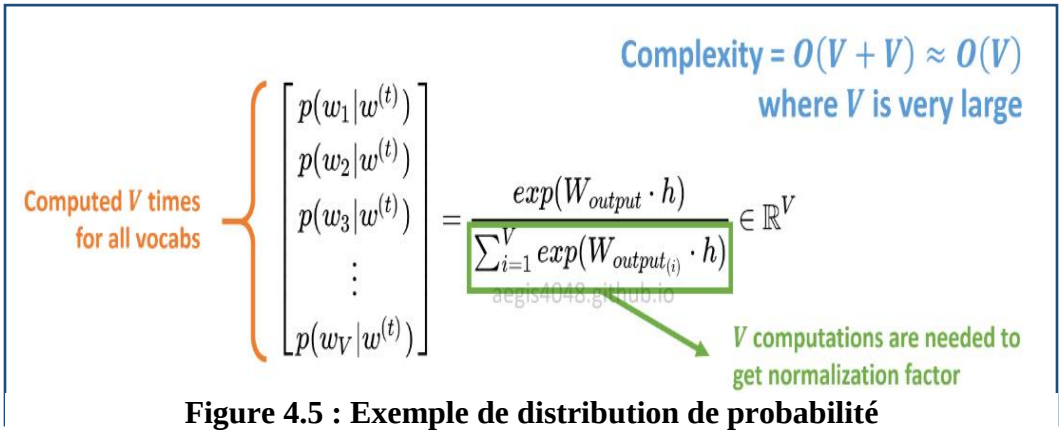


Figure 4.5 : Exemple de distribution de probabilité

V peut facilement dépasser des dizaines de milliers lors de la formation du modèle Skip-Gram. La probabilité doit être calculée en V temps, ce qui la rend coûteuse en calcul. De plus, le facteur de normalisation dans le dénominateur nécessite des V calculs supplémentaires.

D'autre part, la distribution de probabilité dans SGNS est calculée avec:

$$\begin{bmatrix} p(D=1\,|w,Cpos) \\ p(D=1\,|w,Cneg,1) \\ p(D=1\,|w,Cneg,2) \\ p(D=1\,|w,Cneg,3) \\ \vdots \\ p(D=1\,|w,Cneg,k) \end{bmatrix} = \frac{1}{1+\exp(-(\{Cpos\}\cup W\,neg).h)} \quad \ldots\ldots\ldots (8)$$

Cposest un vecteur de mot pour un mot positif et Wnegest un vecteur de mot pour tous K les échantillons négatifs dans la matrice de poids de sortie. Avec SGNS, la probabilité doit être calculée uniquement K + 1 fois, où K est typiquement entre 5 ~ 20. De plus, aucune itération supplémentaire n'est nécessaire pour calculer le facteur de normalisation dans le dénominateur.

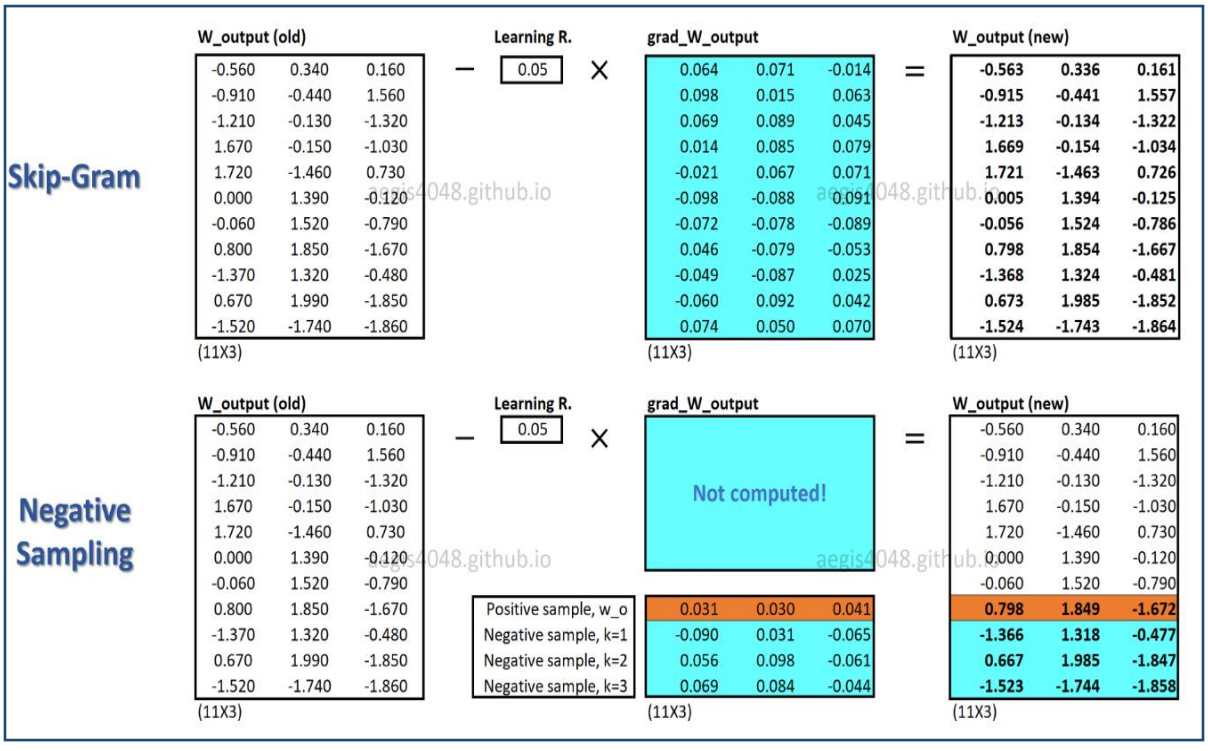


Figure 4.6 : La différence d'entrainement dans les deux solutions proposées.**[55]**

