UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR KHADY ALICE ITOUA

CLASSIFICATION, RÉSEAUX DE NEURONES ET RÉPRESENTATION VECTORIELLE DES DONNÉES TEXTUELLES : ÉTAT DE LA SITUATION

Octobre 2022

RÉSUMÉ

L'objectif de ce travail est de comparer les itemsets fréquents, dans le cadre de la classification de textes ou documents, à des méthodes de vectorisation textuelle classiques qui sont le sac de mots, le TF-IDF et le K-skip-N-gram respectivement de caractères et de mots. Nous les présentons chacune avec des exemples. Ensuite, nous introduisons les algorithmes de classifications utilisés. Il s'agit de la classification ascendante hierarchique (CAH), du k-médoïde, de la carte auto-organisatrice de Kohonen (SOM) qui ont été utilisés dans une étude de référence [Bokhabrine et al., 2019, Bokhabrine et al., 2020] au sein du département mais aussi, du support vector machin (SVM), adapté aux bases de grandes dimensions. Les données utilisées sont celles de l'étude de référence, à savoir, un corpus expérimental composé de textes avec des classes prédéfinies et le livre de Gustave Le Bon. Dans la perspective d'une analyse thématique et en se servant d'outils d'analyse comme notamment les formes fortes, les formes faibles et le taux de bonne classification pour le texte expérimental, nos résultats montrent que les 1-itemsets fréquents donnent de meilleurs résultats que le sac de mots et les K-skip-N-gram. Le Tf-idf semble toutefois être plus performant que le 1-Itemset.

ABSTRACT

The goal of this work is to compare the frequent itemsets, relating to document clustering and classification, with classical textual vectorization methods which are bag of words, TF-IDF and K-skip-N-gram respectively, the characters one and the words one. We present them each with examples. Then, we introduce used classification algorithms. These are the hierarchical upward classification (CAH), the k-medoid, the kohonen self-organizing map (SOM) that were used in a reference study [Bokhabrine et al., 2019, Bokhabrine et al., 2020] within the department but also the vector machin support (SVM), adapted to large dimensions data. The data used are that of the reference study, namely, an experimental corpus composed of texts with predefined classes and the book of Gustave Le Bon (Arabic civilisation). In perspective of a thematic analysis and using analytical tools such as strong forms, weak forms and the rate of good classification for the experimental text, our results show that frequent itemsets give better results than bag of words and K-skip-N-gram. However, Tf-idf seems to be more efficient than 1-Itemset.

AVANT-PROPOS

Ce projet de recherche s'inscrit dans le cadre général des méthodes de classification supervisées et non supervisées qui consistent à donner le meilleur regroupement en classes de l'ensemble à étudier. Qu'il s'agisse de méthodes statistiques classiques ou de méthodes neuronales, la classification est au coeur de nombreux domaines d'application en science des données. Elle a été étudiée de manière approfondie en termes de type de représentation des données, de mesure de ressemblance entre objets, et d'algorithmes de regroupement pour former des classes.

Je remercie tous ceux qui ont contribué à la réussite de ce travail et tout particulièrement, mes directeurs de recherche, Mme Nadia Ghazzali et M. Ismail Biskri. J'exprime ma reconnaissance aux membres du jury ayant accepté d'évaluer mon travail ainsi qu'au personnel du département de Mathématiques et d'Informatique, de l'Université du Québec à Trois-Rivières (UQTR).

Je dédie cet ouvrage :

À Papa et Maman, je vous aime, merci pour vos sacrifices et votre soutien sans faille !

À mon époux, merci chéri, nous formons une belle équipe !

A mon petit frère, mon ventricule droit !

A ma douce grande sœur, chère à mon cœur !

Aux familles ITOUA, MBENGUE et COLY !

À tous mes amis qui m'ont soutenue et encouragée !

TABLE DES MATIERES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1: Introduction

La science des données est pluridisciplinaire et fait appel à trois (03) grands axes que sont la connaissance des réalités du secteur (médecine, botanique, archéologie, finance, ...), la statistique et l'informatique. Ces dernières décennies, elle est passée du champ lexical de la donnée structurée et rare à celle d'une diversité et pléthore de données en tout genre. Avec internet, la donnée textuelle, notamment, s'est faite abondante (mails, forums, commentaires, livres électroniques, blogs, articles de presse,...), menant ainsi de plus en plus de chercheurs à se pencher sur son exploitation. On assiste alors au developpement de diverses applications comme le résumé automatique de texte, la classification de documents, l'analyse des sentiments ou encore, la reconnaissance des éléments grammatico-linguistiques. Dans le cadre de la classification de documents, de nombreuses études visant à décrire et représenter le mieux la donnée textuelle ont été menées. Nous présentons brièvement dans cette introduction, certaines des plus récentes. Elles sont certes importantes mais ne seront pas utilisées dans le cadre de ce mémoire qui s'inscrit dans la suite de précédents travaux [Bokhabrine et al., 2019] [Labiad, 2017] au sein du Département de Mathématiques et d'informatique (DMI) de l'Université du Québec à Trois-Rivières.

Commençons par présenter en substance, Word2Vec [Mikolov et al., 2013], une approche neuronale[1] produisant une représentation vectorielle du mot. Les vecteurs ainsi formés sont des word embeddings. Par définition, ce sont des vecteurs denses (non clairsemés), distribués, de longueur fixe, construits à l'aide de statistiques de cooccurence de mots selon l'hypothèse de distribution [Turian et al., 2010]. Cette hypothèse est définie par l'affirmation ‘’Words that occur in the same context tend to have similar meaning ‘’ [Harris, 1954]. Les vecteurs produits ont la taille du vocabulaire et cherchent à remplir une condition. Ils doivent en effet, soit permettre de prédire un contexte ou fenêtre de mots à partir du mot du milieu (Continuous Bag Of Word 'CBOW') soit l'inverse, c'est-à dire, trouver le mot du milieu en utilisant son contexte (Continuous Skip-gram). Sur de grands textes, les deux (02) approches conduisent à de meilleurs

[1]Elle utilise le neurone artificiel qui sera défini dans le chapitre III

résultats que sur des textes de petite taille. La première a l'avantage d'être plus rapide, de performer plus sur les textes courts et de mieux représenter les mots rares. La seconde quant à elle, sied mieux aux mots fréquents et arrive à refléter davantage les ressemblances sémantiques et syntaxiques entre les mots.

Glove [Pennington et al., 2014], diffère de Word2vec, dans la mesure où il exploite la matrice de colocation (voir chapitre II) pour proposer des vecteurs de mots. Les vecteurs formés sont tels qu'il soit possible [Mnih and Hinton, 2007] d'apprendre à prédire l'insertion du $n^{ième}$ mot en considérant le vecteur de ce mot comme étant une combinaison linéaire des n-1 mots précédents.

Contrairement à Word2Vec et Glove, ELMo offre une représentation vectorielle dynamique du mot [Peters et al., 2018]. En effet, pour un même mot, ELMo propose autant de vecteurs que de significations différentes selon le contexte. Par exemple, considérant les deux contextes ‘’vit le présent et non le passé‘’ et ‘’je t'offre ce présent en guise de reconnaissance‘’, le mot ‘’présent‘’ sera représenté par des vecteurs respectifs distincts. Cela est rendu possible grâce à une architecture particulière. Les vecteurs de mots ELMo sont calculés au-dessus, notamment, d'un modèle de langage bidirectionnel (biLSTM) à deux couches. Les LSTM (Long Short Terme Memory) sont un type particulier de réseau de neurones récurrents (prise en compte de séquences), capables d'apprendre des dépendances à long terme [Schmidhuber et al., 1997] [Staudemeyer and Morris, 2019]. ELMo prend en entrée des représentations de niveau caractère et permet d'outrepasser certaines contraintes habituelles liées à l'altération de l'orthographe du mot. En effet, la convolution utilisée par le modèle est faite à l'échelle des caractères et arrive à extraire des caractéristiques stables de la donnée en entrée de sorte à pouvoir la reconnaitre même en cas d'altération [Indolia et al., 2018] [Zhang et al., 2015]. Ainsi, aucun traitement préalable n'est requis sur le texte brut.

Bert (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2018], propose quant à lui, un traitement parallèle et non séquentiel de la phrase en entrée. Pour se faire, des couches neuronales de type transformeur sont utilisées. Ces derniers, sous leur version de base, sont généralement composés d'une partie dite encodeur et d'une autre dite décodeur. Cependant, Bert étant un modèle de représentation, la partie décodeur y est inexistante. Par

ailleurs, il a été démontré que les performances des transformeurs initiaux se dégradent rapidement à mesure que la longueur de la séquence d'entrée augmente [Cho et al., 2014]. Le traitement des éléments de la séquence en entrée, se faisant parallèlement, un vecteur de position est assigné à chacun d'eux avant l'encodage. Le transformeur se base uniquement sur le mécanisme d'attention [Vaswani et al., 2017], accordant ainsi une importance relative à tous les éléments de la séquence pour représenter chaque mot ou token de celle-ci. Il existe deux (02) types d'attention, l'attention additive [Bahdanau et al., 2014] et l'attention multiplicative [Britz et al., 2017]. Le premier type est celui utilisé par Bert. Ce dernier est composé de douze (12) blocs d'encodeurs. Les sorties intermédiaires issues de chaque bloc sont appelées "états cachés" ou "hidden states" en anglais. Ces multiples phases d'encodage des transformeurs permettraient d'extraire des informations précieuses en traitement du langage naturel [Tenney et al., 2019] telles que celles relatives au balisage des parties du discours ("POS tag" en anglais), les rôles sémantiques, les coréférences, etc. BERT, pour améliorer ses performances, utilise du bruit dans son apprentissage. Les auteurs [Devlin et al., 2018], ont 80% du temps, remplacé les mots par [MASK], 10% du temps, remplacé les mots initiaux par d'autres mots tirés au hasard tandis que les 10% restants étaient laissés comme tels. Selon le classement de la plateforme GLUE [Wang et al., 2019], un outil pour évaluer et analyser les performances des modèles à travers une gamme variée de tâches de traitement du language naturel, les modèles se basant sur Bert comptent actuellement parmi les plus performants.

Dans le cadre de ce mémoire, nous présentons et comparons différentes techniques de vectorisation de la donnée textuelle que sont le sac de mots, le K-skip-N-gram, le TF-IDF et l'itemset fréquent. Il existe plusieurs études comparant différentes méthodes de représentation de la donnée textuelle selon une finalité spécifique. C'est l'exemple de cet article de [Zhang et al., 2015]. Les auteurs y comparent des modèles traditionnels (sac de mots, les n-gram et le TF-IDF) à de l'apprentissage profond (ConvNets basés sur les mots, respectivement les caractères, les réseaux récurrents,...) sur plusieurs ensembles de données. Ils concluent que ConvNet au niveau des caractères est plus efficace. Y ont été pris en compte, bon nombre de facteurs dont notamment la taille de l'ensemble de données et l'organisation des textes dans le corpus.

Nous nous inscrivons pour notre part, dans la continuité des travaux de [Labiad, 2017], [Bokhabrine et al., 2020] et [Bokhabrine et al., 2019]. Ils ont démontré l'efficacité des itemsets

fréquents en classification de texte. Nous menerons pour se faire, multiples expérimentations afin d'en évaluer l'efficacité relative mais aussi absolue dans une perspective d'analyse thématique. Les méthodes de classification utilisées comprennent celles qui l'ont été par les deux auteurs citées en référence à savoir la classification ascendante hiérarchique -CAH-, très utilisée dans ce genre d'exercice, le K-médoïde, adapté aux données clairsemées de type binaire, la carte auto-organisatrice -SOM- de Kohonen, offrant une bonne appréciation visuelle des résultats mais aussi, le support vector machine -SVM- idéal pour les données de grandes dimensions.

Le corps du document est constitué de quatre (04) chapitres en excluant l'introduction et la conclusion. Le premier d'entre eux présente les techniques de vectorisation qui seront implémentées avec des exemples et illustrations. Le deuxième chapitre présente les méthodes de classification utilisées. Le troisième chapitre est celui de la méthodologie et le quatrième chapitre, celui des expérimentations et résultats.

CHAPITRE 2: Représentation vectorielle de la donnée textuelle

Etant non structurée, la donnée textuelle revêt plusieurs formes. Il peut s'agir de mots, groupe de mots, phrases, paragraphes, documents, etc. Son exploitation directe n'est pas toujours possible en traitement du langage naturel. Elle est alors représentée sous plusieurs formes dont notamment la représentation vectorielle. Le vecteur est une entité mathématique définie dans $(\mathbb{N}^n)_(n \epsilon \mathbb{N}_)$. Soit $V = (v_0, \ldots, v_n)_(n \epsilon \mathbb{N}_)$, un vecteur. V est caractérisé par sa norme $\|V\|$, sa direction $\bar{V}$ et son sens.

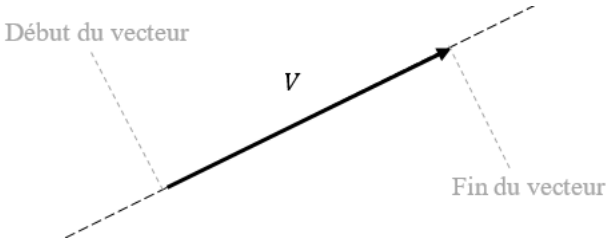


Figure 2.1: Vecteur

La norme $\|V\|$ est la distance entre le début du vecteur et la fin du vecteur. La direction $\bar{V}$ est la droite support du vecteur que nous voyons ci-dessus en pointillés et le sens est l'orientation de la flèche au niveau de la direction.

Plusieurs études ont été effectuées afin d'améliorer la qualité de ces représentations vectorielles. Ces recherches ont conduit à une succession de modèles de langue. Il en existe à ce jour une grande variété et dans ce chapitre, nous présentons ceux qui seront implémentés dans cette étude.

2.1 Le sac de mots

Appelé en anglais 'bag of word (BOW)', le sac de mot est le modèle de langue ou descripteur de texte le plus trivial [Harris, 1954]. Chaque mot du texte est associé à un effectif représentant le nombre de fois où ce mot apparait dans le texte. La longueur du vecteur ainsi

formé est égale à l'effectif des mots distincts constituant le texte. Considérons les deux (02) phrases Ph1 et Ph2 suivantes:

Ph1 = « Nos expérimentations ont confirmé l'importance des règles d'association pour des fins d'extraction de l'information pertinente et de qualité que l'on peut retrouver dans les données textuelles. » [Bokhabrine et al., 2019]

Ph2 = L'information est pertinente.

Ci-dessous, les tableaux de mappage mot/effectif permettant d'obtenir les vecteurs représentant Ph1 et Ph2.

Nos	1	fins	1	peut	1
expérimentations	1	extraction	1	retrouver	1
ont	1	de	1	dans	1
confirmé	1	information	1	les	1
importance	1	pertinente	1	données	1
des	1	et	1	textuelles	1
règles	1	qualité	1	l'	3
association	1	que	1	d'	2
pour	1	on	1	est	0

Table 2.1: Mappage mot/effectif de la phrase 1

Nos	0	fins	0	peut	0
expérimentations	0	extraction	0	retrouver	0
ont	0	de	0	dans	0
confirmé	0	information	1	les	0
importance	0	pertinente	1	données	0
des	0	et	0	textuelles	0
règles	0	qualité	0	l'	1
association	0	que	0	d'	0
pour	0	on	0	est	1

Table 2.2: Mappage mot/effectif de la phrase 2

Soit V1 et V2, les vecteurs représentant Ph1 et Ph2.

V1 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 2, 0)

V2 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1)

La méthode est simple. Cependant, elle conduit à des vecteurs de grandes dimensions et ne capte pas suffisamment l'information contenue dans la donnée textuelle. Son efficacité serait par exemple limitée pour les résultats des moteurs de recherche. En remplaçant les valeurs non

nulles par 1 au niveau des vecteurs, on obtient des one-hot vecteurs.

2.2 Le N-gram

Le N-gram est une forme d'association de mots ou de caractères. Il consiste en une association particulière de N unités continues. Pour être plus explicite, nous commençons déjà par former des 2-gram de mots. Notre texte, correspondant à la phrase Ph1 du point précedent, sera analysé en coupures successives de 2 mots comme le montre l'image ci-dessous. Les 2-gram de

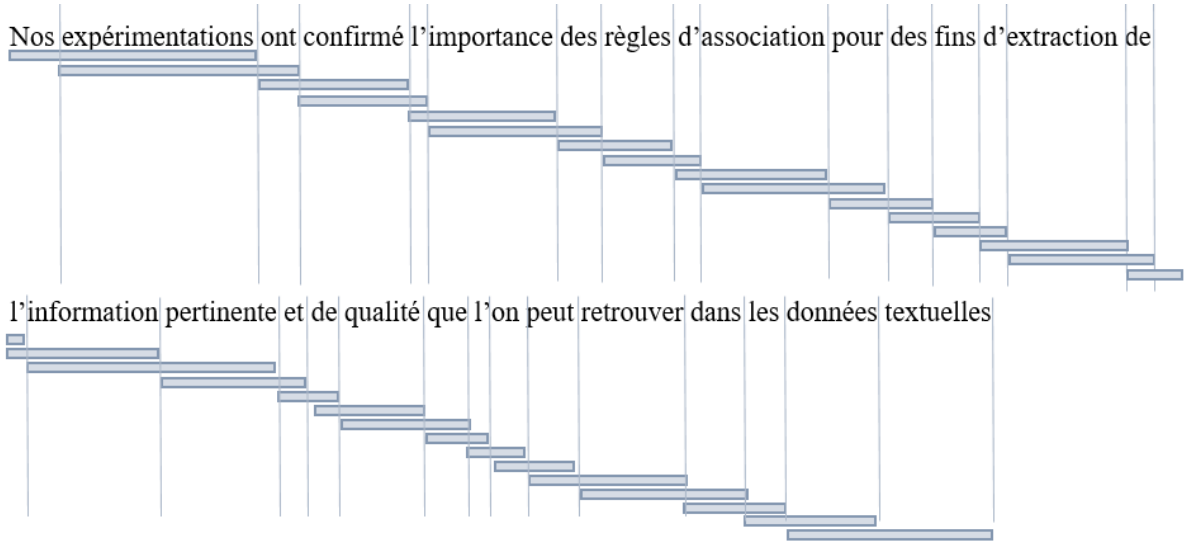


Figure 2.2: Formation des 2-gram

notre texte sont tous les couples de 2 mots successifs qu'il est possible d'y trouver. Les N-gram d'un texte sont toutes les associations successives possibles de N mots dans ledit texte. Pour 31 mots, trente (30) 2-gram ont pu être formés. Soit X un texte et NN-gram, le nombre de N-gram de X et L, le nombre de mots de X :

$$NN - gram = L + 1 - N.$$

Une fois les paires formées, nous pouvons construire une matrice de colocation. Les colocations sont des mots dont l'occurence ensemble est fréquente. La matrice prend en ligne comme en colonne les mots contenus dans le corpus et cela, sans doublons. Sur la figure suivante '**nos expérimentations**' et '**données textuelles**' apparaissent une fois dans le corpus par exemple alors que '**donnée pertinente**' n'est pas rencontré.

	Nos	expérimentations	ont	confirmé	l'	importance	des	règles	d'	association	pour	fins	extraction	de	information	pertinente	et	qualité	que	on	peut	retrouver	dans	les	données	textuelles
Nos	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
expérimentations	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ont	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
confirmé	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l'	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
importance	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
des	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
règles	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d'	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
association	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pour	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fins	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
extraction	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
information	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
pertinente	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
et	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
qualité	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
que	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
peut	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
retrouver	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
dans	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
les	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
données	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
textuelles	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.3: Matrice de colocation

Le vecteur vec(l'), associé à « l'» dans notre matrice de colocation est :

$$vec(l') = (0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0).$$

Elle permet couramment d'attribuer une probabilité d'occurence à un mot ou groupe de mots. Il s'agit d'une probabilité conditionnelle par rapport aux mots précédents. Soit $Ph = [m_1, ..., m_w]$, une phrase. La probabilité que le mot m_w soit le dernier mot de la phrase Ph est :

$$\begin{aligned} P(m_1, m_2, ..., m_w) &= P(m_1, m_2, ..., m_{w-1})P(m_w|m_1, m_2, ..., m_{w-1}) \\ &= P(m_1, m_2, ..., m_{w-2})P(m_{w-1}|m_1, m_2, ..., m_{w-2})P(m_w|m_1, m_2, ..., m_{w-1}) \\ &= ... \\ &= P(m_1)\prod_{i=2}^{w} P(m_i|m_1, m_2, ..., m_{i-1}). \end{aligned}$$

Il s'agit ici d'une application de la règle de la chaine de probabilité résultant de l'application de la règle du produit : $P(m_1, m_2) = P(m_1)P(m_2|m_1)$. Soit $N[m_1, m_2]$ et $N[m_1]$, respectivement le nombre de fois que se retrouvent $[m_1, m_2]$ et $[m_1]$ dans le texte :

$$P(m_2|m_1) = \frac{N[m_1, m_2]}{N[m_1]}.$$

Au-delà de la complexité computationnelle, cette approche ne sied pas toujours à la réalité parce qu'aucun corpus ne contient toutes les phrases et leurs tournures possibles ni même, n'est suffisamment riche pour permettre une approximation.
L'hypothèse de Markov [Friedman and Halpern, 2013] est celle selon laquelle le comportement futur d'un système dynamique dépend uniquement de son historique le plus récent et non de tout son historique. L'algorithme en question se base sur la formule suivante :

$$\begin{aligned} P(m_w|m_1, m_2, ..., m_{_{w-1)}} &\approx P(m_w|m_{w-N+1}, ..., m_{w-1}) \\ &\approx P(m_w|m_{w-1}). \end{aligned}$$

La probabilité d'occurence des 2-gram se calcule donc ainsi :

$$P(m_i|m_{i-1}) = \frac{N[m_{i-1}, m_i]}{N[m_{i-1}]}$$

Pour donc avoir notre tableau de probabilité de 2-gram, il suffit de normaliser le tableau précédent par les fréquences des 1-gram correspondant :

Nos	expérimentations	ont	confirmé	l'	importance	des	règles	d'	association
1	1	1	1	3	1	3	1	3	1

pour	des	fins	extraction	information	pertinente	...	les	données	textuelles
1	1	1	1	1	1	...	1	1	1

Table 2.3: Fréquence des 1-gram

Ainsi: $P(l'|importance) = P(l'|information) = P(l'|on) = 1/3$

Le N-gram de caractères est souvent privilégié au N-gram de mots pour certaines tâches du traitement du language naturel. Il occasionnerait moins de perte d'information et parviendrait à mieux capter les néologismes, le langage informel (exemple de « mdr » pour « mort de rire ») et même les émoticônes [Kim et al., 2018]. Selon ces derniers auteurs, les vecteurs issus de la matrice de colocation, une fois entrainés, seraient à même de refléter la sémantique et la syntaxe du mot.