Avec SGNS, seule une fraction des poids est mise à jour pour chaque échantillon d'entraînement, tandis que SG met à jour tous les millions de poids pour chaque échantillon d'entraînement.

Le NegativeSampling fait ça en transformant la tâche multi-classification en tâche de classification binaire.

Avec SGNS, les vecteurs de mots ne sont plus appris en prédisant les mots de contexte d'un mot central. Il apprend à différencier les mots contextuels réels (positifs) des mots tirés au hasard (négatifs) de la distribution du bruit.

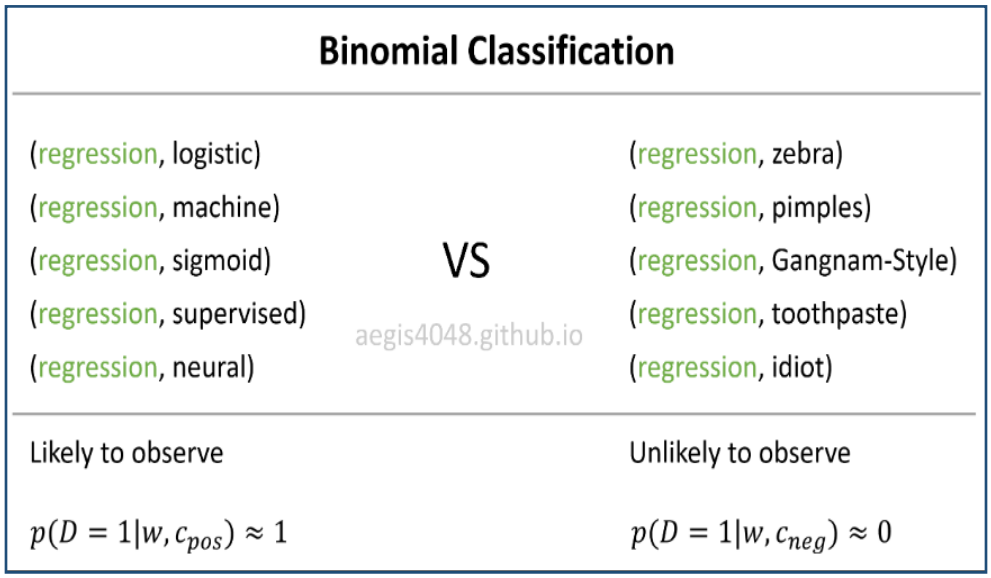


Figure 4.7 : Exemple de distribution positifs et distribution negatifs.

Généralement, vous n'observez pas le mot « regression » avec des mots aléatoires comme « Zebra », ou « pimples ». L'idée est que si le modèle peut distinguer les paires probables (positives) des paires improbables (négatives), de bons vecteurs de mots seront appris.

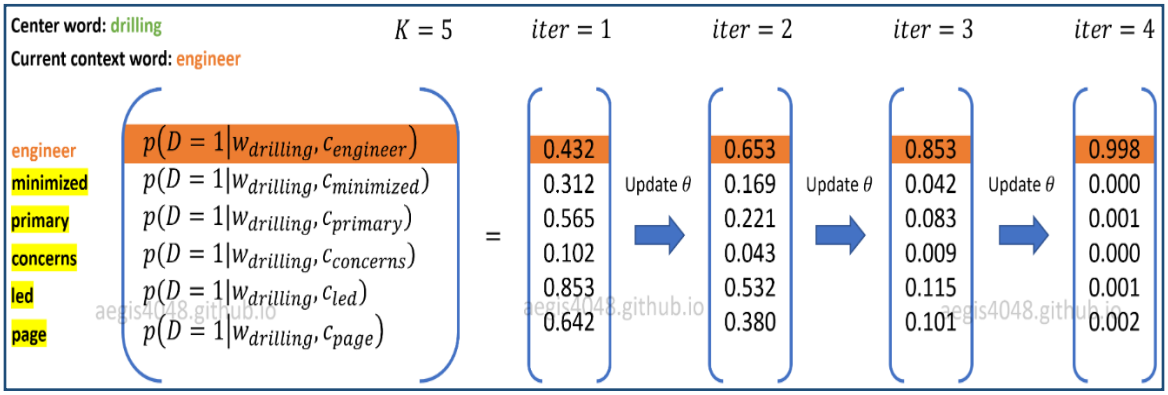


Figure 4.8 : Exemple de distribution positifs et distribution négatifs.

Dans la figure ci-dessus, la paire mot-contexte positive actuelle est (drilling engineer). K=5 échantillons négatifs sont tirés au hasard de la distribution du bruit : minimized, primary, concerns, led, page.
Au fur et à mesure que le modèle parcourt les échantillons d'apprentissage, les poids sont optimisés de sorte que la probabilité d'une paire positive s'affiche p(D=1|w,c_pos)≈1et la probabilité de paires négatives s'affiche p(D=1|w,c_neg)≈0.

4.4 Algorithme item2vec_3

4.4.1 Une fonction d'apprentissage des vecteurs d'éléments d'incorporation

Notre objectif est d'apprendre des vecteurs continus d-dimensionnelspour chaque élément. Soit V = $\{v_1, v_2, v_3, \ldots\ldots, v_m\}$ correspondant à chaque élément, Le d est un hyper paramètre qui est une dimension de l'espace vectoriel de l'élément.

Nous définissons une fonction pour intégrer chaque élément dans le vecteur. La fonction est basée sur word2vec. L'idée principale pour apprendre l'incorporation du vecteur d'élément dans la même transaction est de maximiser une fonction objective. Nous définissons et maximisons la fonction objectif en utilisant une probabilité logarithmique de tous les éléments dans les mêmes transactions:

$$\text{maximize } \frac{1}{n}\sum_{k=1}^{n} \log P(T_k)$$

La probabilité conjointe$P(T_k)$définit comme la fonction softmax:

$$P(T_{target}) = \frac{\exp(\sum_{ia,ib \in Ttarget} v_a . v_b)}{\sum_{k=0}^{n}(\sum_{ia,ib \in Tk} v_a . v_b)}$$

Où Ttarget est la transaction cible qui contient le vecteur cible, et Va et Vb sont une représentation vectorielle d'a et b dans la même transaction, et n est le nombre de transactions. Nous décrivons la probabilité conjointe en utilisant la somme du produit scalaire de chaque élément en fonction exponentielle.

Si Ic se produit dans un grand nombre de transactions, Ic devrait être approximativement à la même distance de tous les éléments qui se produisent avec Ic dans les mêmes transactions. Nous formons la représentation vectorielle en utilisant un gradient de vitesse décent en ajoutant

un gradient du vecteur cible au vecteur cible. La probabilité logarithmique est développée comme suit:

$$\sum_{ia,ib \in Ttarget} v_a . v_b - \log \sum_{k=0}^{n} \exp(\sum_{ia,ib \in Tk} v_a . v_b)$$

Nous l'avons utilisé comme fonction de log-likelihood qui devrait être maximisée.

4.5 Conclusion

Dans ce chapitre nous avons présenté nos trois algorithmes proposés comme des solutions pour nos problèmes dans l'extraction de motifs fréquents.

5 Chapitre 5 : Tests et évaluations

5.1 Introduction

Le but de ce travail s'agit de proposer des méthodes qui permettent d'extraire et d'analyser des items fréquents.

Dans ce chapitre nous commençons par expliquer l'environnement de développement ainsi que les outils permettant l'implémentation de notre travail : le langage Python, l'environnement Pycharm, des bibliothèques numpy, pandas, mathplotlib… etc.

Nous passons, ensuite, à la description des différentes étapes du processus d'évaluation illustrant chaque étape par des résultats expérimentaux.

L'étude de la littérature menée précédemment nous a permis de comprendre l'étendu du sujet et les différents aspects à améliorer au niveau de notre solution.

En effet, les méthodes utilisées précédemment se basent essentiellement sur les word2vec, une méthode principalement utilisé pour l'études des mots dans les textes. Cette méthode bien qu'elle soit bien adaptée pour son domaine, elle présente plusieurs limites.

Nous présentant dans ce chapitre notre solution proposé et son implémentation pour répondre à nos objectifs.

5.2 Environment de development

5.2.1 Environment materiel

Nous avonsutilisé :

-Un ordinateur portable HP avec les caractéristiques suivantes:

- 4 Go RAM
- 4096 MBytes DDR3
- Intel ® core (TM) i7-4600U CPU 2,7 GHZ
- System d'exploitation: windows 10 de 64 bits.

5.2.2 Environment logiciel

5.2.2.1 Python

Python est un excellent langage de programmation orienté objet, interprété et interactif. Il est souvent comparé (favorablement bien sûr) à Lisp, Tcl, Perl, Ruby, C #, Visual Basic, Visual Fox Pro, Schème ou Java ... etc.

Python combine un pouvoir remarquable avec une syntaxe très claire. Il comporte des modules, des classes, des exceptions, des types de données dynamiques de très haut niveau et le typage dynamique. Il existe des interfaces vers de nombreux appels systèmes et bibliothèques, ainsi que vers différents systèmes de fenêtrage.