2.3 Le K-Skip-N-gram

Une généralisation du N-gram est le K-skip-N-gram communément appelé skip-gram. L'appellation K-skip-N-gram est cependant la plus correcte. Le K signifie que pour la construction de chaque N-gram, 0 à K sauts seront effectués. Le N-gram est le cas particulier où K=0 et qu'il n'y a donc aucun saut (skip). Considérons le texte suivant: 'Nos expérimentations ont confirmé' et construisons nos 2-skip-2-gram. Comme le montre la figure suivante, nous en trouvons six (06) contre trois (03) 2-gram. L'éventail de configurations offert par le K-skip-N-gram est plus large. Cela expliquerait le fait qu'en production, il soit souvent plus performant que le N-gram.

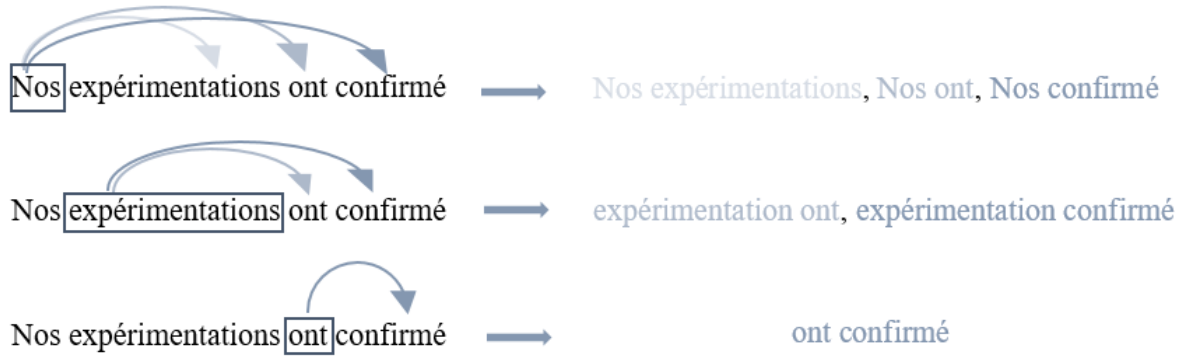


Figure 2.4: Prise en compte du paramètre skip, N=0,1,2. K-skip-N-gram

La modélisation skip-gram serait plus efficace pour couvrir les 3-gram que le fait "d'augmenter la taille du corpus d'entraînement (voire le quadrupler), tout en minimisant la désinformation. Bien que les skip-gram puissent générer des N-gram inutiles, ceux-ci ont tendance à ne pas affecter la couverture des N-gram dans des documents différents" [Kim et al., 2018]. En d'autres termes, l'utilisation du skip-gram, donc du K-skip-N-gram, permet l'extraction de l'information tout comme l'aurait fait le N-gram mais avec une perte moindre d'information. Il englobe ce dernier. En effet, le K-skip-N-gram est équivalent au N-gram pour K=0.
Il aurait par ailleurs, été démontré par une étude [Krasnoshtan, 2019] que pour un corpus de taille L, le nombre de K-skip-N-gram de mots qui puissent être formé est :

$$\frac{LN + N + K' - N^2 - NK'}{N} \binom{N - 1 + K^{'}}{N - 1}.$$

Avec $K' = \min(L - N + 1, K)$.

2.4 Le TF-IDF

Le TF-IDF est un descripteur basé sur la fréquence du mot (Term Frequency 'TF') et sa fréquence inverse de document (Inverse Document Fréquency 'IDF'). Lorsque l'on cherche à extraire l'information d'un document parmi tant d'autres, on ne s'intéresse pas au vocabulaire usuel et incontournable mais plutôt à ce qui fait la spécificité de son contenu (les mots clés). Plus le mot sera rare, plus son IDF sera grand. Pour un terme t dans un document d, le poids $w_{t,d}$ du terme t dans le document d est donné par:

$$\mathrm{w}_{t,d} = TF \times \log(IDF).$$

TF = fréquence du mot dans le document

IDF = nombre total de document/nombre de document où apparaît le mot

Soit A et B, 2 documents.

Document A = 'the man went out for a walk'

Document B = 'the children sat around the fire'

Bag of words correspondant:

Bow_A = ['the', 'man', 'went', 'out', 'for', 'a', 'walk']

Bow_B = ['the', 'children', 'sat', 'around', 'the', 'fire']

La table d'occurence est la suivante: La table TF-IDF est la suivante:

	a	around	children	fire	for	man	out	sat	the	walk	went
A	1	0	0	0	1	1	1	0	1	1	1
B	0	1	1	1	0	0	0	1	2	0	0

Table 2.4: Occurence dans le sac de mots

	a	around	children	fire	for	man	out	sat	the	walk	went
A	0.043	0	0	0	0.043	0.043	0.043	0	0	0.043	0.043
B	0	0.05	0.05	0.05	0	0	0	0.05	0	0	0

Table 2.5: TF-IDF à partir des occurences du sac de mots

$$\mathrm{w}_{the,A} = 1/7 \times \log(2/2) = 0$$

$$\mathrm{w}_{around,B} = 1/6 \times \log(2/1) = 0.5$$

Cette technique a l'avantage de ne pas avoir à traiter les mots fonctionnels. Elle met en avant les mots rares qui souvent permettent de marquer la singularité du texte.

2.5 Les Itemsets fréquents

Pour présenter les Itemsets fréquents, nous utilisons dans cette partie, le modèle du panier de la ménagère permettant de comprendre les règles d'association [Listiawan and Hudha, 2021]

et présentons l'algorithme Apriori.
Les itemsets fréquents sont la réponse aux questions suivantes :

- Quels articles, produits ou items reviennent souvent ?
- Quels sont les items dont l'occurence garantit une certaine fréquence, seuil 'support minimum'?
- Quels items ont tendance à se retrouver dans un même panier ?
- A quel point peut-on avoir confiance en ces associations ?

Le support d'un item est le rapport du nombre de paniers dans lesquels cet item se trouve et du nombre total de paniers. Le support minimum choisi par l'analyste est donc celui en dessous duquel les items ne lui sont d'aucun intérêt particulier. Il est parfois exprimé en effectif, alors seul le numérateur est pris en compte. Les items respectant ce seuil forment alors les itemsets fréquents. Selon leur longueur k, l'appellation k-itemset est souvent utilisée. Par exemple : l'itemset {shampoing, démêlant} est un 2-itemset. La confiance d'un k-itemset (k>1) est la probabilité de rencontrer dans un panier le $k^{ième}$ item en présence des k-1 premiers items . Le fait de trouver des règles d'association implique le choix d'un support s, d'une confiance c mais aussi d'une longueur l (nombre d'items de l'association). Cependant, la nature de nos travaux ne nécessite pas que nous développions davantage sur les règles d'association.

Supposons que l'on veuille trouver tous les k-itemsets d'une base de données de transactions T contenant n items. Le nombre de k-itemset $(1 \leq k \leq n)$ est $2^n - 1$. La figure ci-dessous montre le nombre de k-itemset pour 5 items.

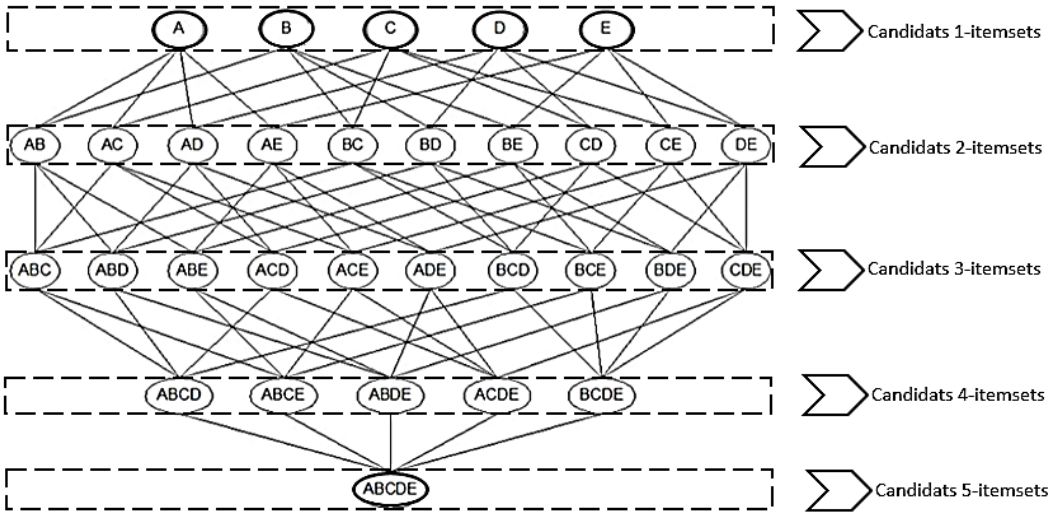


Figure 2.5: Candidats k-itemset fréquents

Pour trouver les itemsets fréquents, l'analyse de tous les k-itemsets engendrerait un énorme cout computationnel rendant quasi impossible l'implémentation dans des situations réelles (milliers de transactions journalières par exemple avec les grandes chaines d'approvisionnement). Une solution est d'y arriver en réduisant le nombre candidats k-itemset. C'est dans ce cadre qu'intervient l'algorithme Apriori. Nous présentons ci-dessous l'algorithme tel qu'initialement présenté par [Agrawal and Srikant, 1994].

La fonction génératrice de k-itemsets candidats s'appelle AprioriGen (), elle se sert des k-1-itemsets fréquents pour réduire le nombre initial de candidats k-itemset fréquents. Par exemple, si parmi les items A, B, C, D et E, seuls A et C sont sélectionnés comme 1-itemsets fréquents avec le support minimum choisi, alors le seul candidat 2-itemset fréquent présenté par AprioriGen (), sera AC. Soit C_k l'ensemble des candidats k-itemset fréquents, minsup, le support minimum, L_k l'ensemble des k-itemsets fréquents, n le nombre total d'items et T l'ensemble des transaction distincts. L'algorithme Apriori permettant d'obtenir L, l'ensemble de tous les itemsets fréquents, est le suivant:

Algorithme 1 : Apriori

Entree: T, minsup

$k \leftarrow 2$;
Tant que $L_{k-1} \neq \phi$ **faire**
$C_k = AprioriGen(L_{k-1})$
$L_k \leftarrow selectionAvecSupport(C_k, minsup)$
$k \leftarrow k + 1$
fin Tant que

Sortie: $L = \bigcup_{k=1}^{n} L_{k-1}$.

Exemple :

Soit le tableau de transactions suivant :

T1	A, E, C
T2	D, B, A
T3	B, A, D
T4	E, C, D
T5	A, B
T6	B, A, D

Table 2.6: Ensemble de transactions

En fixant le support minimum à 3, la figure suivante montre l'action de l'algorithme Apriori sur la réduction des candidats et la rapidité de convergence occasionnée pour trouver les itemsets concernés.

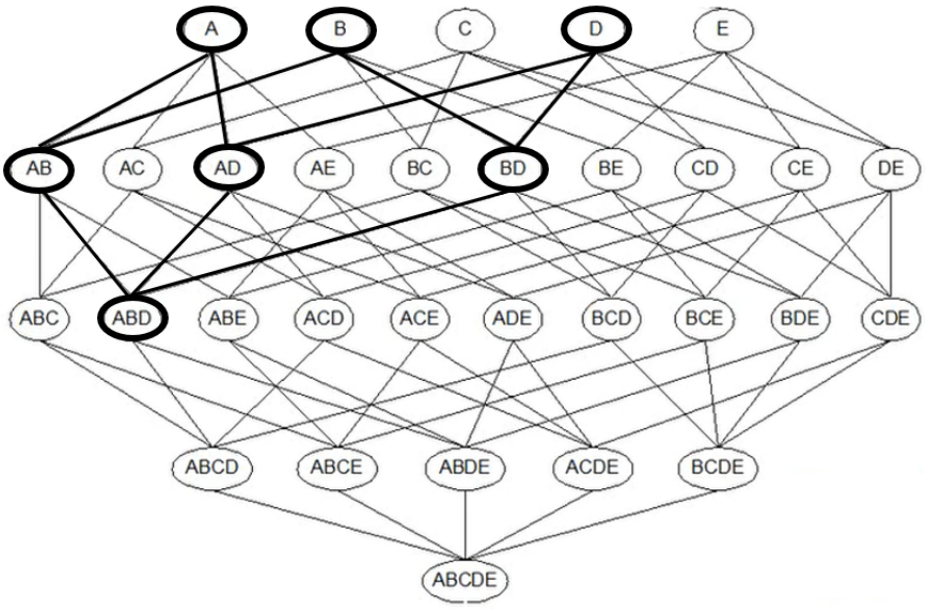


Figure 2.6: Réduction des candidats et convergence accélérée de l'Algorithme Apriori

Les 1-itemsets fréquents sont $L_1 = A, B, D$. Le nombre de candidats 2-itemsets fréquents passe de 10 à 3 avec AprioriGen (). $C_2 = L_2 = AB, BD, AD$ et le nombre de candidats 2-itemset passe de 10 à 1 : $C_3 = L_3 = ABD$. Les itemsets fréquents sont donc au nombre de 7 avec un support minimum de 3 soit : A, B, D, AB, BD, AD et ABD.

CHAPITRE 3: Méthodes de classification statistiques et neuronales

Les méthodes de classification cherchent à classer l'ensemble des individus en des groupes homogènes. Les individus d'une même classe doivent se ressembler par rapport aux variables de l'analyse. Le nombre de classes à former est généralement inconnu mais doit être relativement faible pour que la méthode puisse être considérée comme un résumé de l'information.

Se distinguent traditionnellement les méthodes non hiérarchiques de celles dites de partitionnement. D'autres méthodes dites neuronales ont par la suite vu le jour. Dans notre étude, nous nous inscrivons principalement dans une démarche de classification non supervisée (classification ascendante hiérarchique « CAH », k-médoïde, et le self organizing map « SOM »), c'est-à-dire que l'information sur la réelle appartenance de l'individu à une classe naturelle n'est pas fournie à l'algorithme afin de le guider. Néanmoins, nous intégrons aussi à l'étude une méthode supervisée (support vector machines « SVM »). Contrairement à la classification non supervisée, pour la classification supervisée, les classes existantes sont connues tout comme le groupe auquel appartient chaque individu. Il s'agit alors de classer ces derniers dans la bonne classe en se servant des variables.

3.1 Mesure de similarité et de dissimilarité

La dissimilarité ou dissemblance d est une mesure permettant d'estimer l'éloignement entre des individus tandis que la similarité ou ressemblance s appréhende la proximité.

La dissimilarité respecte trois (03) propriétés fondamentales. $\forall$ i $\neq$ j où i et j sont des entités:

1. La non négativité : $d(i,j) \geq 0$.
2. L'identité : $d(i,i) = 0$.
3. La symétrie: $d(i,j) = d(j,i)$.

On note que la similarité entre i et j est donnée par $s(i,j) = T - d(j,i)$ avec T, la valeur maximale que peut prendre d. Ci-dessous sont citées quelques exemples de dissimilarité :

- distance euclidienne : $d(i,i') = \sqrt{\sum_k (x_i^k - x_{i'}^k)^2}$.
- distance du city-block ou de Manhattan : $d(i,i') = \sum_k |x_i^k - x_{i'}^k|$.
- distance de Tchébitchev : $d(i,i') = \sup_k |x_i^k - x_{i'}^k|$.

Où x_i^k est la valeur de la variable k pour l'individu i.
La dissimilarité entre A et B, deux groupes d'individus à classifier est notée D(A, B). Elle est appelée critère ou indice d'agrégation. Il en existe de plusieurs sortes.

L'indice du saut minimum Cet indice consiste à considérer que la dissimilarité entre A et B est le minimum des dissemblances entre les individus appartenant à A et ceux de B.

$$D(A,B) = \min_{i \epsilon A, i' \epsilon B} d(i,i').$$

L'indice du saut maximum Cet indice consiste à considérer que la dissimilarité entre A et B est le maximum des dissemblances entre les individus appartenant à A et ceux de B.

$$D(A,B) = \max_{i \epsilon A, i' \epsilon B} d(i,i').$$

L'indice du saut moyen Cet indice consiste à considérer que la dissimilarité entre A et B est la moyenne des dissemblances entre les individus appartenant à A et ceux de B.

$$D(A,B) = \frac{1}{Card(A)Card(B)} d(i,i')_{i \epsilon A, i' \epsilon B}.$$

L'indice d'agrégation barycentrique Cet indice utilise le centre de gravité des groupes. Il s'agit d'un point ayant pour coordonnées, les moyennes des individus appartenant au groupe. Si les variables décrivant les n_A individus du groupe A sont au nombre de C alors le centre de

gravité de A est $g_A = (\overline{x}_1, \overline{x}_2, ..., \overline{x}_C)$ où $\overline{x}_C = \frac{1}{n_A} \sum_{k=1}^{n_A} x_C^k$.

$$D(A, B) = \| g_A - g_B \|^2 .$$

L'indice d'agrégation de ward Cet indice, encore appelé celui de la perte d'inertie semble être le plus utilisé pour des raisons que nous évoquerons dans la description de la classification ascendante hiérarchique (CAH) suivante.

$$D(A, B) = \frac{p_A * p_B}{p_A + p_B} \|g_A - g_B \|^2 .$$

Où $p_A = \frac{n_A}{n}$ avec n, le nombre total d'individus à classer.

3.2 La classification ascendante hiérarchique (CAH)

3.2.1 Le principe de la méthode

Dans sa version la plus simple, la CAH fournit une chaine de partitions en utilisant l'algorithme suivant :

- **Etape 0** Chaque individu constitue une classe à part entière.
- **Etape 1** On regroupe les deux individus les plus proches formant ainsi n-l classes où n est le nombre total d'individus à classifier.
- **...**
- **Etape k** Ainsi de suite, le regroupement des deux classes les plus proches crée à cette étape, (n - k) classes.
- **...**
- **Etape n-1** A cette étape, on obtient une seule classe contenant tous les n individus.

Ces agrégations successives sont résumées par une sortie graphique, un arbre de classification appelé le dendrogramme. Ce dernier définit un système dichotomique emboîté de classes. En effet, chaque nœud se forme au regroupement de deux (02) entités.

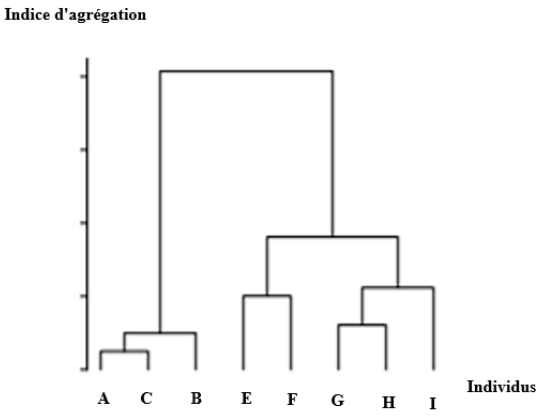


Figure 3.1: Dendrogramme, résumé des agrégations

3.2.2 La CAH selon le critère de Ward

La CAH selon le critère de Ward consiste à chercher à minimiser à chaque étape (regroupement de 2 sous parties de E, l'ensemble des n individus à classifier), la perte d'information. Soit $P = c_1, c_2, \ldots, c_K$, une partition de E en K classes. L'inertie totale de cette partition par rapport au centre de gravité g de E est :

$$
\begin{aligned}
I_t &= \sum_{i=1}^{n} p_i \left\| x_i - g \right\|^2 \\
&= \sum_{k=1}^{K} \sum_{i \epsilon c_k} p_i \left\| x_i - g_{c_k} + g_{c_k} - g \right\|^2 \\
&= \sum_{k=1}^{K} \sum_{i \epsilon c_k} p_i [\left\| x_i - g_{ck} \right\|^2 + \left\| g_{c_k} - g \right\|^2 + 2(x_i - g_{c_k}, g_{c_k} - g)] \\
&= \sum_{k=1}^{K} \sum_{i \epsilon c_k} p_i [\left\| X_i - g_{c_k} \right\|^2 + \sum_{k=1}^{K} \sum_{i \epsilon c_k} p_i \left\| g_{c_k} - g \right\|^2 + 2 \sum_{k=1}^{K} \sum_{i \epsilon c_k} p_i (x_i - g_{c_k}, g_{c_k} - g)].
\end{aligned}
$$

or $(X_i - g_{c_k}, g_{c_k} - g) = 0$, alors :

$$\begin{aligned} I_t &= \sum_{k=1}^{K} p_{c_k} \sum_{i \epsilon c_k} \frac{p_i}{p_{c_k}} [\|x_i - g_{c_k}\|^2 + \sum_{k=1}^{K} \sum_{i \epsilon c_k} p_i \|g_{c_k} - g\|^2 \\ &= \underbrace{\sum_{k=1}^{K} p_{c_k} \sum_{i \epsilon c_k} \frac{p_i}{p_{c_k}} \|X_i - g_{c_k}\|^2}_{intra-classe} + \underbrace{\sum_{k=1}^{K} p_{c_k} \|g_{c_k} - g\|^2}_{inter-classe}. \end{aligned}$$

Théorème de Huygens: L'inertie totale étant constante pour chaque jeu de données, l'inertie intra-classe augmente au fur et à mesure que les regroupements sont effectués tandis que l'inertie inter-classe, elle, diminue.

Le calcul de niveau après agrégation peut s'effectuer à l'aide de la formule de récurrence de Lance WILLIAMS:

$$D^2(A \cup B, M) = \frac{1}{p_A + p_B + p_M} [(p_A + p_M) D^2(A, M) + (p_B + p_M) D^2(B, M) - (p_A + p_B) D^2(A, B)].$$

Si on agrège 2 classes A et B de E, on peut poser sans perte de généralité $A = c_K$ et $B = c_{K-1}$. La nouvelle partition obtenue sera $p'(c_1, c_2, \ldots, c_{K-2}, A \cup B)$. L'inertie inter-classe de la nouvelle partition est :

$$\begin{aligned} I_{p'}^{inter} &= \sum_{k=1}^{K-2} p_{c_k} \|g_{c_k} - g\|^2 + p_{A \cup B} \|g_{A \cup B} - g\|^2 \\ &= \sum_{k=1}^{K-2} p_{c_k} \|g_{c_k} - g\|^2 + (p_A + p_B) \left\| \frac{p_A g_A + p_B g_B}{p_A + p_B} - g \right\|^2 \\ &= \sum_{k=1}^{K-2} p_{c_k} \|g_{c_k} - g\|^2 + \frac{p_A^2}{p_A + p_B} \|g_A - g\|^2 + \frac{p_B^2}{p_A + p_B} \|g_B - g\|^2 + \\ &\quad \frac{2 p_A p_B}{p_A + p_B} (g_A - g, g_B - g). \end{aligned}$$

La perte d'information qui résulte de l'agrégation des 2 classes A et B est donc :

$$
\begin{aligned}
I_p^{inter} - I_{p'}^{inter} &= p_A \|g_A - g\|^2 + p_B \|g_B - g\|^2 - \frac{p_A^2}{p_A + p_B} \|g_A - g\|^2 - \frac{p_B^2}{p_A + p_B} \|g_B - g\|^2 - \\
&\quad \frac{2p_A p_B}{p_A + p_B}(g_A - g, g_B - g) \\
&= \frac{p_A p_B}{p_A + p_B} \|g_A - g\|^2 + \|g_B - g\|^2 - 2(g_A - g, g_B - g) \\
&= \frac{p_A p_B}{p_A + p_B} \|(g_A - g) - (g_B - g)\|^2 \\
&= \frac{p_A p_B}{p_A + p_B} \|(g_A - g_B)\|^2 .
\end{aligned}
$$

La perte d'information est quantifiable avec Ward d'où son choix pour les expérimentations dans cette étude. C'est une des raisons pour lesquelles, elle est généralement plus utilisée.

3.2.3 Choix du nombre de classes

Une fois le dendrogramme obtenu, une droite horizontale peut le couper, définissant ainsi un nombre de classes. L'endroit optimal de la coupure peut être déterminé avec un critère ou indice. Il en existe plusieurs. [Milligan and Cooper, 1985], ont évalué trente (30) critères pour déterminer le nombre de grappes sur des données simulées avec des classes parfaitement distinctes. "Plusieurs procédures ont assez bien fonctionné, tandis que d'autres ont plutôt mal fonctionné. Ainsi, ce dernier groupe de règles semble avoir peu de validité, en particulier pour les ensembles de données contenant des grappes distinctes. Les chercheurs impliqués sont invités à sélectionner un ou plusieurs des meilleurs critères. Cependant, les utilisateurs sont avertis que la performance de certains des critères peut dépendre des données". A ce jour, peu nombreuses sont les librairies de logiciel d'analyse de données qui soient munies d'une grande variété d'indices. Parmi elles, un exemple notoire est NBClust [Charrad et al., 2014], un package du logiciel R qui fournit une liste de trente (30) indices.

”ch”	”duda”	”pseudot2”
”cindex”	”gamma”	”beale”
”ccc”	”ptbiserial”	”gplus”
”db”	”frey”	”hartigan”
”tau”	”ratkowsky”	”scott”
”marriot”	”ball”	”trcovw”
”trace”	”friedman”	”mcclain”
”rubin”	”kl”	”silhouette”
”gap”	”dindex”	”dunn”
”hubert”	”sdindex”	”sdbw”

Table 3.1: 30 indices dans NBClust

3.2.4 Exemple de CAH

Dans cette partie est présenté un exemple d’agrégation partant de la matrice des distances entre 5 individus afin de faciliter la compréhension de la méthode selon le saut minimum et le saut maximum.

– Indice du saut minimum

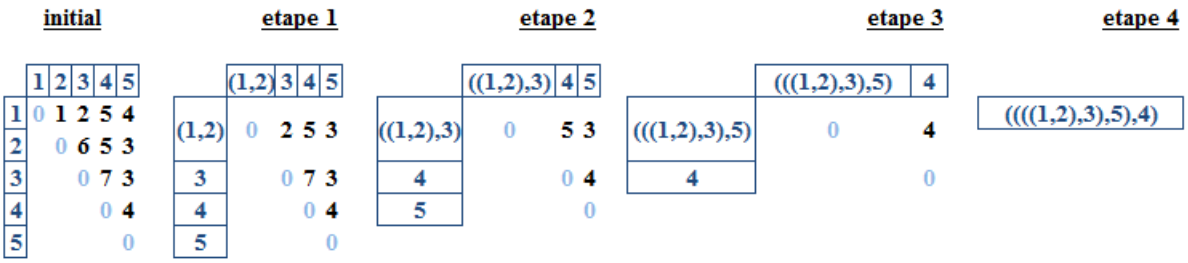


Figure 3.2: Exemple agrégation saut minimum

D’où le diagramme :

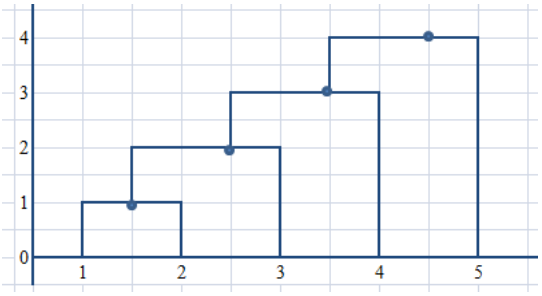


Figure 3.3: Dendrogramme saut minimum

– Indice du saut maximum

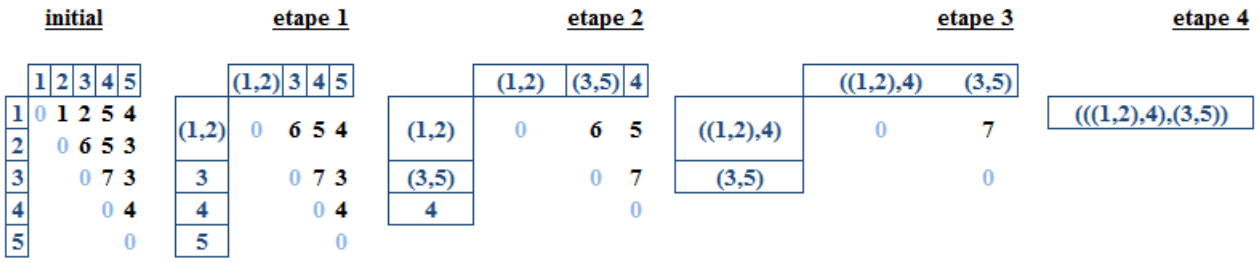


Figure 3.4: Exemple agrégation saut maximum

D'où le diagramme :

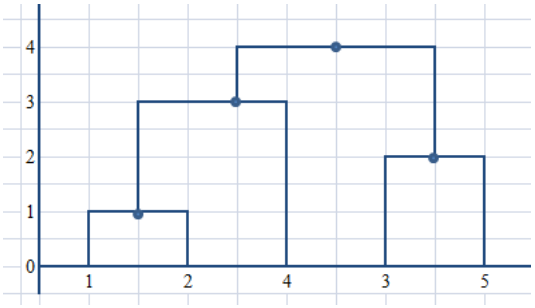


Figure 3.5: Dendrogramme saut maximum

3.2.5 Avantages et inconvénients

Le principal avantage de la CAH est la facilité d'interprétation des résultats graphiques qu'elle propose (exemple du dendrogramme). Aussi, elle ne nécessite pas de choix de nombre de classes au préalable.

L'inconvénient le plus notoire quant à lui, est l'instabilité de ses résultats, qui varient selon la paramétrisation pour un même jeu de données (choix de la dissemblance et de la méthode d'agrégation). Aussi, dans sa version la plus simple, on note une lenteur des calculs avec un important volume de données. Il en existe cependant plusieurs variantes pour réduire le temps de calcul (exemple de celle voisins reciproques) et les machines sont désormais plus puissantes.

3.3 Le K-médoïde

En statistiques, un médoïde est le représentant (parmi les individus), le plus central d'une classe. L'algorithme des k-médoïdes est un algorithme qui consiste à construire des classes autour des médoïdes. L'algorithme fonctionne de manière itérative pour affecter chaque point de données à l'un des K groupes en fonction de la similarité des caractéristiques.

3.3.1 L'algorithme du K-médoïde

Soit $(X_1, \ldots, X_n)$ des observations. Il en existe deux (02) principales variantes. L'une est sans contrainte (Partitions around medoids (PAM)) tandis que l'autre, contraint la mise à jour des médoïdes au sein de la même classe initiale (alternate K-medoid). Tandis que la première est plus fiable, la deuxième est plus rapide. L'algorithme PAM est le plus utilisé et se décrit comme suit :

Initialisation (clusters initiaux)

Choix de k observations $(m_1, \ldots, m_k)$.

Affectation des points X_i à la classe du représentant m_i (médoïde) le plus proche :

$$m_{(i)} = \text{argmin}(dist(m_{t\epsilon(1,k)}, X_i)), \;\; X_i \, \epsilon \, classe(m_i).$$

Calcul de la dissimilarité totale

$$DT = \sum_{i=1}^{N} dist(m_i, X_i).$$

Répéter tant que DT décroît :

Pour chaque médoïde m_i :

Pour chaque point X_i non médoïde :

Permuter X_i et m_i et recalculer DT :

Si DT diminue :

Permutation validée

Sinon :

Permutation annulée

3.3.2 Exemple PAM (partioning around medoïds):

Soit le tableau de données suivant, contenant huit (08) observations de coordonnées dans N^2:

	X	Y
1	2	10
2	2	5
3	8	4
4	5	8
5	7	5
6	6	4
7	1	2
8	4	9

Table 3.2: Données pour exemple PAM

Pour les besoins de l'illustration, nous présentons ci-dessous la matrice des distances euclidiennes.

	1	2	3	4	5	6	7	8
1		5	8,49	3,61	7,07	7,21	8,06	2,24
2			6,08	4,24	5	4,12	3,16	4,47
3				5	1,41	2	7,28	6,40
4					0	4,12	7,21	1,41
5						1,41	6,71	5
6							5,4	5,39
7								7,62
8								

Table 3.3: Matrice des distances euclidiennes

Le tableau suivant montre le choix des médoïdes initiaux et l'affectation des observations pour la formation des classes.

DT = dist(3,5)+dist(3,6)+dist(3,3)+dist(2,1)+dist(2,4)+dist(2,8)+dist(2,2)+dist(7,7)

K=3	Médoïdes I	Médoïdes II	Médoïdes III	Affectation
Médoïdes initiaux	3	2	7	
1	8,49	5	8,06	II
2	6,08	0	3,16	II
3	0	6,08	7,28	I
4	5	4,24	7,21	II
5	1,41	5	6,71	I
6	2	4,12	5,39	I
7	7,28	3,16	0	III
8	6,40	4,47	7,62	II
Classes	{3,5,6}	{1,2,4,8}	{7}	

Table 3.4: Formation des classes initiales PAM

DT = 1,41+2+0+5+4,24+4,47+0+0

DT = 17,12 .

Le tableau suivant montre la première permutation validée de l'algorithme sans contrainte. Les observations sont réaffectées au médoïde.

K=3	Médoïdes I	Médoïdes II	Médoïdes III	Affectation
Médoïdes initiaux	5	2	7	
1	7,07	5	8,06	II
2	5	0	3,16	II
3	1,41	6,08	7,28	I
4	0	4,24	7,21	I
5	0	5	6,71	I
6	1,41	4,12	5,39	I
7	6,71	3,16	0	III
8	5	4,47	7,62	II
Classes	{3,4,5,6}	{1,2,8}	{7}	

Table 3.5: Permutation validée PAM et réaffectation

DT = dist(5,3)+dist(5,4)+dist(5,6)+dist(5,5)+dist(2,1)+dist(2,8)+dist(2,2)+dist(7,7)

DT = 1,41+0+1,41+0+5+4,47+0+0

DT = 12,29.

Aucune autre permutation n'a pu être validée en faisant décroître DT, les classes finales sont donc {3,4,5,6}, {1,2,8} et {7}.

3.3.3 Avantages et inconvénients

Comme avantages de l'algorithme du K-médoïde, on note principalement sa rapidité de convergence, sa faible sensibilité aux valeurs abérrantes et sa facilité d'implémentation. Comme inconvénients, on note l'obligation de choisir un nombre de classes K et la possibilité d'avoir des résultats différents sur une même donnée vu le choix aléatoire des médoïdes initiaux.

3.4 Le réseau de neurone artificiel

Le neurone est l'unité de travail de base du cerveau. C'est une cellule spécialisée dans le traitement et la transmission de l'information (signal bio électrique). Chaque neurone est composé de ramifications (dendrites) d'où proviennent les informations, d'un corps cellulaire contenant un noyau, d'un prolongement du neurone qui conduit le signal électrique du corps cellulaire vers les zones synaptiques, appelé axone. Les axones et dendrites de neurones différents entrent en contact et transmettent l'information de cellule à cellule via des structures spécialisées : les synapses.

Le neurone artificiel imite le neurone biologique. Les synapses du neurone sont modélisées sous forme de poids. La force de la connexion entre une entrée et un neurone est appréhendée par la valeur du poids. [Dongare et al., 2012] Une phase de jonction est prévue pour additionner toutes les entrées pondérées (combinaison linéaire) puis, une fonction d'activation contrôle l'amplitude de la sortie du neurone.

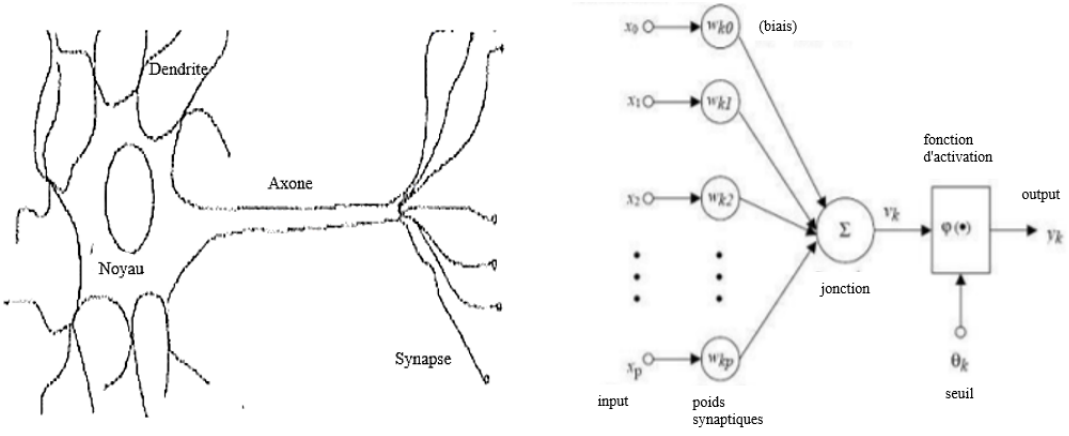


Figure 3.6: Neurone biologique vs neurone artificiel

3.4.1 Réseau de neurones (fonctionnement courant)

Un réseau neuronal artificiel est un système qui se compose d'une collection d'unités connectées appelées neurones (cercles, voir figure 3.7) qui sont organisées en ce que nous appelons des couches (chaque couche avec une couleur distincte, voir figure 3.7). La figure suivante montre la représentation la plus triviale : une couche d'entrée ou d'input, une couche cachée (possibilité d'en mettre plusieurs) et une couche de sortie ou d'output.

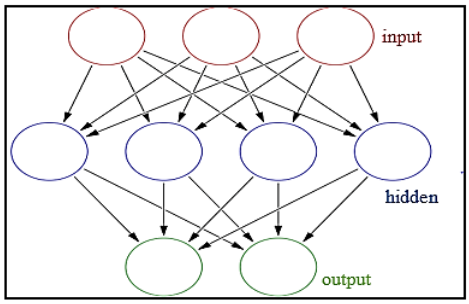


Figure 3.7: Exemple d'un réseau de neurone trivial avec une (01) couche cachée

Le poids attribué à une liaison (flèche noire) reflète la force et la consistance de cette dernière. Une matrice de poids aléatoires entre 0 et 1 est générée au début du processus d'apprentissage du réseau de neurone. Elle est de taille I×HL où I est le nombre de neurones en Input (03 dans notre cas) et HL (hidden layer), le nombre de neurones dans la couche cachée (04 dans notre cas). Cela explique la taille de notre matrice initiale qui est de 3×4.

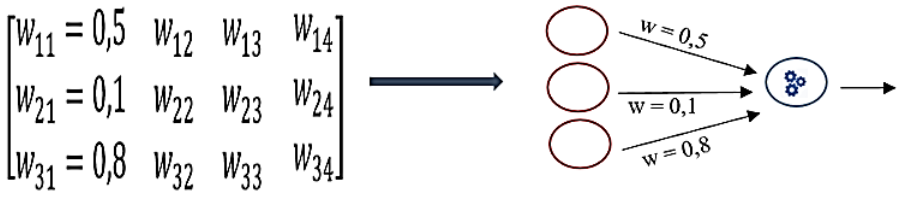


Figure 3.8: Prise en compte des poids dans le calcul neuronal

Cette attribution randomisée est un biais utile permettant au réseau de neurone de mieux apprendre des données et ajuster en conséquence les valeurs de notre matrice au fur et à mesure de son apprentissage. En effet, ce dernier laisse au neurone une marge de manœuvre pour modifier l'output indépendamment des entrées. Ce processus d'ajustement est appelé

propagation inverse (back-propagation). Lorsqu'un réseau de neurone est entrainé, il génère pour chaque unité d'entrainement une matrice d'erreur qui constitue un coût à minimiser. De manière itérative, on arrive donc à l'obtention des poids optimaux minimisant l'erreur.

En pratique, tous les neurones d'une même couche cachée sont dotés de la même fonction d'activation.

3.4.2 fonction d'activation

La fonction d'activation définit la sortie (output) du neurone à partir de données en entrée (input). Il en existe plusieurs catégories mais le fonctionnement général reste le même. Après la phase de jonction qui constitue la phase de pré-activation du neurone, on obtient un réel. Ce dernier est alors utilisé en argument d'une fonction d'activation dont le choix dépend de la nature de la sortie. Soit ϕ, cette fonction. Il en existe de plusieurs sortes et nous donnons ici quelques exemples.
La fonction identité est plus utilisée dans le cadre d'une regression. La fonction sign ne prend que 3 valeurs, ce qui impacte sur la qualité des prédictions. La sigmoïde est continue et donne une valeur qui peut être traduite en probabilité, cependant, elle est lourde en calculs et non centrée. La tanh a le leger avantage d'être centrée par rapport à la sigmoide. La tanh dure vient alléger les calculs par rapport à la tanh. La ReLU règle le problème de disparition du gradient. En effet, les poids du réseau neuronal sont mis à jour à chaque itération de l'appprentissage, de manière proportionnelle à la dérivée partielle de la fonction d'erreur par rapport au poids courant. Parfois, le gradient devient trop petit au bout d'un nombre faible d'itérations, empêchant le changement de la valeur du poids. [Glorot et al., 2011] La ReLU a toutefois montré un avantage remarquable en analyse du sentiment et en classification d'images. La leaky ReLU règle le problème du "ReLU mourant". Ce dernier consiste en ce que toutes les entrées négatives soient automatiquement annulées. La leaky ReLU randomisée, permet davantage que la leaky ReLU, d'éviter un surajustement sur les ensembles de données à petite échelle [Xu et al., 2015].

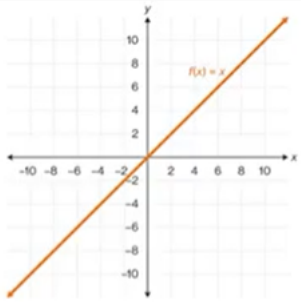
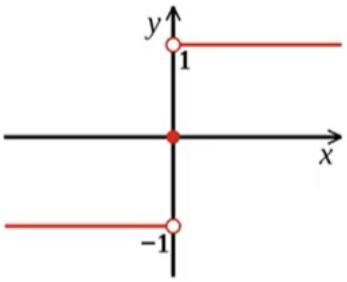
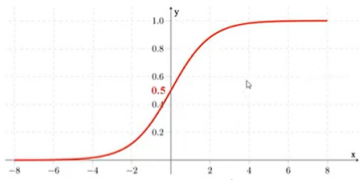
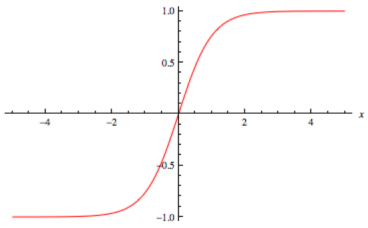
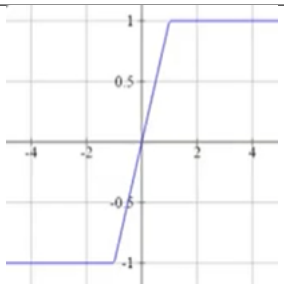
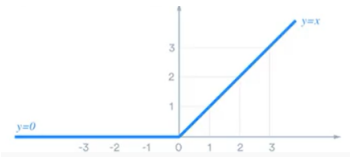
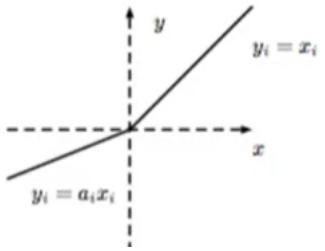
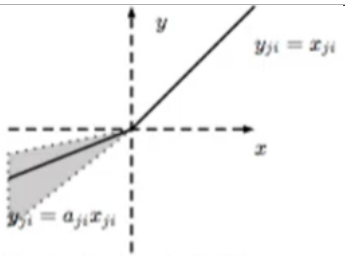
identité	**sign**	**sigmoïde**
		
$\phi(x) = x'.$	$\phi(x) = \begin{cases} -1 & \text{si} \quad x < 0 \\ 0 & \text{si} \quad x = 0 \\ 1 & \text{si} \quad x > 0 \end{cases}.$	$\phi(x) = \frac{1}{1+e^{-x}}.$
tanh	**tanh dure**	**relu**.
		
$\phi(x) = \frac{e^{2x}-1}{e^{2x}+1}.$	$\phi(x) = max\{min[x,1]\}.$	$\phi(x) = max\{x,0\}.$
Leaky ReLU	**Leaky ReLU randomisée**	
		
$\phi(x) = \begin{cases} 0,01x & \text{si} \quad x < 0 \\ x & \text{si} \quad x \geq 0 \end{cases}.$	$\phi(x) = \begin{cases} \alpha x & \text{si} \quad x < 0 \\ x & \text{si} \quad x \geq 0 \end{cases}.$	

Table 3.6: Exemples de fonctions d'activation

3.5 Support Vector Machine (SVM)

3.5.1 Principe

Avant que l'architecture du réseau neuronal ne devienne populaire, une approche courante a été et demeure encore aujourd'hui les SVM (pouvant se traduire par "séparateur à vaste marge"). Il s'agit d'une méthode de classification supervisée qui se focalise sur la séparabilité optimale des données. Or, la plupart du temps, les données ne sont pas linéairement

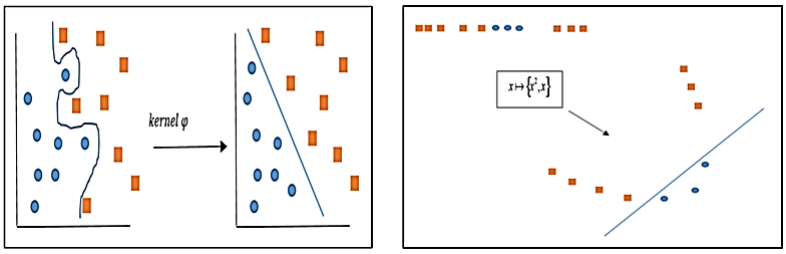


Figure 3.9: Exemple de transformation par une fonction

séparables. Une fonction de transformation appelée 'noyau' est donc utilisée afin qu'elles le soient.