Les nouveaux modules intégrés sont faciles à écrire en C ou C ++ (ou dans d'autres langages, selon l'implémentation choisie). Python est également utilisable comme langage d'extension pour les applications écrites dans d'autres langages nécessitant des interfaces de script ou d'automatisation facile à utiliser. [56]

5.2.2.2 Pycharm

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

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

Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux.[57]

5.2.2.3 Numpy

Est une bibliothèque permettant d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres, des fonctions sophistiquées (diffusion), on peut aussi l'intégrer le code C / C ++ et Fortran. [58]

5.2.2.4 Matplotlib

Est une bibliothèque de traçage pour le langage de programmation Python et son extension mathématique numérique NumPy. Il fournit une API orientée objet permettant d'incorporer des graphiques dans des applications à l'aide de kits d'outils d'interface graphique à usage général tels que Tkinter, wxPython, Qt ou GTK +. [59]

5.2.2.5 Gensim

Est une bibliothèque Python pour la modélisation de sujets, l'indexation de documents et la recherche de similitudes avec un grand corpus. Le public cible est la communauté du traitement du langage naturel (PNL) et de la recherche d'informations (IR). [60]

5.2.2.6 Pytorch

PyTorch est une bibliothèque d'apprentissage automatique open source basée sur la bibliothèque Torch, utilisée pour des applications telles que la vision par ordinateur et le traitement du langage naturel, principalement développée par le laboratoire de recherche AI de Facebook (FAIR). Il s'agit d'un logiciel gratuit et open-source publié sous la licence BSD modifiée. Bien que l'interface Python soit plus raffinée et soit l'objectif principal du développement, PyTorch dispose également d'une interface C ++ [61].

5.3 Implémentation

5.3.1 Algorithme item2vec_1

5.3.1.1 La fonction du prétraitement

```
corpus = open("ToyData.txt").read()
def preprocessing(corpus):
    training_data = []
    Transactions = corpus.splitlines()
    #print(Transactions)
    for i in range(len(Transactions)):
        Transactions[i] = Transactions[i].strip()
        Transaction = Transactions[i].split()
        x = [item.strip(string.punctuation) for item in Transaction]
        training_data.append(x)
    return training_data

print(preprocessing(corpus))
```

Figure 5.1 : la fonction du prétraitement.

Apres l'importation de la base de donnée, La fonction preprocessing devise les lignes de la base de donnée et les mettre dans une liste sous forme des transactions, Puis elle va construire une liste qui contient ces transactions sous forme des items dans une deuxième liste.

5.3.1.2 La fonction d'entrainement

```
model = Word2Vec(preprocessing(corpus), size=100, window=5, min_count=1, workers=4)
model.wv.save_word2vec_format('train_gens.txt', binary=False)
```

Figure 5.2 : la fonction d'entrainement.

L'entraînement est commencé, de sorte que l'ensemble de données peut être un itérable, lisant les données d'entrée du disque, sans charger l'intégralité de corpus dans la RAM. Notez que les phrases itérables doivent être redémarrables, pour permettre à l'algorithme de diffuser plusieurs fois sur l'ensemble de données. La deuxième commande nous permettre de sauvegarder les résultats sous forme de model dans un fichier « txt ».

5.3.2 Algorithme item2vec_2

5.3.2.1 Lire la base de donnée

```
def thread_read_data_(self):

    f = open(self.log_filename)
    while True:
        if len(self.items) < 1000:
            items = f.readline()
            if not items:
                f = open(self.log_filename)
                items = f.readline()
            item_ids = items.strip().split(' ')
            self.items.append(item_ids)
```

Figure 5.3: **fonction pour lire la base de donnée**

Après l'importation de la base de données la fonction thread_read_data qui organise la base données avec rend chaque transaction dans une liste et pour différencié entre un item et un autre avec espace.

5.3.2.2 Générer du vocabulaire

```
def gen_vocab(self):

    assert self.log_filename != ''
    vocab_freq_dict = defaultdict(int)
    total_word_count = 0
    total_sent_count = 0
    with open(self.log_filename, encoding='utf-8') as f:
        for line in f:
            total_sent_count += 1
            item_ids = line.strip().split()
            if self.read_data_method == 'memory':
                self.sentences.append(item_ids)
            for item_id in item_ids:
                vocab_freq_dict[item_id] += 1
                total_word_count += 1
    vocab, word2id, id2word = {}, {}, {}
    index = 0
    for item_id, freq in vocab_freq_dict.items():
        if freq < self.min_count:
            continue
        vocab[item_id] = freq
        word2id[item_id] = index
        id2word[index] = item_id
        index += 1
    return vocab, word2id, id2word, total_word_count, total_sent_count
```

Figure 5.4: fonction pour générer les vocabs.

Après ce que nous avons fait les transactions dans une seule liste maintenant on va calculer pour chaque item ça fréquence et trouvé ça indexation Et mettre chacun d'eux dans sa propre liste.

5.3.2.3 Sauvegarder le modèle

```
model = KeyedVectors.load_word2vec_format(Path.cwd() / 'model_Negative.txt', binary=False)
```

Figure 5.5: La fonction d'enregistrer l'entrainement.

Cette fonction nous permettre d'enregistrer le modèle obtenu après l'entrainement de notre data base en utilisant l'algorithme item2vec_2.