3.5.2 Choix de la fonction noyau

Le noyau peut être linéaire ou non. Soit $K : A \mapsto B$ une fonction noyau. K est linéaire si pour tout u, v appartenant à A et g appartenant à $\mathbb{R}$:

$$K(u+v) = K(u) + K(v) \text{ et } K(gu) = gK(u).$$

La condition nécessaire et suffisante pour qu'une fonction soit un noyau est qu'elle soit symétrique (invariable malgré la permutation de ses variables) et définie positive. Les fonctions noyau les plus utilisées peuvent être regroupées en trois (03) principales catégories.

Remarquons que les fonctions sigmoïde et tangente hyperbolique forment une et même famille. Comme vu plus haut, elles sont souvent utilisées dans le fonctionnement des réseaux de neurones en tant que fonction d'activation.

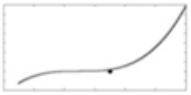
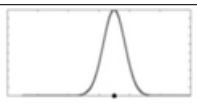
Polynomiales	de base radiale	De type sigmoïde
		
$K(i,i') = (1+\sum_j i_j i'_j)^d$.	$K(i,i') = exp(\frac{-(i-i')^2}{2\sigma^2})$.	$K(i,i') = tanh(ci^t i' + h)$.

Table 3.7: Trois (03) principales familles de noyaux

Après transformation, nous sommes emmenés à un nouveau mappage de données qui peut occasionner certaines lenteurs computationnelles. C'est là qu'intervient « l'astuce du noyau ».

3.5.3 L'astuce du noyau

Elle s'applique à tout algorithme pouvant s'exprimer sous forme d'une expression basée sur des produits scalaires entre des vecteurs observations d'entrée. Elle suppose qu'on peut calculer le produit scalaire $\langle \phi(i), \phi(i') \rangle$ directement sans jamais avoir à calculer explicitement $\phi(i)$ ou $\phi(i')$. On choisit alors le noyau ou kernel K tel que $K(i,i') = \langle \phi(i), \phi(i') \rangle$. Par exemple, soit : $\varphi : (i_{[1]}, i_{[2]}) \mapsto (i^2_{[1]}, \sqrt{2} i_{[1]} i_{[2]}, i^2_{[2]})$. Alors, on a:

$$
\begin{aligned}
K(i,i') &= \langle \varphi(i), \varphi(i') \rangle \\
&= \left\langle (i^2_{[1]}, \sqrt{2} i_{[1]} i_{[2]}, i^2_{[2]}), (i'^2_{[1]} - \sqrt{2} i'_{[1]} i'_{[2]}, i'^2_{[2]}) \right\rangle \\
&= i^2_{[1]} i'^2_{[1]} + \sqrt{2} i_{[1]} i_{[2]} \sqrt{2} i'_{[1]} i'_{[2]} + i^2_{[2]} i'^2_{[2]} \\
&= i^2_{[1]} i'^2_{[1]} + 2 i_{[1]} i_{[2]} i'_{[1]} i'_{[2]} + i^2_{[2]} i'^2_{[2]} \\
&= (i_{[1]} i'_{[1]} + i_{[2]} i'_{[2]})^2 \\
&= (\langle i, i' \rangle)^2.
\end{aligned}
$$

3.5.4 Maximisation de la marge de séparation

L'objectif est de trouver l'autoroute de séparation offrant une marge maximale (d sur la figure 3.10). Le vecteur $\vec{w}$ se veut perpendiculaire à la zone franche et un individu est classé dans la classe où se situe son projeté sur $\vec{w}$. Par convenance mathématique, nous considérerons

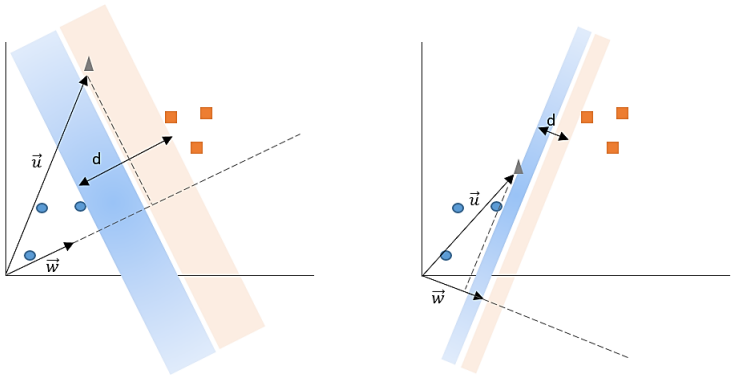


Figure 3.10: Recherche de l'autoroute de séparation optimale

tout point comme faisant partie d'un ensemble X = $\{x_i, i\epsilon\mathbb{N}\}$. Selon que x_i soit un rond où un carré, nous admettons que $\overline{w}.\overline{u} + \leq -1\ ou \geq 1$. Soit Y = $\{y_i, i\epsilon\mathbb{N}\}$ tel que $y_i = 1$ ou -1 si x_i est respectivement un carré ou un rond. On a : $y_i(x_i\overline{w} + b) \geq 1$. Dans le cas où x_i est sur le bord de notre zone franche alors : $y_i(\overline{x_i w} + b) = 1$. Nous recherchons la marge maximal d.

$$d = (\overline{x}_c - \overline{x}_r)\frac{\overline{w}}{\|\overline{w}\|} = \frac{1}{\|\overline{w}\|}(\overline{x}_c\overline{w} - \overline{x}_r\overline{w}) = \frac{1}{\|\overline{w}\|}((1-b) - (-1-b)) = \frac{2}{\|\overline{w}\|}.$$

Maximiser d reviens donc à minimiser $\frac{1}{2}\|\overline{w}\|^2$.

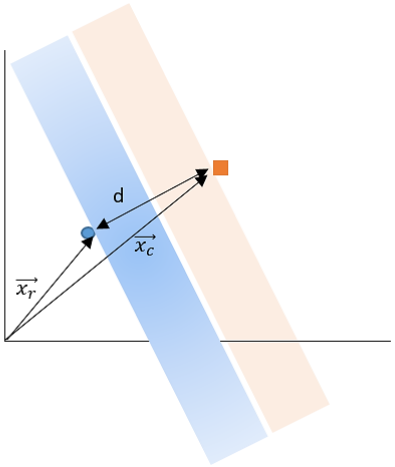


Figure 3.11: Valeur de d

Pour cela, nous utilisons le multiplicateur de Lagrange.

$$L = \frac{1}{2}\|\overline{w}\|^2 - \sum \alpha_i[y_i(\overline{x}_i\overline{w} + b) - 1]$$

$$\frac{\partial L}{\partial x} = \overline{w} \sum \alpha_i [y_i \overline{x}_i] = 0 \;\Rightarrow\; \boldsymbol{\overline{w} = \sum \alpha_i y_i \overline{x}_i}$$

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \;\Rightarrow\; \boldsymbol{\sum \alpha_i y_i = 0}.$$

Notre multiplicateur devient:

$$L = \frac{1}{2}(\sum \alpha_i y_i \overline{x}_i)(\sum (\alpha_j y_j \overline{x}_j) - \sum \alpha_i [y_i (\overline{x}_i \sum \alpha_j y_j \overline{x}_j) + b) - 1]$$

$$L = \frac{1}{2}(\sum \alpha_i y_i \overline{x}_i)(\sum \alpha_j y_j \overline{x}_j) - (\sum \alpha_i y_i \overline{x}_i)(\sum \alpha_j y_j \overline{x}_j) - b \sum \alpha_i y_i + \sum \alpha_i$$

$$L = \sum \alpha_i - \frac{1}{2}(\sum \alpha_i y_i \overline{x}_i)(\sum \alpha_j y_j \overline{x}_j)$$

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j (\alpha_i \alpha_j y_i y_j \boldsymbol{\overline{x}_i . \overline{x}_j}).$$

L'intérêt est de découvrir que l'optimisation de notre Lagrangien dépend uniquement du produit scalaire $\overline{x}_i . \overline{x}_j$ (ou encore $\langle \overline{x}_i, \overline{x}_j \rangle$) de nos points.

3.5.5 Avantages et inconvénients

Le principe de la méthode est simple et facile à comprendre. Basé sur un fondement théorique solide, le SVM est particulièrement adapté à des bases de données avec un nombre de variables largement supérieur au nombre d'observations. C'est l'une des raisons pour lesquelles nous avons choisi de l'utiliser, c'est d'ailleurs, le seul algorithme de classification supervisée que nous utiliserons dans notre étude.

Bien que ne nécessitant que très peu de ressources computationnelles, le SVM en sa forme classique ne sied pas au données avec un relativement trop grand nombre d'individus [Cervantes et al., 2007]. Ses performances sont amoindries lorsque les classes se confondent.

3.6 Self-organizing map (réseau Kohonen)

3.6.1 Couche de Kohonen

Le SOM est un réseau de neurone très particulier aussi bien dans son architecture que dans son algorithme. Utilisé dans une contexte d'apprentissage non supervisé (classes d'appartenance des individus non connues), elle n'apprend pas par correction de l'erreur comme vu plus haut mais plutôt par compétition. SOM est utilisé pour réduire et cartographier les données sur des dimensions inférieures.

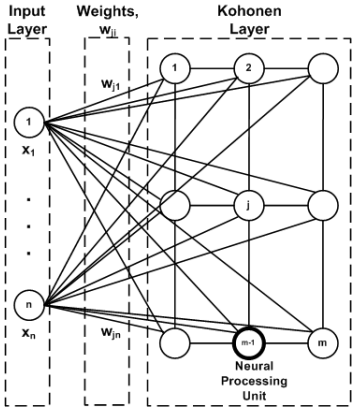


Figure 3.12: Architecture SOM

Avant le début de l'apprentissage, la couche de Kohonen est appelée carte de départ. Elle peut revêtir plusieurs formes dont la plus utilisée est celle en deux dimensions.

Pour chaque entrée (observation), les neurones de la couche de Kohonen compètent afin de savoir qui devra s'activer pour répondre au vecteur d'entrée. Le nœud (neurone) est choisi en fonction de sa similarité avec le vecteur. Ce choix s'accompagne de la sélection d'un certain voisinage. Ce qui veut dire que, de la proximité entre nœuds, nous pouvons fortement décrire une proximité entre vecteurs associés.

Soient $(\mu_1^t, \ldots, \mu_N^t) \epsilon (R^n)^N$, l'ensemble des neurones de notre couche de Kohonen, et $(X_1, \ldots, X_N) \epsilon (R^n)^K$, nos observations. Nous notons $V(\mu_j)$, l'ensemble des nœuds voisins de μ_j. Utilisons une suite de réelles positif α_t telle que $\sum_{t>0} \alpha_{t^2} > \infty$ et $\sum_{t>0} \alpha_t = \infty$. L'algo-

rithme du Self organizing map suivant, est inspiré des travaux de synthèse du professeur Xavier Dupré (ENSAE-paris).

Initialisation

Les neurones $(\mu_1^0, \ldots, \mu_N^0)\ \epsilon\ (\mathbb{R}^n)^N$ adoptent une répartition régulière

Choix du neurone le plus proche

On choisit aléatoirement une observation X_i avec $1 \leq i \leq N$. Le neurone gagnant μ_{k*}^t est celui satisfaisant :

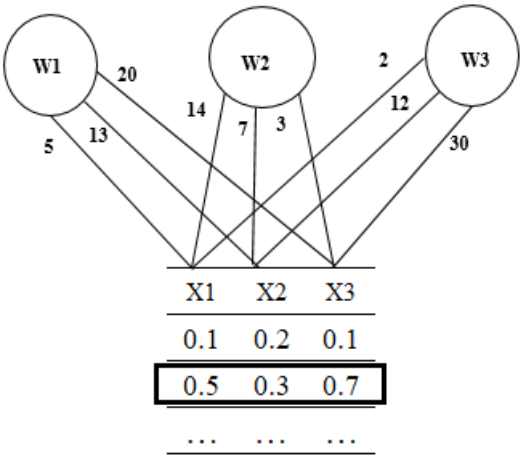
$$\left\|\mu_{k*}^t - X_i\right\| = min_{1 \leq i \leq N} \left\|\mu_j^t - X_i\right\|.$$

Mis à jour

Pour chaque $_j^t$ dans le voisinage de μ_{k*}^t, V(μ_{k*}^t) et cela jusqu'à convergence :

$$\mu_j^{(t+1)} \leftarrow \mu_j^t + \alpha_t(X_i - \mu_j^t).$$

Considérons l'exemple suivant avec une couche topologique à 3 neurones. Pour une observation quelconque, nous voulons illustrer comment la compétition entre neurones se déroule. La deuxième observation sera utilisée pour les besoins de l'exercice. Déterminons le neurone gagnant pour cette entrée.



Il nous faudra calculer la distance euclidienne entre l'entrée et chacun des trois (03)

neurones en compétition.

$$d_1 = \sqrt{\sum_i^3 (x_i - w_{1,i})^2} = \sqrt{(0.5-5)^2 + (0.3-13)^2 + (0.7-20)^2} = 23,5.$$

$$d_2 = \sqrt{\sum_i^3 (x_i - w_{2,i})^2} = \sqrt{(0.5-14)^2 + (0.3-7)^2 + (0.7-3)^2} = 15,2.$$

$$d_3 = \sqrt{\sum_i^3 (x_i - w_{3,i})^2} = \sqrt{(0.5-2)^2 + (0.3-12)^2 + (0.7-30)^2} = 31,6.$$

La plus distance euclidienne calculée est celle avec le neurone 2. Pour l'entrée sélectionnée, il remporte alors la compétition pour cette itération.

3.6.2 Avantages et inconvénients

Un avantage notoire du SOM est la non nécessité de définir le système neuronal de manière spécifique (pas d'apprentissage par exemple). Aussi, il offre une panoplie de possibilité de visualisation des données. Pour finir, il sied de par son architecture à une implémentation en calculs parallèles et convient donc à l'utilisation de données de grands volumes.
Comme inconvénients, les résultats ne sont pas stables d'une exécution à une autre et l'efficacité de la méthode est intimement liée à la métrique choisie.

CHAPITRE 4: Méthodologie

Nous présentons dans ce chapitre, les étapes essentielles nécessaires à la compréhension de notre futur phase d'implémentation.

4.1 Schémas méthodologique

La figure 4.2 est un aperçu graphique du cheminement méthodologique suivi lors de la phase d'implémentation. Nous donnons ci-après un aperçu de ce en quoi consiste chaque étape.

Chargement et préparation du texte : Une fois le corpus (ensemble de textes ou documents)

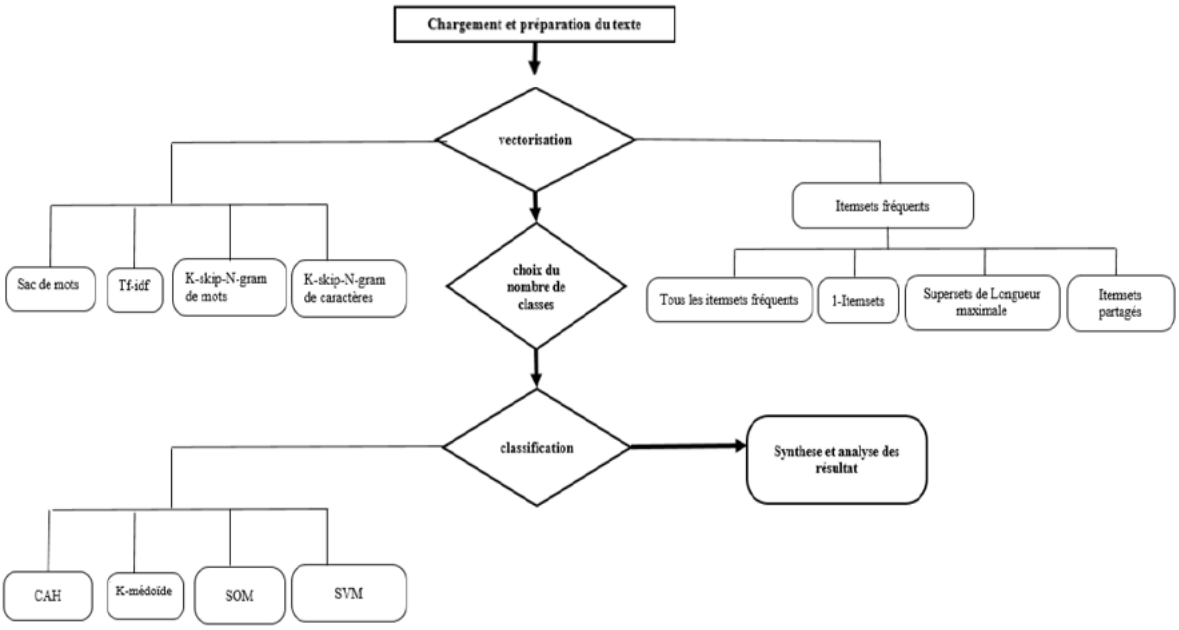


Figure 4.1: Schémas méthodologique

chargé dans l'interface (que nous présentons plus bas), un traitement consistant principalement aux opérations de tokenisation et de suppression des stop-words est effectué.

Vectorisation : Contrairement au sac de mots et au Tf-idf, Le K-Skip-N-gram et l'itemset fréquent font appel à certains paramètres. Pour le premier, il faut fixer une valeur de K et de N, pour le second, il faut choisir 'arbitrairement' la valeur du support minimum (où en tester

plusieurs). Une fois nos descripteurs (méthodes de vectorisation) spécifiés, ils sont appliqués aux textes du corpus. Chaque texte étant maintenant représenté par un vecteur selon le descripteur. Le corpus est maintenant résumé en une base de donnée de type tableur, résultat de la concaténation des vecteurs de textes et avec autant de ligne qu'il y a de textes distincts dans le corpus.

Choix du nombre de classes : Une fois la base de donnée constituée, nous y appliquons deux indices (les plus courants) que nous présenterons plus bas, en faisant varier le nombre de classes jusqu'au tiers du nombre d'observations ou de lignes, ceci par considération de trois (03) algorithmes de classification non supervisée (SOM, K-MEDOID 'PAM' et CAH). Le nombre de classes ayant été sélectionné le plus de fois (vainqueur du vote) est alors choisi. Nous gardons en cas d'égalité, un deuxième choix de nombre de classes.

Synthèse et analyse des résultats : Vu le nombre important d'expérimentations qu'offre cette étude, nous effectuons des synthèses partielles. Par exemple, les itemsets fréquents sont exploités de différentes manières 'sous-descripteurs' (tous les itemsets, les 1-itemsets, les supersets de longueur maximale et les itemsets fréquents partagés). Les résultats issus de ces quatre sous-expérimentations sont comparés puis nous choisissons le meilleur des sous-descripteurs. Aussi, en faisant varier le K et N des K-skip-N-gram, nous faisons un récapitulatif et choisissons la meilleure combinaison (N*, K*) pour les mots et (N**, K**) pour les caractères. Dans la synthèse finale, sont analysés et comparés les résultats des descripteurs suivant :

- Le sac de mots ;
- Le Tf-idf ;
- Le meilleur sous-descripteur parmi les itemsets fréquents ;
- Le K*-skip-N*-gram de mots ;
- Le K**-skip-N**-gram de caractères.

4.2 Traitement de la donnée textuelle

Nous illustrons dans cette partie différentes facettes de la tokenisation mais aussi de la subtilité qu'il peut y avoir à choisir de supprimer ou non les stop-words (mots fonctionnels).

4.2.1 Tokenisation

La tokenisation est un processus permettant de segmenter le texte en tokens. Il s'appuie traditionnellement sur des délimiteurs (l'espace pour la plupart des langues européennes par exemple). Partant de cette optique, le token pourrait naïvement être associer au mot. Cependant, la définition est toute autre compte tenu de différents aspects (spécificité linguistique, complexité orthographique, …) et angle d'analyse (lexicographique, grammatical).
Une approximation jugée pertinente du token pourrait être le lemme. Il s'agit d'une « unité autonome constitué d'un sens et d'une forme ». La locution nominale (fer à repasser) et la locution adjective (à pic) sont des lemmes constitués de mots dépendants, formant des blocs insécables pour garder leur sens. Une variante du lemme est le morphème se différenciant par son caractère unitaire. Il en existe 2 types : les unités lexicales (lexèmes) et unités grammaticales (grammèmes).

lexème		**grammème**	
Indépendant (mot racine)	Dépendant (radical)	Indépendant (affixe)	Dépendant (art. pron. prép. conj. adverbes inclassables,…)
Exemple			
jardin	Jardin-	Jardiner ; jardinage	…
continu	Continu-	continuation	Continûment

Table 4.1: Morphème dépendant ou indépendant

L'approche de base de la tokenisation, s'appuyant sur l'espace, n'est pas toujours appropriée. C'est le cas notamment avec les locutions. 'Une locution est un groupe de mots qui forme une unité lexicale. Autrement dit, c'est un adverbe, un verbe, une préposition, une conjonction ou une interjection composé(e) de deux ou de plusieurs mots'. Exemple : chemin de fer, mise en jeu, comme il faut, quelque chose, avoir lieu, en vain, …
Face à la complexité de certaines langues comme l'arabe par exemple ou le mandarin, de plus en plus de chercheurs utilisent les Word embeddings afin de traduire les similarités morphologiques et syntaxiques.

Les mots ci-dessus se rapportent à un seul lemme qui est:

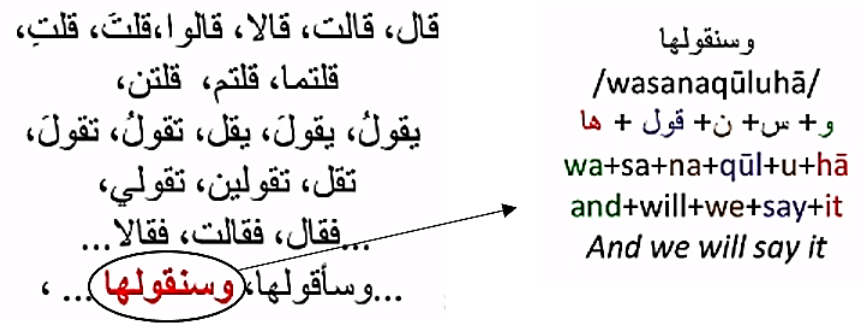


Figure 4.2: Exemple de formes dérivées relatives à un seul lemme en arabe

قُو ل

Les verbes en arabe peuvent se mettre sous 5400 formes différentes. Comparé à l'anglais qui enregistre 6 formes, il s'agit d'une pléthore. Plusieurs types d'éléments viennent ainsi se greffer à la racine : le sujet, le genre, le nombre, le complément, etc.
Afin de saisir certaines subtilités du langage, une alternative consiste à l'utilisation d'un lexique de référence (dictionnaire). Le dictionnaire est parcouru selon deux (02) approches possibles [Rehman et al., 2013]:

- Le **minimum matching segmentation (MinMatch)** cherche les correspondances en unités de mots. Pour le groupe de mots « l'ensemble », par exemple, les tokens seront « le » et « ensemble » tandis que pour « pomme de terre » les tokens seront « pomme », « de » et « terre ».
- Le **maximum matching segmentation (MaxMatch)** quant à lui, cherche les correspondances en partant des groupes de mots les plus longs sémantiquement validés par le dictionnaire. Pour le groupe de mots « l'ensemble », par exemple, les tokens seront « le » et « ensemble » tandis que pour « pomme de terre » le token sera « pomme de terre ».