5.3.3 Projection 2d

```
import pandas as pd
from sklearn.decomposition import PCA #Grab PCA functions
import matplotlib.pyplot as plt

df = pd.read_csv("NegativeSampling\word2vec\model_Negative.csv")
targets = df['7'].to_numpy()
print(targets)
pca = PCA(n_components=2)
result = pca.fit_transform(df)
fig, ax = plt.subplots()
ax.plot(result[:, 0], result[:, 1], 'o')
for label, x, y in zip(targets, result[:,0], result[:,1]):
    plt.annotate(label, xy=(x, y), xytext=(-2, 2), textcoords='offset points', ha='right', va='bottom')
ax.set_title('Items')
plt.show()
```

Ce code nous permettons de dessiner la projection 2d pour comparer entre les solutions par vecteurs.

5.4 Expérimentation

5.4.1 Description de la base de donnée (ToyData.txt)

Dans cette section, nous montrons quelques résultats expérimentaux pour confirmer si notre méthode fonctionne avec les données de transaction. Nous appliquons notre méthode à un jeu de données de jouet pour montrer comment cela fonctionne. Les données sur les jouets sont créées d'une façonconformément à la règle selon laquelle les éléments à nombre pair sont bien en cooccurrence avec les éléments à nombre pair, et les éléments à nombre impair sont bien en cooccurrence avec les éléments à nombre impair. L'élément {7} se produit toutes les transactions. Par conséquent, l'élément {7} est un élément indépendant dans l'ensemble de données.

Tableau 5.1 -La base de données ToyData.

Numéro de transaction	Transaction
01	1 2 3 5 7
02	1 3 7
03	3 4 7
04	1 3 5 7
05	2 4 7
06	2 4 6 7
07	1 2 3 7
08	4 5 6 7
09	4 6 7
10	2 6 7
11	3 5 7
12	1 3 7
13	1 3 5 7
14	2 4 7
15	2 4 6 7
16	1 2 3 4 5 7
17	1 3 5 7
18	2 4 6 7
19	1 3 5 7
20	4 6 7

5.4.2 Evaluation des résultats

La dimension des vecteurs est de deux pour confirmer le comportement des vecteurs en utilisant plot. Les résultats sont présentés dans la figure 5.6 (pour item2vec_1) et la figure 5.7 (pour items2vec_2). En les comparant, nous pouvons constater que les emplacements des éléments de nombre impair (1,3,5) se rapprochent dans la figure 5.7 que dans la figure 5.6. Comme pour les éléments de nombre pair (2,4,6). Dans la figure 5.7, l'élément {7} est presque à la même distance de chaque cluster. L'élément {7} est un élément indépendant. Par conséquent, l'élément {7} est approximativement à la même distance de chaque élément.

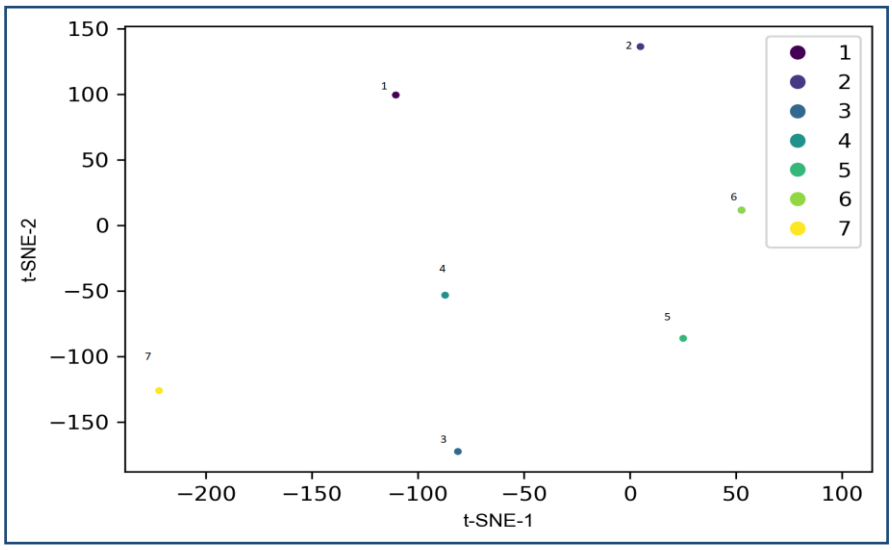


Figure 5.6 : Projection 2d du modèle extrait en utilisant item2vec_1.

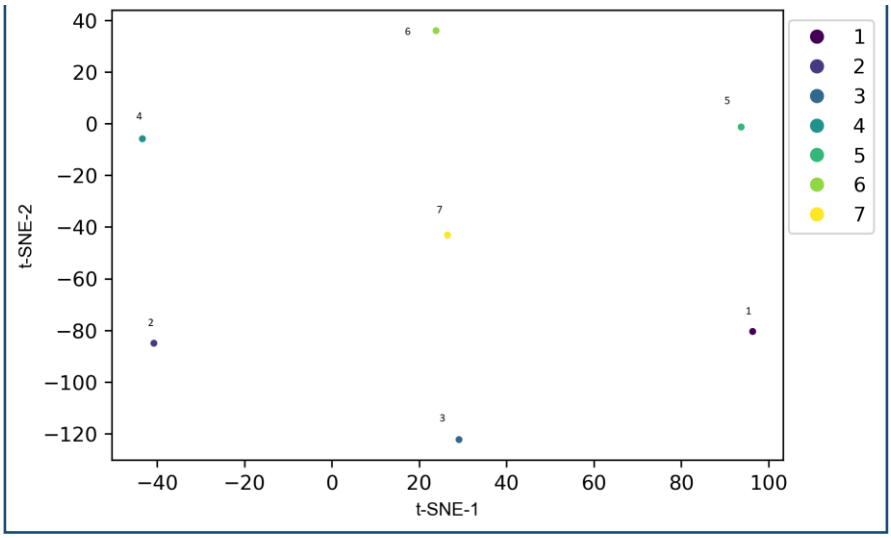


Figure 5.7 : Projection 2d du modèle extrait en utilisant item2vec_2.