Les tokens sont exploités dans leur forme la plus générale en maximisant la fiabilité sémantique par rapport au texte d'origine. Les lexèmes indépendants, les locutions nominale et verbale sont ainsi privilégiés. Les affixes eux, disparaissent par un processus de stemming pour garder le radical puis la lemmatisation (infinitif, singulier, …), permet de remplacer ce dernier par le mot racine assurant ainsi la validité sémantique.

4.2.2 Stop-word

Le tableau suivant contient en guise d'exemple, des 2-skip-2-gram de mots. Intéressons-nous de près aux éléments en gras. Il s'agit de combinaisons qui sont non porteuses de sens. Or, le but de cette association est de pouvoir identifier les combinaisons les plus pertinentes quant à la catégorisation de notre texte et à la distinction de ce dernier par rapport aux autres. Les mots

Nos expérimentations	**Nos ont**	**Nos confirmés**	**expérimentations ont**
expérimentations confirmé	**expérimentations l'**	ont confirmé	**ont l'**
ont importance	**confirmé l'**	confirmé importance	**confirmé des**
l'importance	**L' des**	**L' règles**	**importance des**
importance règles	**importance d'**	Des règles	**Des d'**
Des association	**Règles d'**	Règles association	**Règles pour**
D'association	**D' pour**	**Association pour**	**Association des**
Association fins	**Pour des**	Pour fins	**Pour d'**
Des fins	**Fins d'**	Fins extraction	D'extraction
D'de	**D' l'**	Extraction de	Extraction l'
Extraction information	**De l'**	**De information**	De pertinente
L'information	**L' pertinente**	**L' et**	Information pertinente
Information et	Information de	**Pertinente et**	Pertinente de
Pertinente qualité	**Et de**	**Et qualité**	**Et que**
De qualité	**De que**	**De l'**	Qualité que
Qualité l'	**Qualité on**	**Que l'**	**Que on**
Que peut	**L'ont**	**L'peut**	**L'retrouver**
On peut	On retrouver	**On dans**	Peut retrouver
Peut dans	**Peut les**	Retrouver dans	**Retrouver les**
Retrouver données	**Dans les**	Dans données	Données textuelles
Les données	Les textuelles	Données textuelles	

Table 4.2: Effet des Stop-words sur la pertinence des associations de mots

incontournables, présents dans la quasi-totalité des textes ne sont en général pas pris en compte : pronoms personnels, articles, auxiliaires être ou avoir, … Leur présence ajoute du flou, du bruit et n'apporte aucune plus-value significative dans la plupart du temps (compréhension du contexte par exemple). Il s'agit des stop-words (mots fonctionnels ou vides) et leur suppression fait partie des opérations usuelles de prétraitement de texte en traitement du langage naturel. En classification de documents, leur suppression n'affecte généralement pas les performances du modèle bien au contraire. Cependant, ce n'est pas une pratique qui devrait se faire systématiquement mais selon la nature de l'exercice. En « sentiment analysis » par exemple, plusieurs tests et précautions supplémentaires sont nécessaires pour ne pas altérer les résultats. Si de la

phrase: « Ma sœur n'est pas contente », nous ne retenons que « 'sœur', 'contente' » alors le sentiment négatif ne peut être capté. [Saif et al., 2014] Cela a d'ailleurs notamment été prouvé dans une étude sur l'effet de différents modes de traitement des mots vides sur la classification de la polarité des sentiments sur Twitter. Il en est ressorti que leur suppression au moyen d'une liste standard, avait un impact négatif sur les résultats.

4.3 Choix des paramètres N et K pour les K-skip-N-gram

Pour un texte donné, il y a autant de vectorisations possibles (K×N) que de combinaisons de N et de K. Afin de choisir ces deux (02) valeurs, nous faisons varier respectivement de 1 à 5 et de 2 à 5, les paramètres K et N. Le choix du couple (N, K) de mots ou de caractères dépendra fortement de l'appréciation relative des sorties graphiques des résultats:

- Lorsque les classes d'appartenance des textes sont connues alors le choix des paramètres est axé sur le fait de pouvoir retrouver ou non ces classes ;
- Dans le cas contraire, selon la méthode de classification, la validation visuelle consistera à :

- **Classification ascendante hiérarchique (dendrogramme) :** éviter le phénomène de chainage et observer des formations nettes de classes ;
- **K-médoïde (graphique des deux (02) premiers axes) :** vérifier que les classes formées sont homogènes et éviter qu'il y ait plusieurs classes avec un seul individu.
- **SOM (graphiques de mappage et de distance) :** S'assurer d'un minimum de concordance entre les regroupements d'individus (textes) au niveau du graphique de mappage (positionnement des textes) et les intensités de liaisons sur le graphe des distances (Proximité métrique entre textes de même nœud ou voisinage).

4.4 Spécification du vocabulaire pour les sous-descripteurs des itemsets

Les k-itemsets sont les itemsets fréquents de longueur k, c'est-à-dire, qui sont constitués de k items. Les supersets de longueur maximale sont les itemsets fréquents de longueur k tel que $k = \max(k)$. Les itemsets partagés sont les itemsets générés uniquement à partir des items partagés par 2 transactions au minimum. Les itemsets fréquents englobent donc les autres sous-descripteurs. Les 1-itemsets sont quant à eux des mots mais les mots ne sont pas tous des 1-itemsets car ces derniers doivent pour cela respecter la contrainte du support minimal.

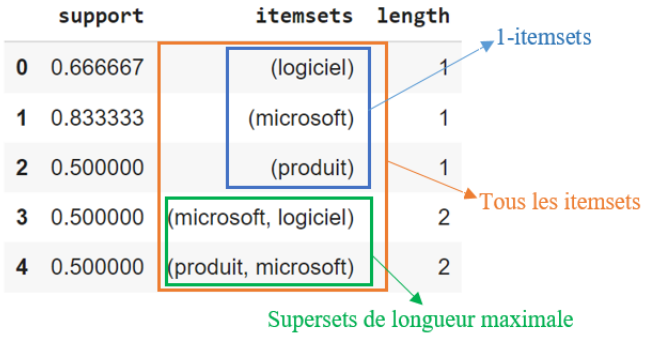


Figure 4.3: Itemsets vs 1-itemsets vs supersets de longueur maximale

4.5 Exemple (du texte au vecteur avec l'itemset fréquent)

Nous présentons dans cette partie le processus menant du texte à une base de type tableur en utilisant les itemsets fréquents. L'image suivante représente le texte qui nous servira de tremplin pour notre exemple :

Jordan a joué trois saisons pour l'entraîneur Dean Smith à l'Université de Caroline du Nord. En tant que étudiant de première année, il a été membre de l'équipe du championnat national des Tar Heels en 1982. Jordan a rejoint les Bulls en 1984 en tant que troisième choix au repêchage. Il est rapidement devenu une star de la Ligue et a diverti les foules avec son score prolifique. Ses capacités de saut, démontrées en effectuant des slam dunk de la ligne de lancers francs dans les Slam Dunk Contests, lui ont valu les surnoms Air Jordan et His Airness. Il a également acquis la réputation d'être l'un des meilleurs joueurs défensifs du basketball. [6] En 1991, il remporta son premier titre de champion de la NBA avec les Bulls et suivit cet exploit avec des titres en 1992 et 1993, obtenant ainsi un "trois tours". Bien que Jordan se soit brutalement retiré du basketball avant le début de la saison 1993-1994 de la NBA et commence une nouvelle carrière en jouant dans le baseball mineur, il est revenu aux Bulls en mars 1995 et les a menés à trois autres championnats en 1996, 1997 et 1998. Ainsi qu'un record de 72 victoires en saison régulière lors de la saison 1995-1996 de la NBA. Jordan a pris sa deuxième retraite en janvier 1999, mais est revenu pour deux saisons NBA supplémentaires de 2001 à 2003 en tant que membre des Wizards.

Figure 4.4: Texte exemple pour illustration d'une vectorisation

4.5.1 Traitement du texte

Ci-dessous nous pouvons voir ce même texte après segmentation en phrases. Cette opération de découpage en phrases est indispensable pour prendre les itemsets fréquents comme descripteurs de texte.

```
texte_1
```

```
['Jordan a joué trois saisons pour l entraîneur Dean Smith à l Université de Caroline du Nord.',
 'En tant que étudiant de première année, il a été membre de l équipe du championnat national des Tar Heels en .',
 'Jordan a rejoint les Bulls en  en tant que troisième choix au repêchage.',
 'Il est rapidement devenu une star de la Ligue et a diverti les foules avec son score prolifique.',
 'Ses capacités de saut, démontrées en effectuant des slam dunk de la ligne de lancers francs dans les Slam Dunk Contests, lui ont valu les s
 'Il a également acquis la réputation d être l un des meilleurs joueurs défensifs du basketball.',
 'En , il remporta son premier titre de champion de la NBA avec les Bulls et suivit cet exploit avec des titres en  et , obtenant ainsi un tr
 'Bien que Jordan se soit brutalement retiré du basketball avant le début de la saison  de la NBA et commence une nouvelle carrière en jouant
 'ainsi qu un record de  victoires en saison régulière lors de la saison  de la NBA.',
 'Jordan a pris sa deuxième retraite en janvier , mais est revenu pour deux saisons NBA supplémentaires de  à  en tant que membre des Wizards
```

Figure 4.5: Segmentation du texte en phrases

En effet, chaque phrase est considérée, après traitement comme une transaction participant à la description du texte qui la contient. Après un découpage de chaque phrase en token. Nous comparons chaque token formé à une liste résultant de la concaténation de celles de la ponctuation et des stop-words de la langue française. Cette liste est agrémentée de divers autres composantes afin d'améliorer l'apurement :

,’,’a’,’et’,’entre’,’qu’,’b’,’c’,’d’,’e’,’f’,’g’,’h’,’i’,’j’,’k’,’l’,’m’,’n’,’o’,’p’,’q’,’r’,’s’,’t’,’u’, ’v’,’w’,’x’,’y’,’z’, ’.’,’1’,’2’,’3’,’4’,’5’, ’6’,’7’,’8’,’9’,’0’,’un’,’tant’,’ainsi’,’alors’,’puisque’, ’donc’, ’ou’, ’si’, ’lors’

Si le token ne fait pas partie de cette liste alors il est gardé, dans le cas contraire, il est supprimé. Les dates ainsi que les chiffres entre crochets et les lettres isolées disparaissent notamment. Ci-après, ce même texte après traitement :

[[’Jordan’, ’joué’, ’saisons’, ’entraîneur’, ’Dean’, ’Smith’, ’Université’, ’Caroline’, ’Nord’],
[’étudiant’, ’première’, ’année’, ’membre’, ’équipe’, ’championnat’, ’national’, ’Tar’, ’Heels’**]**,
[’Jordan’, ’rejoint’, ’Bulls’, ’troisième’, ’choix’, ’repêchage’**]**,
[’rapidement’, ’devenu’, ’star’, ’Ligue’, ’diverti’, ’foules’, ’score’, ’prolifique’**]**,
[’capacités’, ’saut,’, ’démontrées’, ’effectuant’, ’slam’, ’dunk’, ’ligne’, ’lancer’, ’franc’, ’Slam’, ’Dunk’, ’Contests’, ’valu’, ’surnoms’, ’Air’, ’Jordan’, ’Airness’**]**,
[’également’, ’acquis’, ’réputation’, ’être’, ’meilleurs’, ’joueurs’, ’défensifs’, ’basketball’**]**,
[’remporta’, ’premier’, ’titre’, ’champion’, ’NBA’, ’Bulls’, ’suivit’, ’cet’, ’exploit’, ’titre’, ’obtenant’, ’tour’**]**,
[’Jordan’, ’brutalement’, ’retiré’, ’basketball’, ’début’, ’saison’, ’NBA’, ’commence’, ’nouvelle’, ’carrière’, ’jouant’, ’baseball’, ’mineur’, ’revenu’, ’Bulls’, ’mar’, ’menés’, ’autres’, ’championnats’**]**,
[’record’, ’victoires’, ’saison’, ’régulière’, ’saison’, ’NBA’**]**,
[’Jordan’, ’pris’, ’deuxième’, ’retraite’, ’janvier’, ’revenu’, ’saisons’, ’NBA’, ’supplémentaires’, ’membre’, ’Wizards’**]]**

4.5.2 Itemsets fréquents et représentation vectorielle du texte

Les phrases du texte sont considérées comme des transactions. Elles sont alors représentées sous forme d’un ensemble d’items (tokens).

	TEXTE 1
Phrase 1	[[Jordan], [joué], [saisons], [entraîneur], [Dean], [Smith], [Université], [Caroline], [Nord]]
Phrase 2	[[étudiant], [première], [année], [membre], [équipe], [championnat], [national], [Tar], [Heels]]
Phrase 3	[[Jordan], [rejoint], [Bulls], [troisième], [choix], [repêchage]]
Phrase 4	[[rapidement], [devenu], [star], [Ligue], [diverti], [foules], [score], [prolifique]]
Phrase 5	[[capacités], [saut], [démontrées], [effectuant], [slam], [dunk], [ligne], [lancer], [franc], [Slam], [Dunk], [Contests], [valu], [surnoms], [Air], [Jordan], [Airness]]
Phrase 6	[[également], [acquis], [réputation], [être], [meilleurs], [joueurs], [défensifs], [basketball]]
Phrase 7	[[remporta], [premier], [titre], [champion], [NBA], [Bulls], [suivit], [cet], [exploit], [titre], [obtenant], [tour]]
Phrase 8	[[Jordan], [brutalement], [retiré], [basketball], [début], [saison], [NBA], [commence], [nouvelle], [carrière], [jouant], [baseball], [mineur], [revenu','Bulls','mar'], [menés], [autres], [championnats]]
Phrase 9	[[record], [victoires], [saison], [régulière], [saison], [NBA]]
Phrase 10	[[Jordan], [pris], [deuxième], [retraite], [janvier], [revenu], [saisons], [NBA], [supplémentaires], [membre], [Wizards]]

Table 4.3: Modélisation du texte sous forme de suite de transactions

L'input de l'algorithme Apriori est un tableau de booléens avec autant de colonnes que d'items distincts et de lignes que de nombre de transactions.

	Airness	Bulls	Contests	Dunk	Ligue	Slam	Smith	Wizards	acquérir	air	année	baseball	basketball	...
0	False	False	False	False	False	False	True	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	True	False	False	
2	False	True	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	True	False	False	False	False	False	False	False	False	
4	True	False	True	True	False	True	False	False	False	True	False	False	False	
5	False	False	False	False	False	False	False	False	True	False	False	False	True	
6	False	True	False	False	False	False	False	False	False	False	False	False	False	
7	False	True	False	False	False	False	False	False	False	False	False	True	True	
8	False	False	False	False	False	False	False	False	False	False	False	False	False	
9	False	False	False	False	False	False	False	True	False	False	False	False	False	

En appliquant l'algorithme Apriori sur nos 10 transactions avec un support minimum de 0,5, le résultat est le suivant:

	support	itemsets
0	0.5	(jordan)

Figure 4.6: Itemsets fréquents du texte avec un support de 0,5

4.6 Concaténation des vecteurs

Une fois que chaque texte du corpus est représenté par un vecteur, tous les vecteurs sont concaténés afin de former une base avec autant de ligne que de textes dans le corpus. Les colonnes ne contiennent aucun doublon même si les textes ont des descripteurs en commun.

	descr1	descr2	descr3	descr4
texte 1	1	1	1	1

	descr1	descr5	descr2	descr4
texte 2	1	1	1	1

→

	descr1	descr2	descr3	descr4	descr5
texte 1	1	1	1	1	NaN
texte 2	1	1	Nan	1	1

Figure 4.7: Concaténation des vecteurs formés à partir des textes

Les NaN sont remplacés par 0 et toute valeur non nulle, par 1. Une matrice binaire est ainsi formée.

4.7 Normalisation de la base formée

Nous utilisons notamment la CAH dans l'une de ses versions de base. Cela implique que les données binaires pourraient avoir tendance à produire des résultats arbitraires et sensibles à l'ordre des observations. Les bases seront donc normalisées. Soit Normalizer () et StandardScaler (), les fonctions servant à normaliser respectivement respectivement la ligne et la colonne.

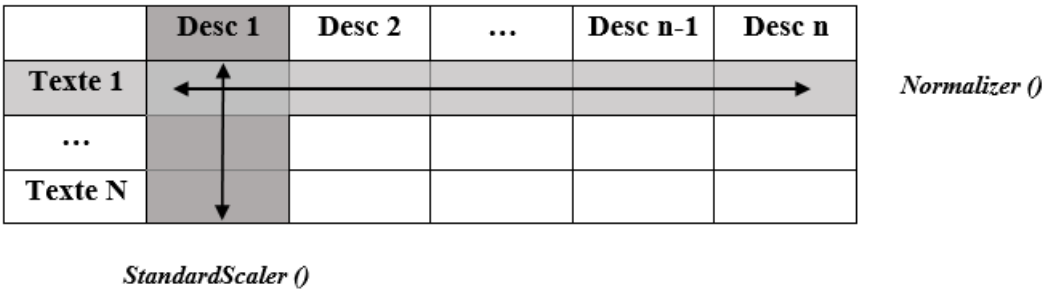


Figure 4.8: Différence entre StandardScaler() et Normalizer()

Pour les besoins de l'exemple, nous avons utilisé 4 textes relatifs à la NBA et 4 autres à Mozart. Ci-dessous, la figure 4.9 illustre l'impact du type de normalisation sur les résultats obtenus. StandardScaler() aurait tendance à faire perdre aux observations leurs spécificités. Normaliser selon les lignes semble être donc plus adapté à la nature de l'exercice.

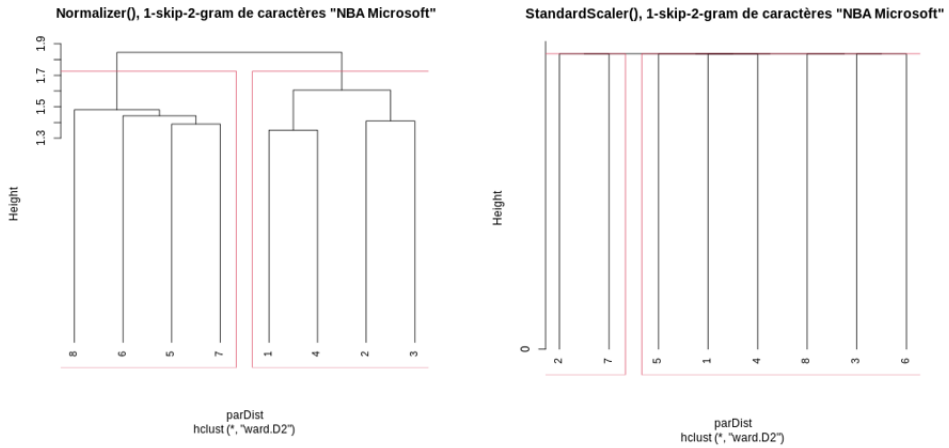


Figure 4.9: Impact sur une CAH de l'utilisation de StandardScaler() VS Normalizer()

4.8 Choix du nombre de classes

Nous utilisons les indices de Dunn et de Silhouette qui font partie des plus utilisés. La figure suivante montre le nombre de résultats de recherche sur Google associé à un échantillon de huit (08) indices.

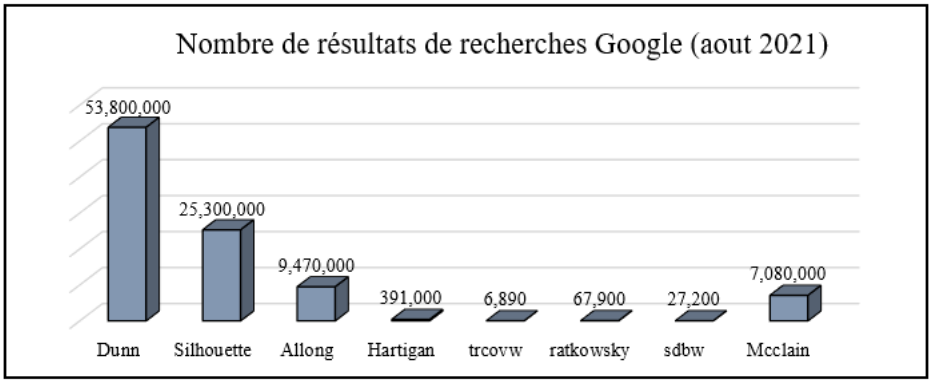


Figure 4.10: Popularité des indices Dunn et Silhouette

Soit A et B, deux (02) classes. On note $a_{i(i\epsilon\mathbb{N})}$ et $b_{i(i\epsilon\mathbb{N})}$, les éléments respectifs de A et de B. L'indice de Dunn D est le rapport de la plus petite distance entre les observations

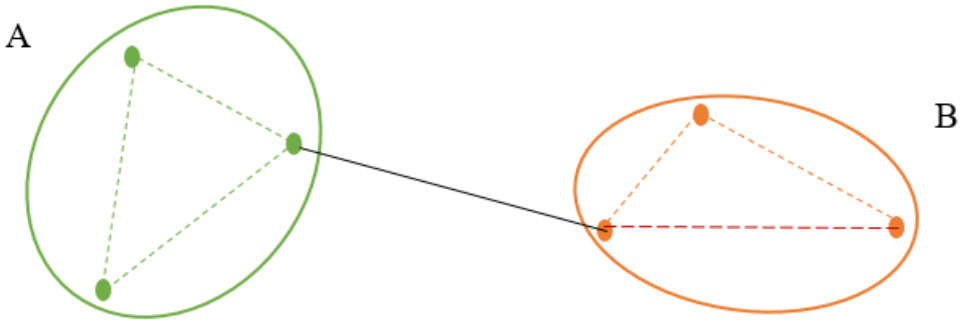


Figure 4.11: Plus petite distance entre les classes A et B

qui ne sont pas dans la même classe (segment noir sur la figure 4.8) à la plus grande distance intra-classe. L'indice de Dunn a une valeur comprise entre zéro et l'infini. Plus elle est grande, plus la qualité de la représentation est jugée meilleure.

$$D = \frac{\min_{i,j\epsilon\mathbb{N}}.dist(a_i, b_i)}{\max_{i,j\epsilon\mathbb{N},i\neq j}(dist(a_i, a_j), dist(b_i, b_j))}$$

L'indice de silhouette S, utilise la valeur silhouette de chaque observation. Soit y une observation.

$$s(y) = \frac{(Coh(y) - Sep(y))}{\max(Coh(y), Sep(y))}$$

Où :

Coh(y) est la cohésion de y, il s'agit de la distance moyenne de y avec les observations de la

même classe. Sep(y) est la séparation, il s'agit du minimum des distances moyennes entre y et les observations des autres classes. Soit N, le nombre total d'observations. Le coefficient silhouette S correspondant aux classes formées est :

$$S = \frac{1}{N} \sum_{1}^{N} s(x)$$

Dans notre travail, nous utilisons les indices de Dunn et Silhouette et procédons par vote pour le choix d'un même nombre de classes pour tout algorithme de classification confondu. Comme le montre l'exemple sur la figure 4.12, le vote conduit ici au choix de trois (03) classes.