Le point sur lequel nous devrions discuter est bien les performances du modèle non pas basé sur la distance des vecteurs mais celui base sur la similarité cosinus. Nous pouvons constater que les éléments de nombre impair (1, 3,5) ont une similarité cosinus proches de 1 et donc pour les éléments de nombre pair (2, 4,6). De plus, la similarité entre un élément de nombre impair et un élément de nombre pair ont tendance à être petites ou négatives. Cela suggère que la méthode a également la capacité de séparer les éléments qui n'apparaissent pas dans les mêmes transactions.

Tableau 5.2 : Similarités cosine et la distance réalisée par item2vec_1, En bas à gauche la distance. En haut à droite, les similarités.

Items	1	2	3	4	5	6	7
1		-0.18	0.13	0.2	0.06	-0.07	-0.06
2	1.18		0.08	-0.13	0.15	-0.07	0.09
3	0.86	0.92		0.08	0.26	0.15	-0.04
4	0.8	1.12	0.92		0.12	0.07	-0.06
5	0.94	0.85	0.74	0.88		0.004	0.06
6	1.07	1.07	0.85	0.93	0.99		-0.08
7	1.06	0.91	1.04	1.06	0.93	1.09	

Tableau 5.3 : Similarités cosine et la distance réalisée par item2vec_2, En bas à gauche la distance. En haut à droite, les similarités.

Items	1	2	3	4	5	6	7
1		-0.13	-0.01	-0.08	0.03	0.04	-0.03
2	1.13		-0.1	-0.03	-0.03	-0.09	0.14
3	1.01	1.1		-0.1	0.02	-0.17	-0.16
4	1.07	1.03	1.1		-0.02	0.15	0.01
5	0.97	1.03	0.98	1.02		-0.03	-0.04
6	0.96	0.91	1.17	0.84	1.03		-0.11
7	1.03	0.86	1.16	0.99	1.04	1.1	

Selon ces deux tableaux, nous remarquons que les distances obtenues entre les deux types de nombres (pairs et impaires) testés, donnes valeurs significatifs par rapport aux attentes expliqués au début. Le deuxième algorithme, montre de meilleurs performances quant aux distances retournés entres les nombres introduits.

Ces résultats montrent donc que cet algorithme sera retenu car il est plus efficace et valide les visualisations précédentes.

5.4.3 Temps d'exécutions

Tableau 5.4 - Temps de réponse des algorithmes.

	Nb Transation	Nb item	Apriori	FpGrowth	Item2vec_1	Item2vec_2
Chess	3196	75	1,45s	0,5s	0,33s	12s
Skin	245057	11	0,03s	1,7s	2,8s	40,03s
RecordLink	574913	29	0,32s	8,30	17,76s	57,31s
Connect	67557	129	Mémoire plante	9,02s	8,75s	2mn et 11s
Mushrom	8416	119	20,8ms	104ms	629ms	36,71s
Pumsb	49046	2113	Mémoire plante	6,19s	15,7s	18mn et 19s

- **Discussion :** D’après ce tableau, nous pouvons voir les performances obtenues avec les différents algorithmes en utilisant plusieurs jeux de testes. Nous remarquons qu’Apriori est le moins favorable, car il présente plusieurs obstacles au niveau de la mémoire vive qui s'épuise rapidement quand le nombre de transactions et des items augmente, et ce même avec un petit nombre d'item, par contre, les meilleurs résultats en termes de temps on était obtenu avec cet algorithme même. Les algorithmes fpgrowth et item2vec_1 donnent des résultats très similaire et assez comparable en ce qui concerne le temps d'exécution sauf pour les deux BD Mashroom et pumbs où fpgrowth et bien meilleur. Le dernier algo item2vec_2 quant à lui, prend trop de temps avant de finir comparer aux autres algorithmes, ce qui le rend moins efficace.

5.4.4 Consommation de la mémoire

Tableau 5.5 - Consommation de la mémoire des algorithmes

	FP-growth	Apriori	Item2vec_1	Item2vec_2
Chess	0.8GB	0.8GB	0.4GB	0.82GB
Skin	0.16GB	0.18GB	0.12GB	0.23GB
RecordLink	0.8GB	1.6GB	0.56GB	0.78GB
Connect	1.2GB	Débordement du Mémoire	0.32GB	0.92GB
Mushroom	0.1GB	0.4GB	0.08GB	0.41GB
Pumsb	1.4GB	Débordement du mémoire	0.44GB	1.05GB

- **Discussion :** D’après ce tableau qui est au-dessus nous pouvons voir les consommations mémoire obtenu après l’exécution des différents algorithmes sur plusieurs bases de données, Nous remarquons qu’ item2vec_1 est le plus favorable car il ne consomme pas beaucoup de mémoire vive même si le nombre de transactions ou des items augmente.

Les algorithmes FP-Growth et item2vec_2 donne des résultats similaires par contre l'algorithme Apriori se bloque dans les bases de données connect et pumsb.

5.4.5 Comparaison entre les résultats obtenus

Tableau 5.6 - Les motifs extrais par les trois algorithmes.

	Apriori	Item2vec_1	Item2vec_2
ToyData	7,3,4	7,3,2	7,3,4
Chess	29,40,52	29,10,55	29,40,11
Skin	11,2,3	11,3,2	11,3,2
RecordLink	11,24,27	11,19,3	11,28,27
Mushroom	34,85,86	34,28,29	34,85,90

Tableau 5.7 - Le pourcentage d'être les résultats justes

	Item2vec_1	Item2vec_2
ToyData	63%	100%
Chess	11%	80%
Skin	100%	100%
recordLink	30%	78%
Mushroom	30%	80%

- **Discussion** : pour comparer nos algorithmes on a calculé les motifs fréquents de chaque nos algorithmes et on a mis l'algorithme apriori comme référence, nous avons procédé de façon directe avec les mêmes bases de données. Les résultats démontrent qu'item2vec_2 est beaucoup plus précis dans la majorité des cas. Ce qui nous mène à la conclusion qu'il est plus performant que l'item2vec_1, de même que ses résultats sont similaires à ceux d'apriori**.**

5.5 Conclusion

Dans ce chapitre nous avons l'environnement matériel et l'environnements logiciel, les différents méthodes et algorithmes utilise pour implémenter nos solutions. Nous avons aussi introduit tous les résultats obtenus de chaque algorithme avec les analyses nécessaires pour comparer les performances des deux.

Conclusion générale et perspectives

Dans le cadre de ce travail, nous avons traité le problème d'extraction des motifs fréquents comme le problème de la mauvaise qualité des motifs extraits et Le temps de réponse très élevé. Au cœur de ce mémoire nous avons présenté les différentes techniques d'extraction des motifs fréquents et séquentiels. Pour mieux cerner la problématique posée nous avons explicité les techniques de word embedding tels que le word2vec ainsi ces algorithmes comme C-BOW et le SKIP-GRAM. En deuxième temps, nous somme passer aux algorithmes d'extraction des motifs fréquents, nous avons présenté les algorithmes utilisé dans ce domaine.

Ensuite, nous avons proposé trois variantes d'un algorithme qui extrait les motifs fréquents en se basant sur la technique de word2vec, ils ont appelé item2vec_1, item2vec_2, et item2vec_3.

Nous n'avons malheureusement pas pu aboutir à la réalisation de tous nos objectifs posés au début de ce travail, suite à plusieurs difficultés rencontrées lors du travail :

- Manque immense de matériel pour faire des calculs.
- Modèle de machine learning très compliqué à poser.
- Le temps accordé pour ce travail ne suffit pas pour un travail supplémentaire

Quoi que nous avons réalisé le principal objectif de notre travail. Cependant nous envisagerons quelques perspectives qui permettent l'amélioration et la conformité de notre travail notamment :

- Utiliser une large base de données dans l'entrainement pour améliorer les résultats.
- Classifier un item qui n'existe pas dans les transactions ou la base de données.
- Utilisation d'autres techniques de wordembeddings comme le GloVe.

Bibliographie

[01] https://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257

[02] B.Salima et Lionel Janin, Intelligence artificielle et travail,Rapport France, mars 2018

[03] A. M. Giancarlo Zaccone, Md. Rezaul Karim, Deep Learning with TensorFlow: Explore neural networks with Python. 2017.

[04] O.Mehdi et K.salim, Classification d'objets avec le Deep Learning, Univ de bouira, 2018.

[05] https://mrmint.fr/algorithme-k-means

[06] Samir amir, Une mesure de similarité entre phrases basée sur des noyaux sémantiques, janvier 2016.

[07] Moualek, D. Y. (2017). Deep Learning pour la classification des images (Thèse de doctorat), Université Abou Bah Bellcaid0 Tlemcen.

[08] G. Gelly, "Reseaux de neurones recurrents pour le traitement automatique de la parole To cite this version : HAL Id : tel-01615475 Thèse de doctorat de l ' Université Paris-Saclay préparée à l ' Université Paris-Sud par M .Gregory Gelly de la parole," 2017.

[09] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., &Khudanpur, S. (2010). Recurrent neural network based language model. In Eleventh annual conference of the international speech communication association.

[10] Graves, A., &Jaitly, N. (2014, January). Towards end-to-end speech recognition with recurrent neural networks. In International conference on machine learning (pp. 1764-1772).

[11] Karpathy, A., &Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3128-3137).

[12] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives.IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),38:1798–1828,2013

[13] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. The Journal of Machine Learning Research, 18(1), 6869-6898.

[14] Van den Oord, A., Dieleman, S., &Schrauwen, B. (2013). Deep content-based music recommendation. In Advances in neural information processing systems (pp. 2643-2651).

[15] Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning (pp. 160-167).