		Nombre de classes	
		2	**3**
CAH			
	Dunn	**0,7**	0,6
	Silhouette	**0,2**	0,1
K-medoïde			
	Dunn	0,5	**0,8**
	Silhouette	0,3	**0,8**
SOM			
	Dunn	0,2	**0,4**
	Silhouette	0,1	**0,2**

→

	Nombre de classes	
	2	**3**
Résultats du vote	2	4

Figure 4.12: Vote et choix du nombre de classe

CHAPITRE 5: Expérimentations et Résultats

Nous expérimentons dans cette partie sept (07) descripteurs et sous-descripteurs de texte : le K-skip-N-gram (de mots, de caractères), le sac de mots ‘ bag of word (BOW)’, le TF-IDF et l’Itemset fréquent (tous les itemsets, les 1-itemsets, les supersets de longueur maximale et les itemsets partagés).

5.1 Language et librairies

5.1.1 Interface de travail et langage

Nos expérimentations ont été effectuées sur l’interface en ligne « Google Colab ». Elle offre un environnement convivial permettant entre autres, comme le montre la figure ci-dessous de charger des fichiers (1), d’ajouter des titres ou sous-titres (2), de masquer du code en gardant seulement en vue le titre chapeau du code (3), etc. Un autre avantage qu’offre cette

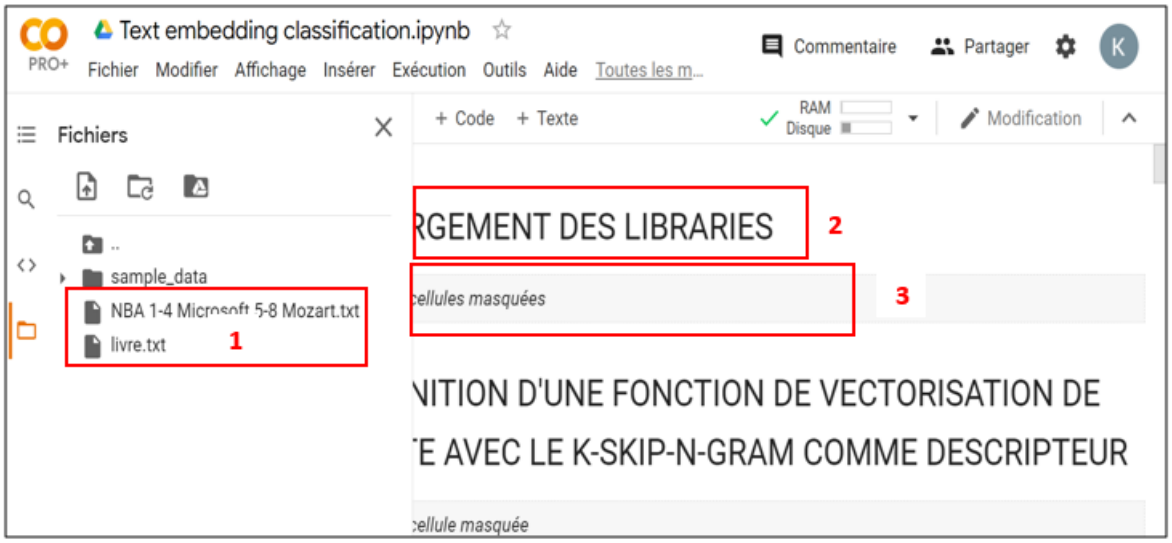


Figure 5.1: Interface Google colab

interface est la facilité d’alterner des cellules utilisant des langages différents dans l’accomplissement d’une même tache. Dans notre cas nous avons utilisé les langages R [R Core Team, 2013] et Python [Van Rossum and Drake Jr, 1995].

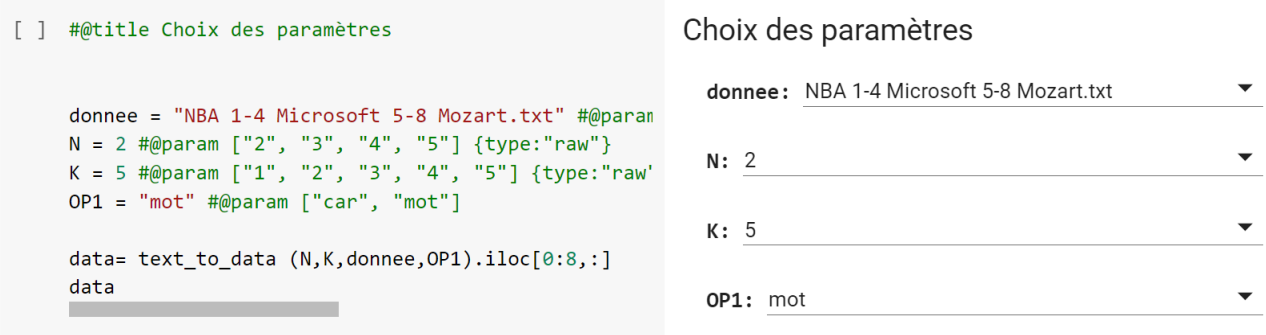


Figure 5.2: Exemple d'un cellule de code python

5.1.2 Les librairies NLTK et SPACY de Python

Pour le traitement et la vectorisation de nos données textuelles nous avons utilisé NLTK et SPACY qui sont des librairies spécifiques au traitement du langage naturel. Elles permettent le découpage et la manipulation de la donnée textuelle avec des fonctions taillées sur mesure. Par exemple, **fr-core-news-md.load()** permet de charger le modèle de langue pour le français et **nltk.download('stopwords')** permet de télécharger une liste assez fournie des stop-words pour le nettoyage du texte.

5.1.3 La librairie FUNCTOOLS de Python

Cette librairie est un incontournable, permettant un gain de temps énorme pour notamment certaines manipulations de structure. Considérons par exemple les listes l_1 = [[A], [B], [C]] et l_2 = [[D], [E]]. La fonction functools.reduce (add , l_1, l_2 , [ ]) permet par exemple d'obtenir la concaténation du contenu des liste et non des listes elles mêmes. En renvoit donc à une liste l_3 = [[A], [B], [C], [D], [E]]. Elle est très utile dans la construction de K-skip-N-grams.

5.1.4 La librairie SKLEARN de Python

Il s'agit d'une librairie permettant d'implémenter plusieurs algorithmes d'apprentissage machine. Nous l'utilisons notamment pour l'expérimentation avec le SVM. Elle contient des modules permettant de diviser des données en un ensemble d'entrainement et un autre de

test. Il est aussi possible d'utiliser des hyperparamètres. C'est l'exemple de la proportion des données d'entrainement par rapport à l'ensemble des données et cela, tout en garantissant une représentativité des différentes classes prédéfinies. Aussi, les fonctions Normalizer () et StandardScaler () du sous module 'Preprocessing' servent à normaliser selon respectivement la ligne et la colonne.

5.1.5 Les packages KOHONEN, CLUSTER ET FACTOEXTRA de R

Pour la classification, nous avons recours au langage R. les principales librairies utilisées sont Kohonen, cluster et factoextra. Elles seront respectivement utilisées pour le SOM, CAH et le k-médoïde.

5.2 Données utilisées

Nous utilisons pour nos expérimentations deux corpus, l'un de petite taille et avec des thématiques très distinctes puis l'autre de grande taille avec des thématiques assez liées. Les deux corpus ont été utilisés dans l'étude de [Bokhabrine et al., 2019], notre principale étude de référence citée en introduction. Le premier corpus a spécialement été conçu par l'auteur de ladite étude.

Corpus I

Ce corpus comprend 14 textes de petite taille développant chacun, une des 3 thématiques très distinctes constituant des classes naturelles : Michael Jordan (texte 0 à 3), Microsoft (texte 4 à 7) et Mozart (texte 8 à 13).

Corpus II

Ce corpus est composé des livres 1 et 2 sur La civilisation des Arabes écrits en français par Gustave Le Bon (1884) [Bon, 1884]. Il comprend six (06) chapitres développant respectivement de : « L'Arabie », « Les Arabes », « Les Arabes avant Mahomet », « Mahomet – Naissance de l'empire arabe », « Le Coran » et « Les conquêtes des Arabes ». La segmentation des textes se fait à l'intérieur de chaque chapitre ou groupe de chapitre car il ne nous est pas possible de savoir par exemple à la vue d'un segment de texte s'il parle « des arabes » ou « des arabes avant

Mahomet » ou encore de « Les conquêtes des Arabes ». Cette manière de faire permet aussi de ne pas former des segments de texte composés de sous-segments issus de textes de plusieurs chapitres à la fois. Nous n'aurions sinon aucun repère dans un contexte de classification non supervisée.

5.3 K-skip-N-gram (de caractères, de mots) comme descripteur

Avec le corpus I, aucun K-skip-N-gram de mots expérimenté n'a conduit à une classification non supervisée menant à la confirmation de nos classes naturelles (connaissance à priori) tandis que tous les K-skip-N-gram de caractères ont permis de retrouver nos trois (03) groupes d'individus (textes).

K-Skip-N-Gram													
	De mots							De caractères					
	k	1	2	3	4	5		k	1	2	3	4	5
n							n						
2		×	×	×	×	×	2		✓	✓	✓	✓	✓
3		×	×	×	×	×	3		✓	✓	✓	✓	✓
4		×	×	×	×	×	4		✓	✓	✓	✓	✓
5		×	×	×	×	×	5		✓	✓	✓	✓	✓

Table 5.1: Tableau récapitulatif des tests pour le paramétrage des K-skip-N-gram de mots et de caractères

[×] : Trois classes non retrouvées [✓] : Trois classes retrouvées

Les couples (2,5), (2,1) de caractères et (2,5) de mots ont été retenus pour le corpus I tandis que pour le corpus I, les couples (3,5) de caractères et (2,5) de mots ont été retenus. Les paramètres K et N sont dès lors fixés.

5.3.1 Choix du nombre de classes

En faisant varier le nombre de classes jusqu'à quatre (04), soit le tiers du nombre de textes dans le corpus I, tout algorithme de classification confondu (SOM, K-MEDOID, CAH),

il ressort du vote, par considération des indices de Dunn et Silhouette, un nombre optimal de trois (03) classes, ce qui est conforme à la connaissance à priori que nous avons dudit corpus.

			Nombre de classes			Choix
			2	3	4	
5-skip-2-gram de caractères	hierarchical					3
		dunn	0.9362	0.9502	**0.956**	
		silhouette	0.0577	**0.064**	0.0612	
	pam					
		dunn	0.9362	0.9502	**0.956**	
		silhouette	0.0577	**0.064**	0.0612	
	som					
		dunn	**0.9108**	**0.9108**	0.9025	
		silhouette	**0.0523**	0.0309	0.0503	
5-skip-2-gram de mot	hierarchical					3
		dunn	0.9836	**0.9869**	**0.9869**	
		silhouette	0.0059	**0.0066**	0.0051	
	pam					
		dunn	0.9836	**0.9869**	0.9855	
		silhouette	0.0054	**0.0066**	0.0055	
	som					
		dunn	**0.9791**	0.9786	0.9786	
		silhouette	**0.0006**	-0.0032	0.0003	

Table 5.2: Choix du nombre de classes pour le corpus I (Dunn et Silhouette) 'K-skip-N-gram'

En faisant varier le nombre de classes entre 3 et 7 (soit le tiers du nombre de textes dans le corpus II), il ressort du vote par maximum des valeurs de Dunn et de silhouette que les coupures expérimentés pour le deuxième corpus sont celles en 7 et en 5 classes avec le K-skip-N-gram.

			Nombre de classes					Choix 1	Choix 2
			3	4	5	6	7		
5-skip-3-gram de caractères	hierarchical							7	5
		dunn	0.8243	0.8315	0.8599	0.8599	**0.8993**		
		silhouette	0.0415	0.04	**0.0419**	0.0388	0.035		
	pam								
		dunn	0.8315	0.8315	0.8701	**0.8815**	**0.8815**		
		silhouette	**0.0397**	0.0303	0.0342	0.0335	0.0252		
	som								
		dunn	0.7453	0.8508	0.8599	0.8133	**0.8781**		
		silhouette	-0.013	0.0107	**0.0279**	0.02	0.0073		
5-skip-2-gram de mot	hierarchical							5	na
		dunn	0.9707	0.9802	**0.9841**	**0.9841**	**0.9841**		
		silhouette	0.0142	**0.0155**	0.0144	0.0126	0.0111		
	pam								
		dunn	0.9791	0.9802	**0.9841**	**0.9841**	**0.9841**		
		silhouette	**0.0166**	0.0155	0.0144	0.0126	0.0106		
	som								
		dunn	0.9089	0.9303	**0.9368**	0.9237	0.9125		
		silhouette	-0.0106	-0.0012	**0.0009**	-0.0061	-0.0055		

Table 5.3: Choix du nombre de classes pour le corpus II (Dunn et Silhouette) 'K-skip-N-gram'

5.3.2 Corpus I (NBA, Microsoft, Mozart)

Les 1-skip-2-gram de caractères permettent de retrouver nos 03 classes (NBA, Microsoft, Mozart) avec la CAH confirmant ainsi la connaissance à priori sur le corpus I. Au niveau des 1-skip-2-gram de mots par contre, un important phénomène de chainage est observé et cela malgré l'utilisation du l'indice d'agrégation du 'ward.D2' (où le carré des distances est utilisé). Nous remarquons néanmoins les regroupements pertinents des textes 2 et 3 puis 12 et 13 appartenant effectivement à des classes différentes (NBA, Mozart).

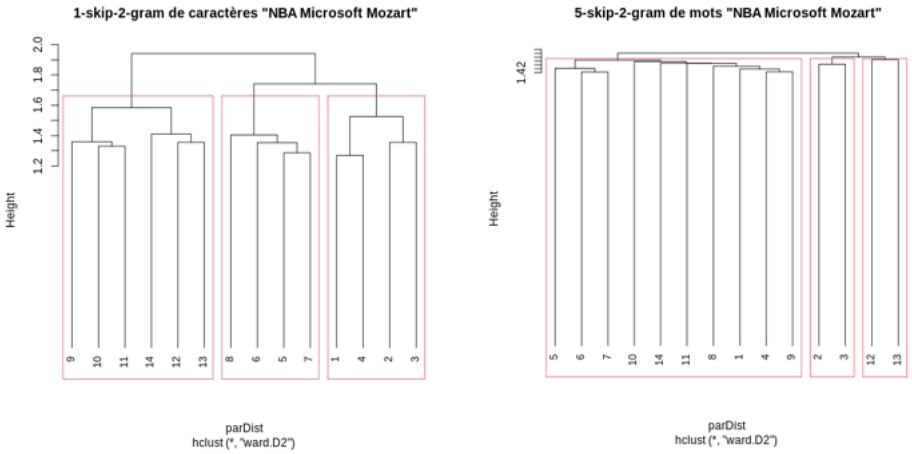


Figure 5.3: Corpus I, dendrogrammes K-skip-N-gram de caractères et de mots

En regardant en détails les données, nous remarquons que sur les 5533 '5-skip-2-gram' de mots formés avec le corpus I, les textes 10, 14, 11, 8, 1, 4 et 9 ne se rejoignent que faiblement. Seuls 32 des 5533 variables sont partagées avec tout au plus deux (02) de ces sept (07) textes, les 5501 autres sont possédées de manière mutuellement exclusive. Autrement dit, pour chacune de ces 32 variables, il n'y a parmi les sept (07) textes, que deux (02) pour lesquels, elles ont la valeur 1.

Les textes 5, 6 et 8, même si faisant partie de la même classe formée avec les 7 autres, se détachent nettement d'eux. Nous remarquerons qu'il y a 172 variables sur 5533 dont la valeur est non nulle et partagée par au moins deux (02) d'entre les 3 textes mais qu'à la différence majeure d'avec le sous-groupes des 7 autres, il y'en a 6 pour lesquels la valeur est positive pour à la fois les textes 5, 6 et 7. Les liens unissant ces trois (03) derniers sont donc relativement plus élevés que ceux de l'autre sous-groupe même si, contrairement aux groupes (2,3) et (12, 13), ils (les

liens) demeurent négligeables comme nous le verrons dans la suite.

Les vecteurs représentants les textes 2 et 3 prennent ensemble la valeur 1 sur 60 dimensions. Ni l'un ni l'autre n'est lié d'aucune façon avec les autres textes. Le groupe (12 ,13), quant à lui, se démarque de tous les autres textes, cependant, la liaison entre les individus de cette classe ne repose que sur 6 dimensions soit 10 fois moins que dans le groupe (2,3).

Aucun K-skip-N-gram de mots expérimenté n'a permis de retrouver nos 3 classes avec le l'algorithme du K-médoïde tandis que tous les couples (N, K) essayés ont été concluant avec les caractères. Les résultats avec les 5-skip-2-gram de mots semblent moins bons avec la CAH qu'avec le K-médoïde. Aucun K-skip-N-gram de mots expérimenté n'a conduit à retrouver nos

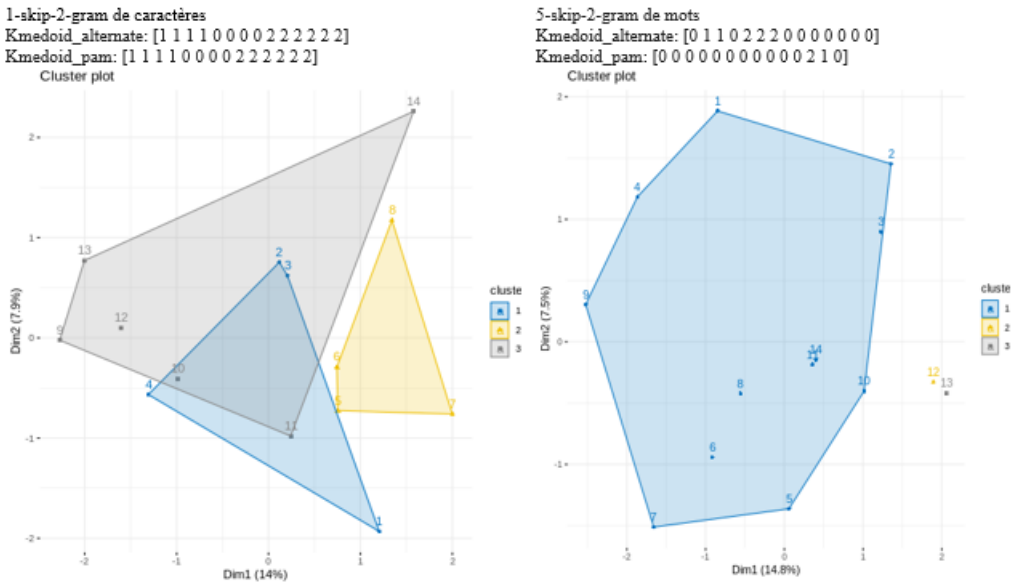


Figure 5.4: Corpus I, K-médoïde K-skip-N-gram de caractères et de mots

trois (03) classes. Avec les K-skip-N-gram de caractères, seuls les couples (N=2, K=1), (N=4, K=1) et (N=5, K=1) n'ont pas donné de bon résultats pour le SOM. Néanmoins, une interprétation conjointe des graphiques de distance et de mappage, nous permet de mettre en exergue une proximité non négligeable au niveau des nœuds regroupant respectivement les textes (4, 5) et (10,11,13).

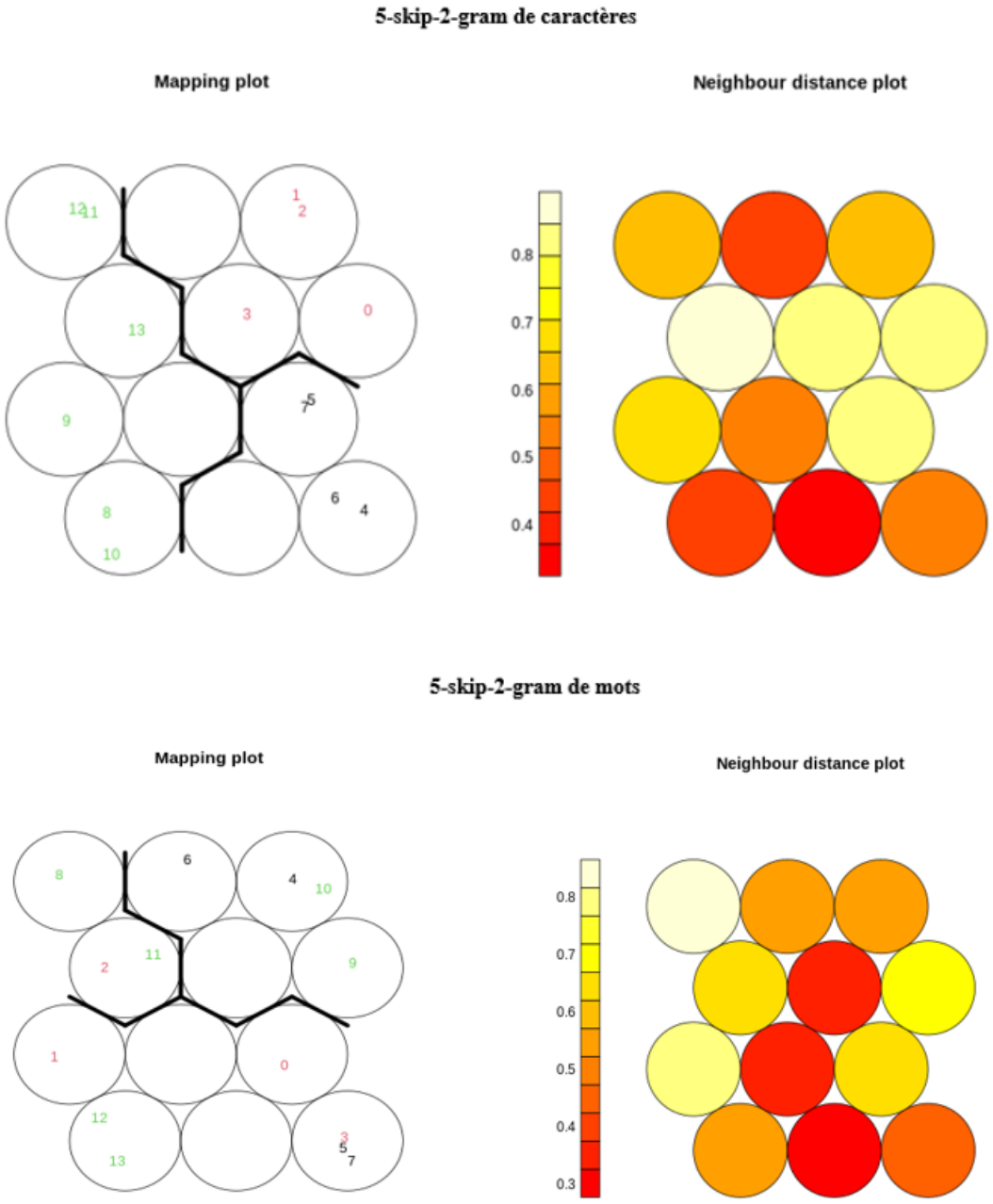


Figure 5.5: Corpus I, graphiques de mappage et de distance pour les K-skip-N-gram de caractères et de mots

Dans cette expérimentation, les résultats avec le K-skip-N-gram de mots sont moins bons que ceux avec K-skip-N-gram de caractères. En effet, les méthodes (CAH, K-médoïde et SOM) mènent dans le deuxième cas, à retrouver exactement nos trois (03) classes.