[16] Matsugu, M., Mori, K., Mitari, Y., &Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. Neural Networks, 16(5-6), 555-559.

[17] PARIZEAU Marc. Réseaux de neurones. 2004

[18] NAKAMOTO Pat. Neural networks and deep learning : deep learning explained to your granny a visual introduction for beginners who want to make their own deep learning neural network. 2017
[19] TOUZET Claude. Les réseaux de neurones artificiels, introduction au connexionnisme. 1992
[20] Brigitte Bigi, TALN Informatique, avril 2006.

[21] Natasha Latysheva, why do we use word embeddings in nlp?, 2019.

[22] Efficient Estimation of word representations in vector space, Tomas mikolov, Google inc, Mountain View, CA.

[23] https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[24] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, KorayKavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. JMLR, 12:2493–2537.

[25] Global Vectors for Word Representation, Jeffrey Pennington, Richard Socher, Christopher D. Manning. 2014

[26] https ://nlp.stanford.edu/projects/glove, 20 Avril 2019.

[27] A. Karpathy, the Unreasonable Effectiveness of Recurrent Neural Networks, 2015.

[28] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2014). word2vec.

[29] Rong, X. (2014). word2vec parameter learning explained. arXiv preprint arXiv :1411.2738.

[30] https://machinelearningmastery.com/what-are-word-embeddings/

[31] J. Zaki ,SPADE : An efficient algorithm for miningfrequentsequences Machine Learning Journal, Vol. 42(1−2), pp. 31−60, January 2001.

[32] C.seghier, B.kebadji, Réseau bayésien pour l'extraction des motifs fréquents à partir de données massives , Master , Saad dahlebblida ,2013.

[33] J.Cugliari, Fouille de Données, Master 2 IDS-Kharkiv, S2 2014-2015, Université Lumière-Lyon2

[34] Ph.Preux, Fouille de données, Notes de cours, Université de Lille 3.

[35] S.Tuffery, Cours de data mining, M2 Ingénierie économique et financière, Université Rennes 1, 2014

[36]R.Agrawal and R.Srikant. Fastalgorithms for mining association rules. pages 487–499, 1994.

[37]A.Salleb, Recherche de motifs fréquents pour l'extraction de règles d'association et de caractérisation, doctorat, Orléans, 2003

[38]International Journal of Computer Systems, ISSN-(2394-1065), Vol. 01, Issue: 03, December, 2014

[39]M.Benelhadj, entrepôt de données et fouille de données un modèle binaire et arborescent dans le processus de génération des règles d'association, doctorat, constantine

[40]S.Amina, Une approche basée agent pour la fouille de données, magister en informatique, batna, 2013

[41]Amir Al,Muhammad Zain Ami, A Briefly Explanation of Éclat Algorithm with Practical Implementation.

[42] O.KHEMIRI,Proposition d'une nouvelle approche d'extraction des motifs fermés fréquents,Mémoire de Mastère,TUNIS EL MANAR ,2018.

[43] J.Han and M.Kamber, Data mining concepts and techniques, 2nd edition , Diane Cerra San Francisco.

[44] B.Idiri, Méthodologie d'extraction de connaissances spatio-temporelles par fouille de données pour l'analyse de comportements à risques - Application à la surveillance maritime, doctorat, paris, 2013.

[45] A.Marascu, Extraction de motifs séquentiels dans les flux de données, Docteur en Sciences, France, 2009

[46] M.Fabrègue, Extraction d'informations synthétiques à partir de données séquentielles Application à l'évaluation de la qualité des rivières, doctorat, strasbourg, 2014.

[47] M.hassen, les règles d'association séquentielles, magister, 2006.

[48] A. Rachid, B.Tassadit, , K. Mamadou, Fouille de données : Règles séquentielles, master.

[49] E.Egho, Extraction de motifs séquentiels dans des données séquentielles multidimensionnelles et hétérogènes Une application à l'analyse de trajectoires de patients, doctorat, Lorraine, 2014.

[50] C.Raïssi, Extraction de séquences fréquentes : des bases de données statiques aux flots de données, Montpellier, 2008

[51] M.SadekReguiegYssaad , extraction de motifs séquentiels application à la maintenance industrielleAval / sonatrach, magister, université des Sciences et de la Technologie d'Oran Mohamed Boudiaf, 2010

[52]Mr. ALLIA Mohamed Rachid Mlle. BOUADI Tassadit Mr. El MOUTAOUKILSami Mr. KEIRA Mamadou,Fouille de donnéesRègles séquentielles , UNIVERSITE MONTPELLIER II.

[53] N. NAFFAKHI, Apprentissage supervisé pour la classification des images à l'aide de l'algèbre P-tree Université de Tunis Institut Supérieur de Gestion de Tunis, Février 2004.

[54] A.BenZakour, Extraction des utilisations typiques à partir de données hétérogènes historiées en vue d'optimiser la maintenance d'une flotte de véhicules, doctorat, Bordeaux, 2012.

[55]https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling

[56] https://www.python.org/

[57] https://www.jetbrains.com/pycharm/

[58] http://www.numpy.org/

[59] https://matplotlib.org

[60] https://pypi.org/project/gensim/

[61] https://en.wikipedia.org/wiki/PyTorch