5.3.3 Corpus 2 (livre I et II de la civilisation des arabes)

Comme vu un peu plus haut, Le premier choix de nombre de classes se portait sur sept (07) et le deuxième sur cinq (05) pour les K-skip-N-gram de caractères. Trois (03) des classes formées restent invariantes au nombre de coupures. Les textes 16 et 17 rejoignent le groupe des textes 14 et 15. Les textes 20 et 21 rejoignent celui des textes 18, 6 et 19.

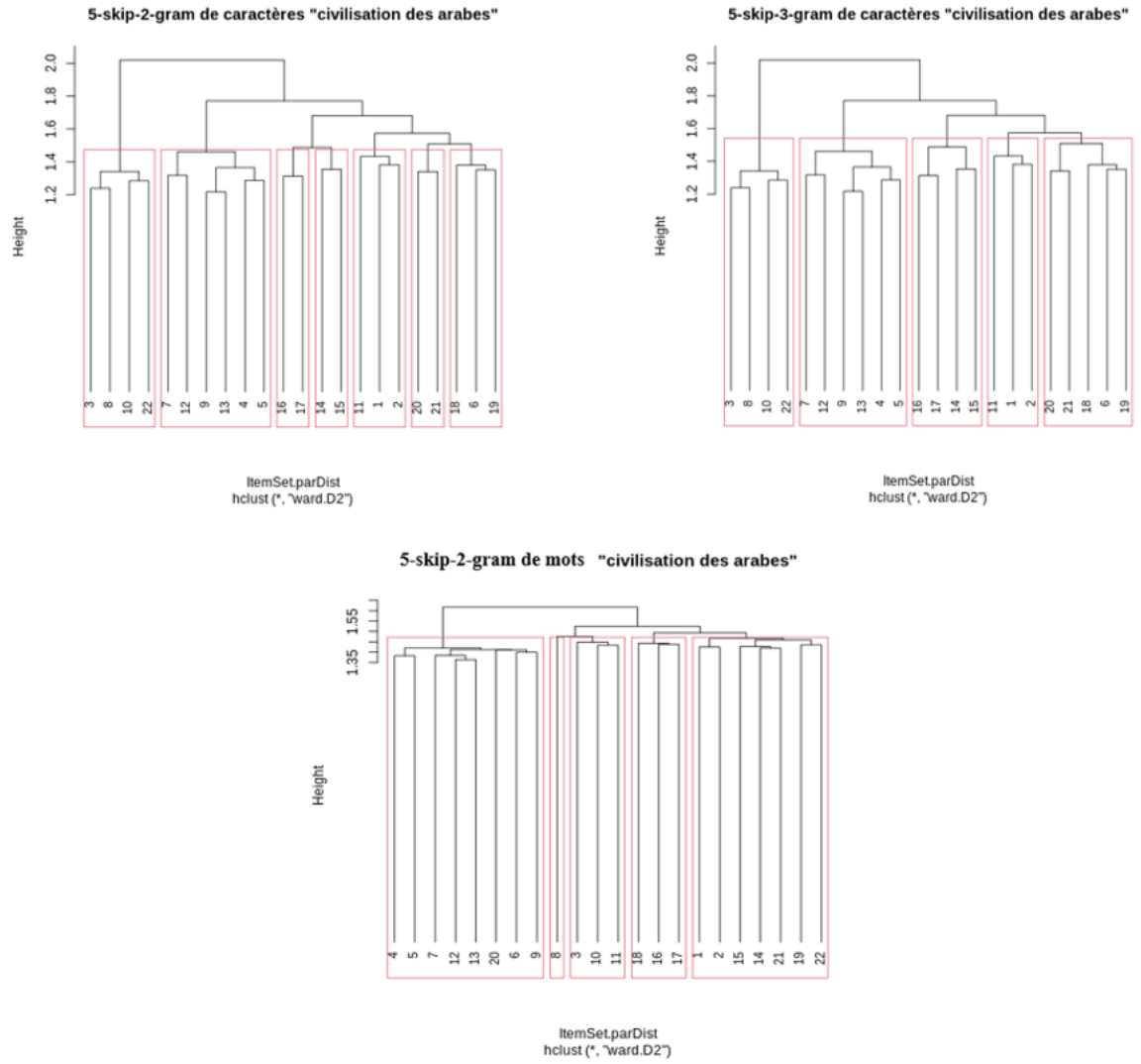


Figure 5.6: Corpus II, dendrogrammes K-skip-N-gram de caractères et de mots

La CAH avec les 5-skip-2-gram de mots, construit des classes différentes de celles qui précèdent. Néanmoins, nous pouvons voir sur les trois (03) dendrogrammes, la persistance de quelques regroupements, il s'agit des textes (1, 2), (16, 17), (14,15), (12,13), (20,21) et (4,5). Contraire-

ment au 5-skip-3-gram de caractères, et cela est aussi bien valable avec la CAH qu'avec le K-médoïde, les classes formées par les 5-skip-2-gram de mots sont moins balancées. Avec le K-médoïde, nous observons, un groupe de texte revenant dans tous les cas de figures, il s'agit de (16, 17, 18).

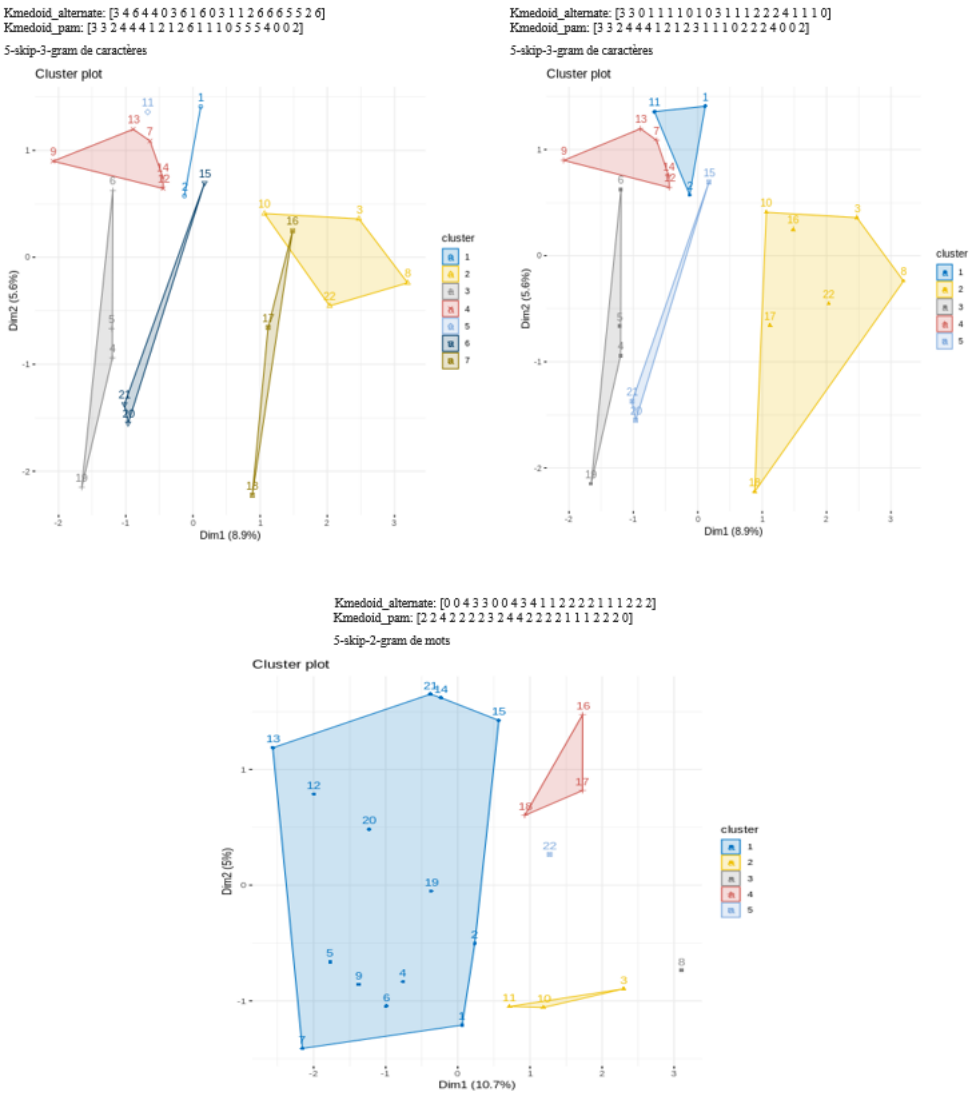


Figure 5.7: Corpus II, K-médoïde K-skip-N-gram de caractères et de mots

L'interprétation conjointe des graphiques de mappage et de distance nous amène à nous intéresser aux nœuds (neurones) de regroupements dont le distance entre les individus (textes) est le plus faible. Dans ce cas, les résultats avec les K-skip-N-gram de caractères respectivement de mots diffèrent. Le premier cas met en exergue les textes (14,15,17), (4,5,6), (1,20,21) et (7,9,12, 13) tandis que le deuxième, (6,17,18) et (12, 14, 4, 5). Les textes 16 et 17 se retrouvent toujours dans la même classe.

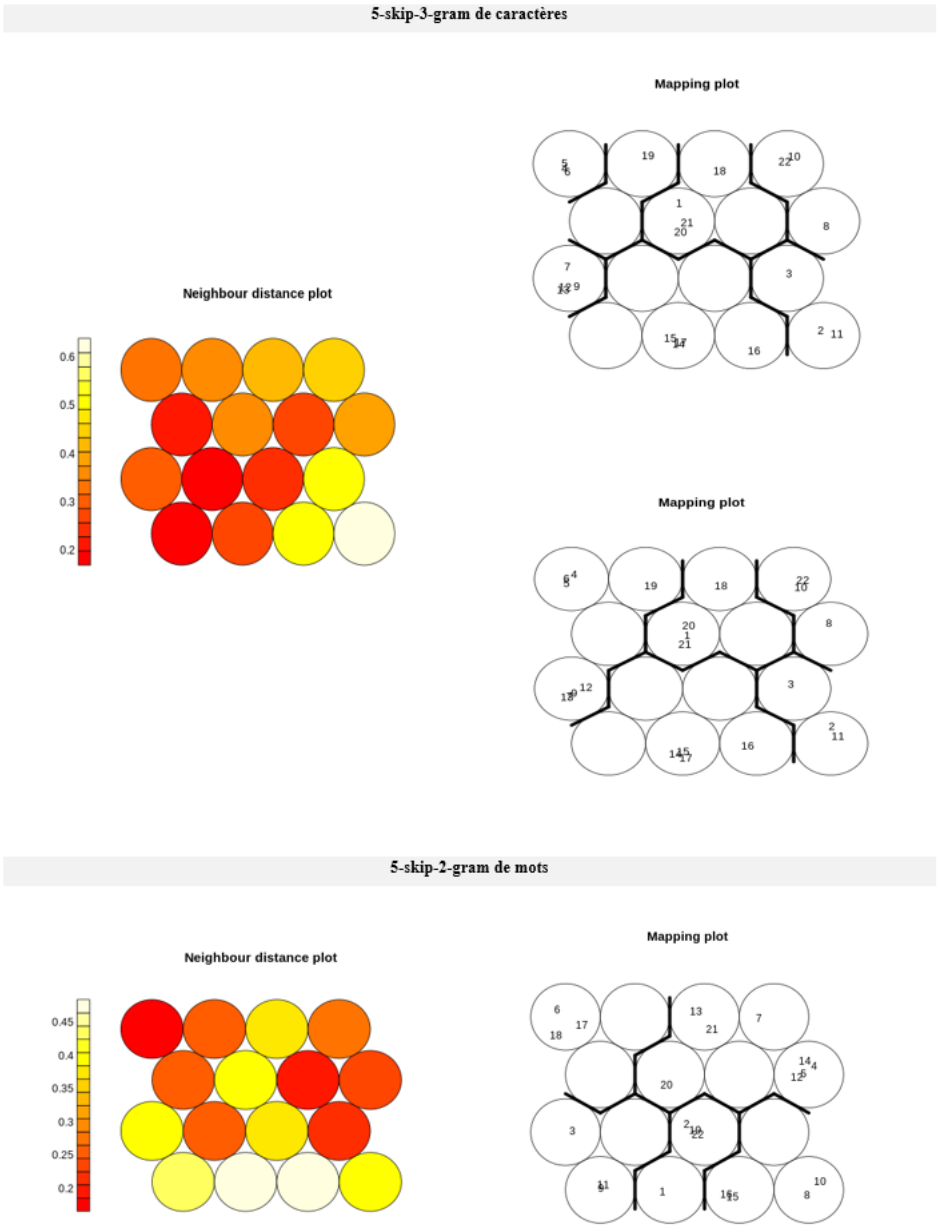


Figure 5.8: Corpus II, graphiques de mappage et de distance pour les K-skip-N-gram de caractères et de mots

Le tableau suivant se veut un récapitulatif des résultats de cette sous partie, étant donné que nous n'avons pas de connaissance à priori de l'existence de classes naturelles, nous utilisons et utiliserons par les suite les formes fortes (regroupements insensibles à la méthode de classification non supervisée utilisée) ainsi que les formes faibles (regroupements instables). Il sera utilisé en fin de chapitre à des fins de synthèse.

5-skip-3-gram de caractères	5-skip-2-gram de mots
Formes fortes	
(16, 17), (20,21), (4,5)	(17, 18)
Formes faibles	
(14, 15), (1,2), (12,13)	(10,11), (4,5)

Table 5.4: Corpus II, formes fortes et formes faibles pour les K-skip-N-gram de caractères et de mots

5.4 Mots (Classic Bag of words) comme descripteur

5.4.1 Choix du nombre de classes

Nous suivons la même démarche que précédemment. Il ressort du vote, par considération des indices de Dunn et Silhouette, un nombre optimal de trois (03) classes, ce qui est conforme à la connaissance à priori que nous avons dudit corpus I.

			Nombre de classes			Choix
			2	3	4	
CORPUS I	hierarchical					3
		dunn	**0.9411**	**0.9411**	0.8516	
		silhouette	**0.2333**	0.1937	0.124	
	pam					
		dunn	0.857	**0.9411**	0.8516	
		silhouette	**0.2254**	0.1937	0.124	
	som					
		dunn	0.6748	**0.7362**	0.6624	
		silhouette	0.097	**0.1302**	0.0834	

Table 5.5: Choix du nombre de classes pour le corpus I (Dunn et Silhouette) 'sac de mots'

			Nombre de classes					Choix 1	Choix 2
			3	4	5	6	7		
CORPUS II	hierarchical							3	7
		dunn	0.7045	0.7177	**0.8366**	**0.8366**	**0.8366**		
		silhouette	**0.3457**	0.2891	0.2765	0.2716	0.2614		
	pam								
		dunn	0.7934	0.7724	0.7891	0.7599	**0.8366**		
		silhouette	**0.3209**	0.2814	0.2563	0.2499	0.2614		
	som								
		dunn	**0.6253**	0.599	0.2751	0.287	NA		
		silhouette	**0.3415**	0.225	0.0724	0.0873	NA		

Table 5.6: Choix du nombre de classes pour le corpus II (Dunn et Silhouette) 'sac de mots'

Pour ce qui est du corpus II, nous expérimenterons les coupures en 3 et 7 classes.

5.4.2 Corpus I (NBA, Microsoft, Mozart)

En prenant comme descripteur le mot ou sac de mots, toutes les méthodes de classification utilisées permettent de retrouver nos (03) trois classes naturelles (connaissance à priori).

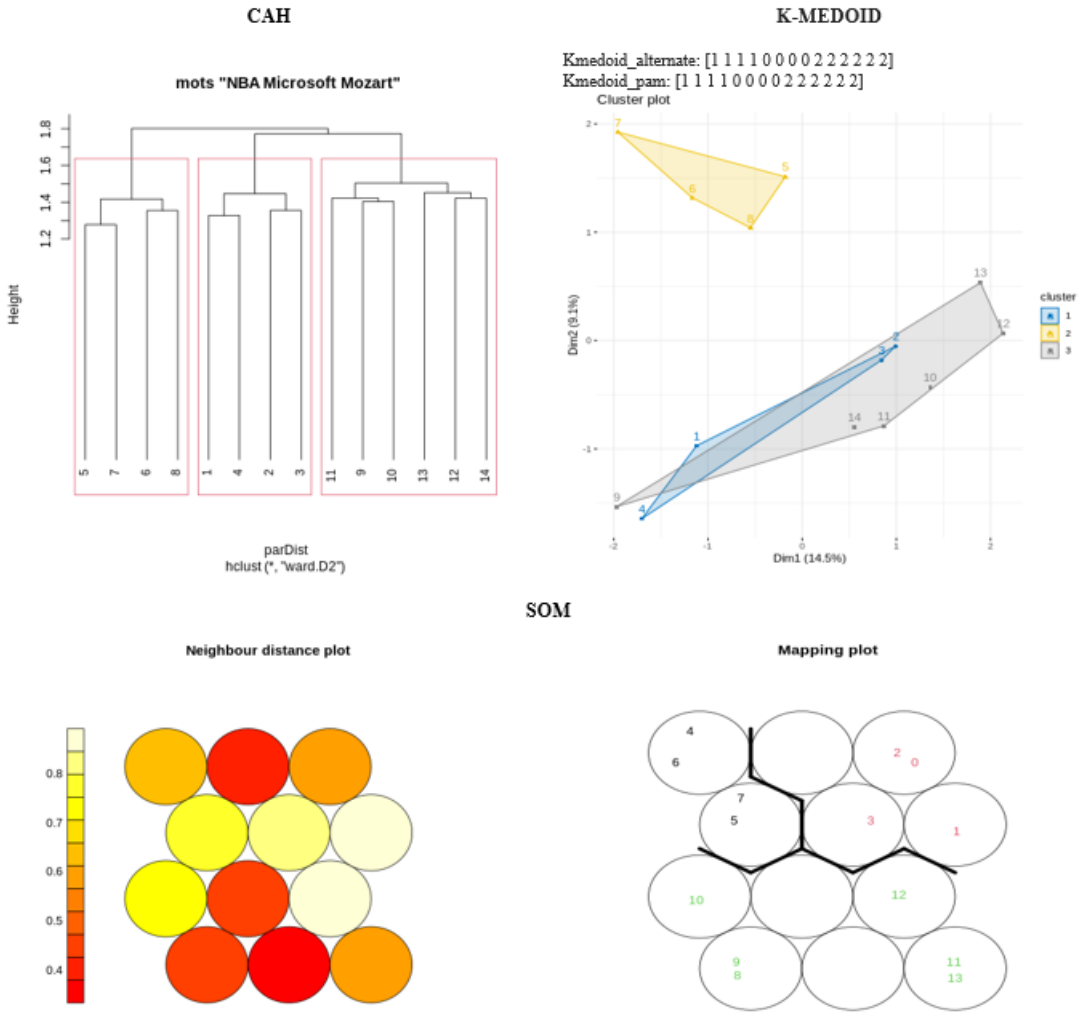


Figure 5.9: Résultats CAH, K-médoïde et SOM, corpus I, sacs de mots

5.4.3 Corpus 2 (livre 1 et 2 de la civilisation des arabes)

Avec la CAH, le découpage en trois (03) groupes (premier choix) conduit à des classes équilibrées. En passant à un découpage de sept (07), aucun singleton (classe avec un seul texte)

n'est formé.

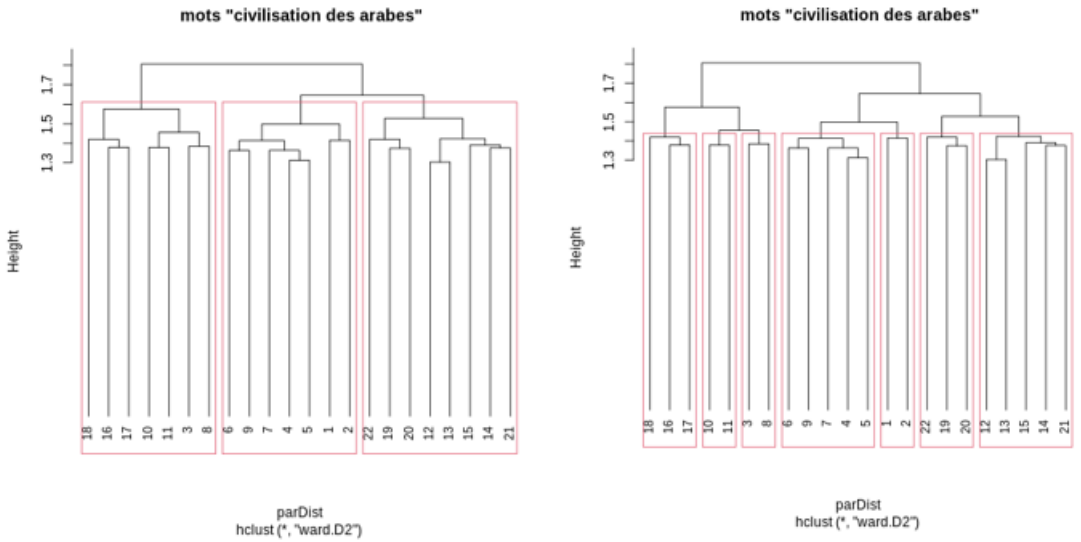


Figure 5.10: Corpus II, dendrogrammes K-skip-N-gram de caractères et de mots

Le K-médoïde quant à lui, conduit dans le premier cas, à une classe de grande taille (14) et deux de petite taille (3 et 5) puis dans le deuxième cas, à respectivement deux (02) singletons, regroupements de quatre (04) et une classe de six (06) textes. Certains groupes formés se retrouvent aussi bien avec la CAH qu'en utilisant le K-médoïde. Il s'agit notamment de (16,17,18), (19,20,22) et (4,5,6,7).

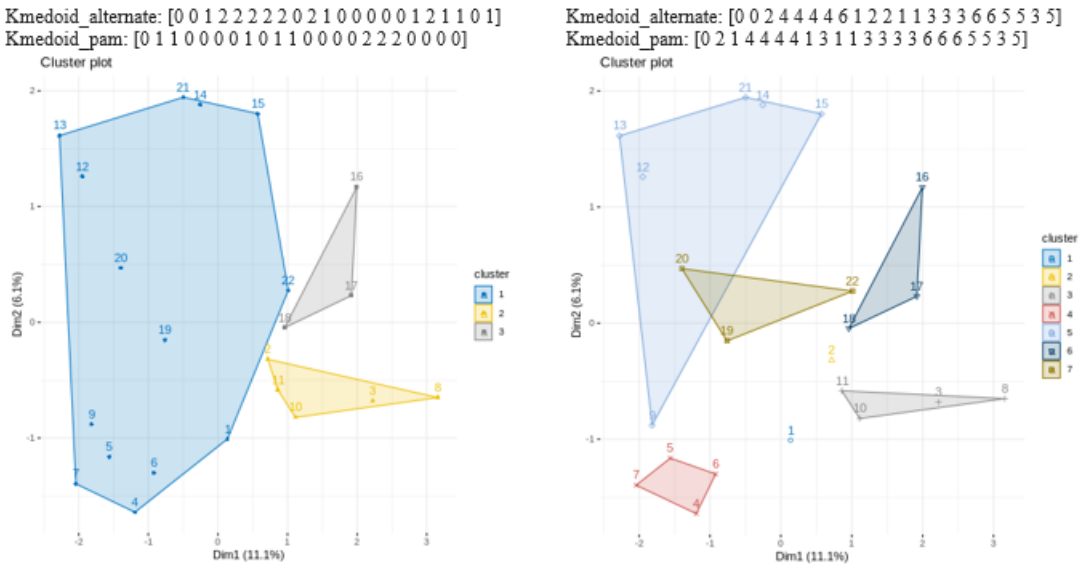


Figure 5.11: Corpus II, K-médoïde, sacs de mots

Une expérimentation menée sur le même corpus dans le mémoire « Vers une plate-forme informatique pour l'expérimentation d'outils de classification » [Bokhabrine et al., 2019], avait conduit au résultat suivant pour le K-médoïde.

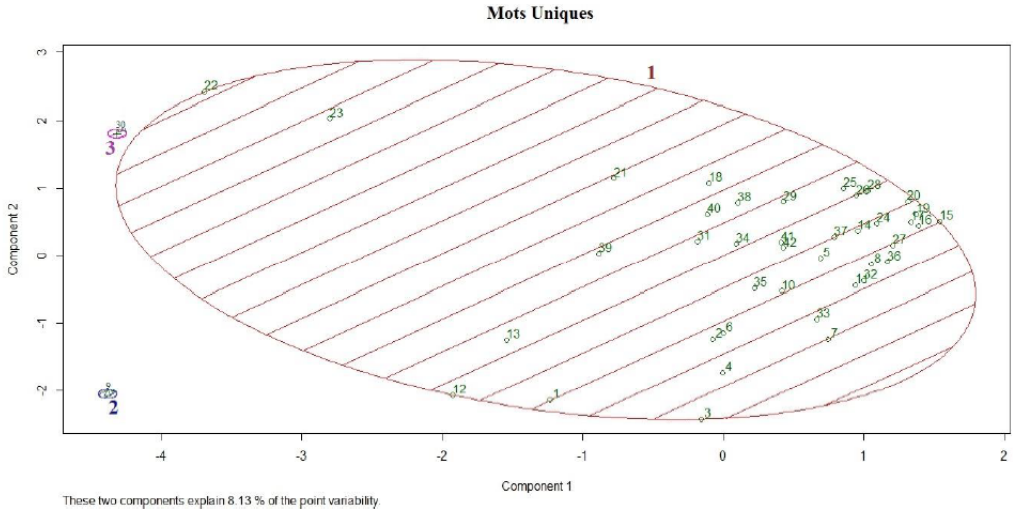


Figure 5.12: Corpus II, K-médoïde, étude précédente utilisant des segments de textes plus petits, sac de mots

La même étude conduit au résultat suivant pour la CAH. Nous observons d'importants phénomènes de chainage. Le critère de d'agrégation utilisé ici n'est pas le « Ward.D2 » mais plutôt le « complete linkage ».

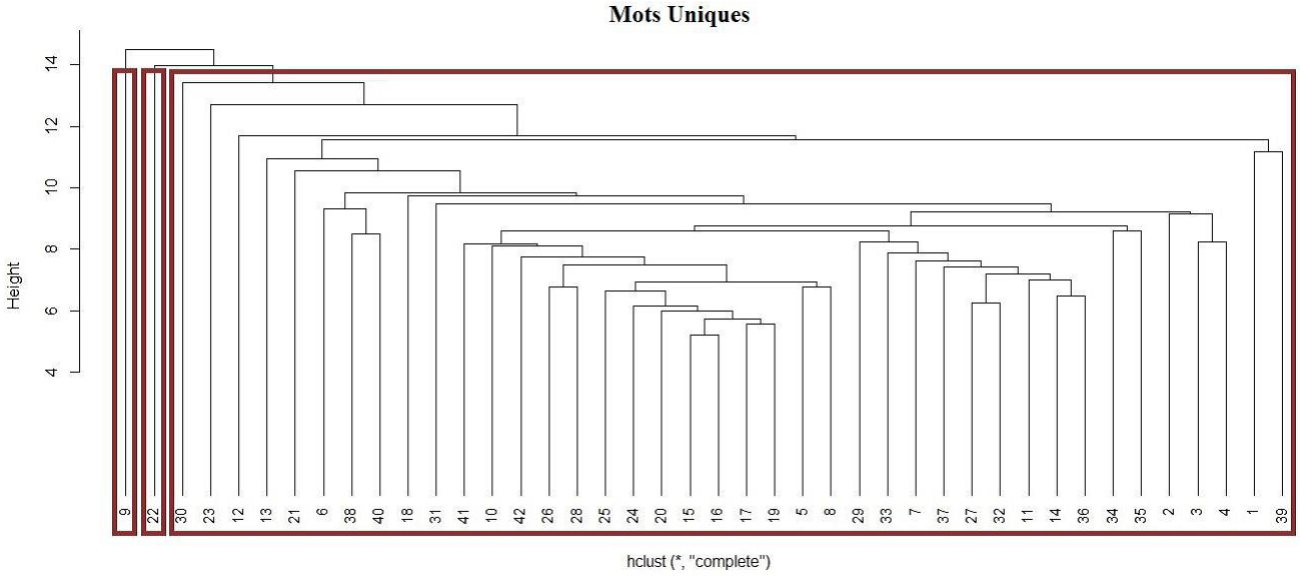


Figure 5.13: Corpus II, dendrogramme d'une étude précédente utilisant des segments de textes plus petits, sac de mots

Cette différence de résultats pourrait aussi être due à la taille des segments de texte utilisée pour la classification. Il est effectivement plus probable de trouver des mots communs entre des textes de grande taille qu'entre des segments de petite taille. Aussi, nous ne savons pas si les vecteurs en entrée étaient binaires comme dans notre cas ou s'il s'agissait plutôt d'effectifs.

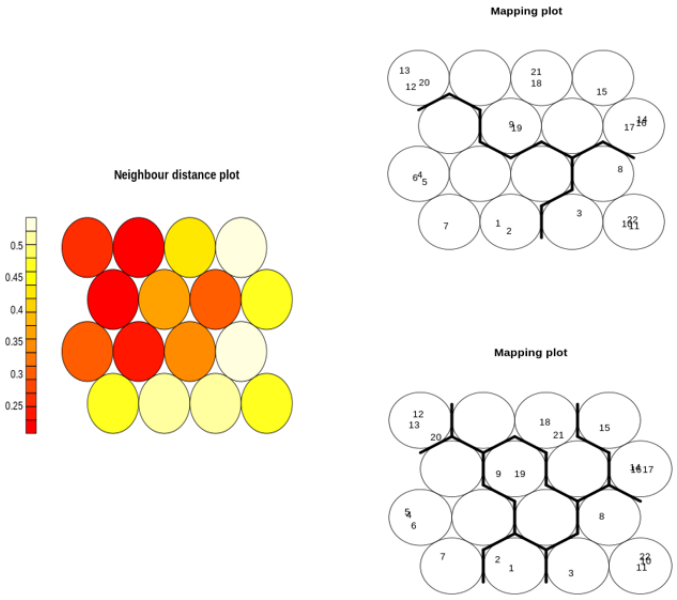


Figure 5.14: Corpus II, SOM, graphiques de mappage et de distance, sac de mots

L'interprétation conjointe des graphes de distance et de mappage nous fait d'abord nous intéresser aux nœuds (neurones) regroupant les textes (12,13,20) et (4,5,6). Ensuite nous notons une proximité assez marquée (au moins trois individus), des textes (14,16,17) et (10,11,22).
Le tableau ci-dessous se veut une synthèse des résultats de cette sous-partie. Il sera exploité en fin de chapitre dans le cadre d'une synthèse des résultats.

Formes faibles	**Formes fortes**
(20,22)	(4,5,6,7)
(1,2)	(10,11)
	(16,17)
	(12,13)

Table 5.7: Corpus II, formes fortes et formes faibles pour le sac de mots

5.5 TF-IDF comme descripteur

5.5.1 Choix du nombre de classes

Nous suivons la même démarche que précédemment. Il ressort du vote, par considération des indices de Dunn et Silhouette, un nombre optimal de trois (03) classes, ce qui est conforme à la connaissance à priori que nous avons dudit corpus. Nous expérimenterons pour le corpus II, les découpages en 5 et 3 classes.

			Nombre de classes			Choix
			2	3	4	
CORPUS I	hierarchical					3
		dunn	0.9365	0.9798	**0.9906**	
		silhouette	0.0477	**0.0672**	0.053	
	pam					
		dunn	0.8631	0.9798	**0.9906**	
		silhouette	0.036	**0.0672**	0.053	
	som					
		dunn	0.8659	**0.9798**	0.9371	
		silhouette	-0.0034	**0.0672**	0.0472	

Table 5.8: Choix du nombre de classes pour le corpus I (Dunn et Silhouette) 'TF-IDF'

			Nombre de classes					Choix 1	Choix 2
			3	4	5	6	7		
CORPUS II	hierarchical							5	3
		dunn	0.8993	**0.9248**	**0.9248**	0.8795	0.9138		
		silhouette	**0.0469**	0.0452	0.0362	0.039	0.0456		
	pam								
		dunn	0.8827	0.8441	0.8441	0.8765	**0.8996**		
		silhouette	**0.0471**	0.0348	0.0309	0.0429	0.0443		
	som								
		dunn	0.8741	0.87	**0.8857**	0.8415	0.8802		
		silhouette	0.0402	0.0181	**0.0229**	0.0152	0.0152		

Table 5.9: Choix du nombre de classes pour le corpus II (Dunn et Silhouette) 'TF-IDF'

5.5.2 corpus I (NBA, Microsoft, Mozart)

A ce stade, c'est sans grande surprise (au vu de ce qui précède) que nous arrivons encore à retrouver nos trois (03) classes naturelles en prenant comme descripteur le TF-IDF et cela quelle que soit la méthode de classification utilisée.

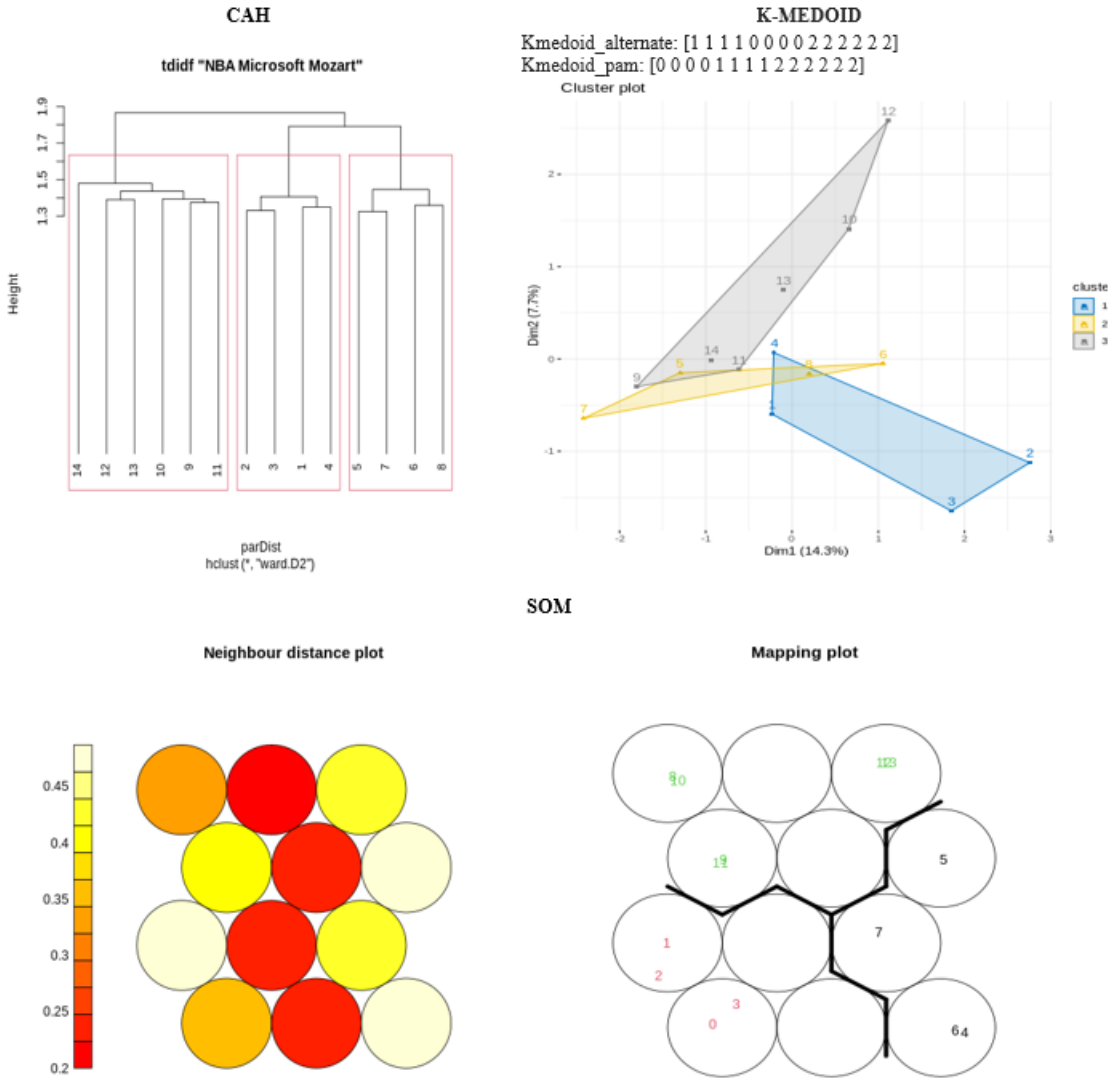


Figure 5.15: Résultats CAH, K-médoïde et SOM, corpus I, TF-IDF

5.5.3 Corpus 2 (livre 1 et 2 de la civilisation des arabes)

La coupure en trois (03) classes avec la CAH conduit à des classes équilibrées. En passant à cinq (05) classes, aucun singleton ne se forment et les regroupements sont d'au moins trois (03) individus. Nous remarquons une classe assez inhabituellement rencontrée depuis le début de nos expérimentations, celles formée par les textes 1, 2 et 3. Nous verrons par la suite s'il s'agit ou non d'une forme forte.

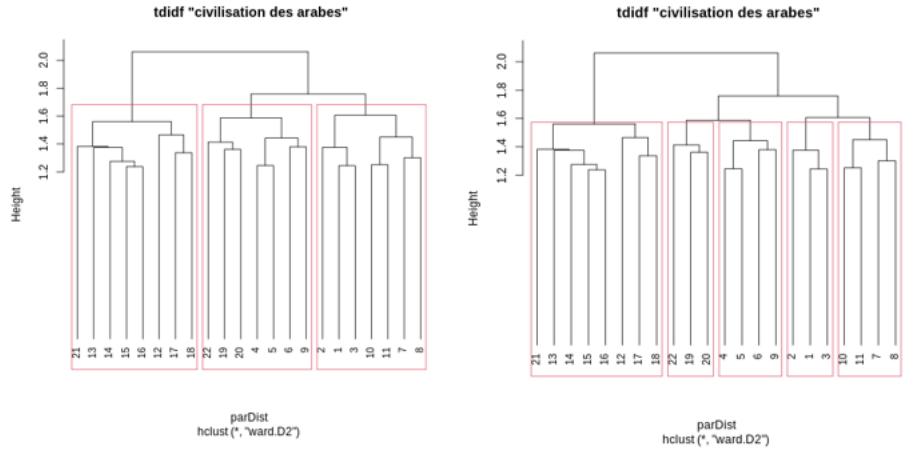


Figure 5.16: Corpus II, dendrogrammes, TF-IDF

Comme la CAH, le K-médoïde conduit dans le premier cas, à trois (03) groupes de tailles quasi similaires. Dans le deuxième cas, nous observons un détachement au niveau des textes 17 et 18, formant maintenant, à eux seuls une classe.

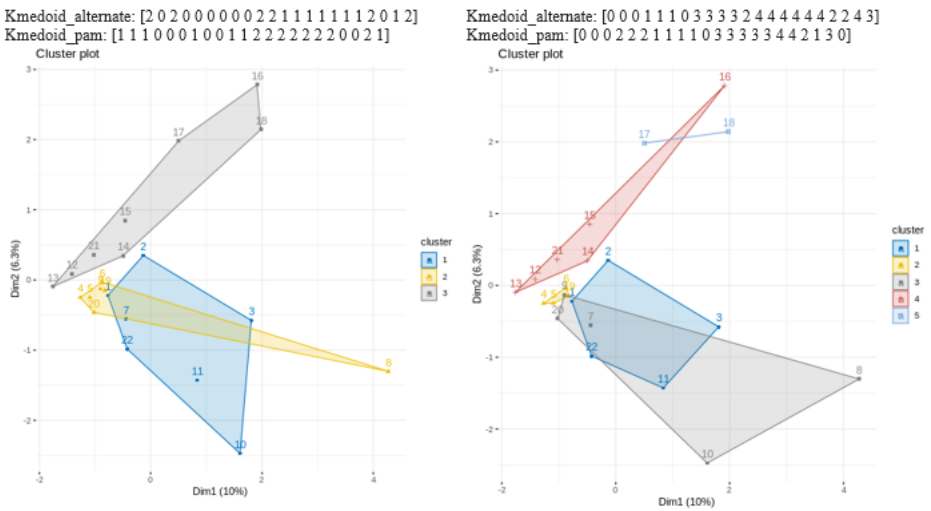


Figure 5.17: Corpus II, K-médoïde, TF-IDF

Les textes 4, 5, 6 et 19 forment aussi une classe à part entière et nous notons les textes 1,2,3 appartiennent à une même classe dans les deux situations.

L'interprétation conjointe des graphiques de mappage et de distance nous amène à nous intéresser aux nœuds regroupant les textes (19,20,22), (9,10,11), (4,5,6), (7,8) et (1,2,3). Nous notons une concentration inhabituelle de textes au niveau d'un neurone alors qu'en terme de distance, la proximité entre eux est relativement faible. Il s'agit des textes 13,14,15 et 16.

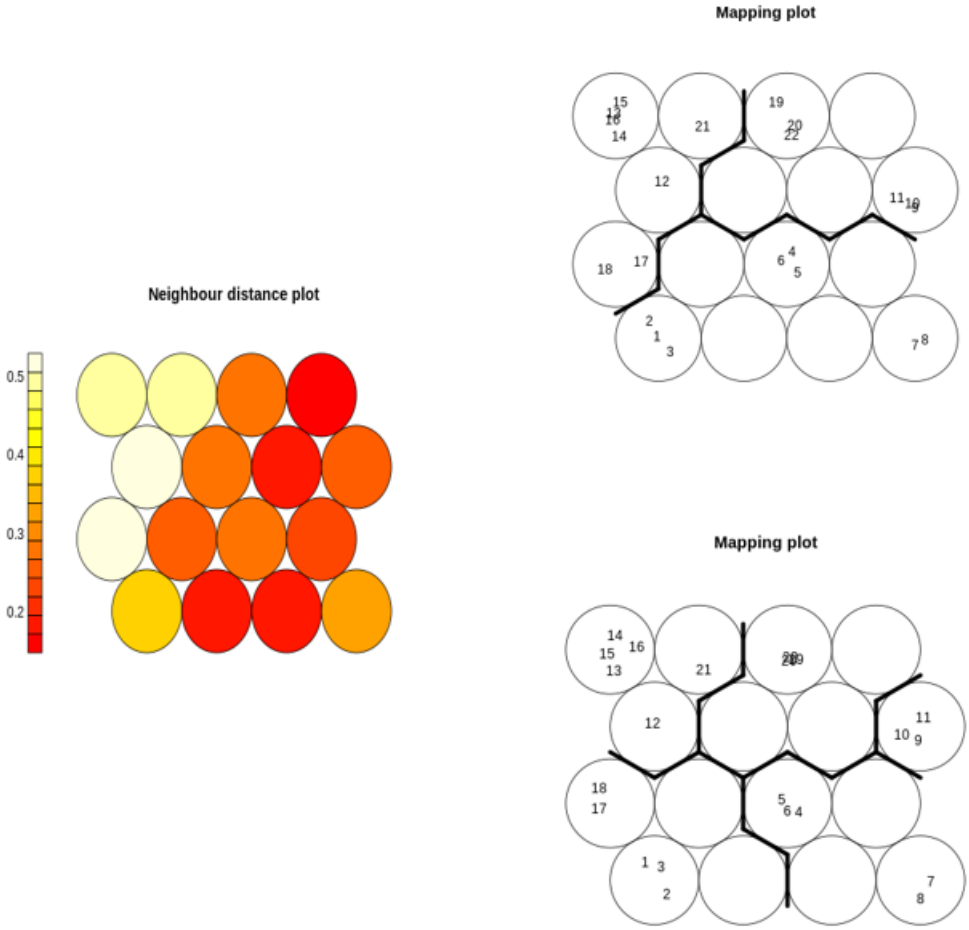


Figure 5.18: Corpus II, SOM, graphiques de mappage et de distance, TF-IDF

Le tableau suivant se veut un récapitulatif de cette sous partie et sera commenté dans la suite.

Formes faibles	Formes fortes
(10,11) (7,8)	(12,13,14,15,21) (19, 20, 22) (4,5,6) (1,2,3) (17,18)

Table 5.10: Corpus II, formes fortes et formes faibles pour le TF-IDF

5.6 Itemsets fréquents comme descripteur

5.6.1 Choix du nombre de classes

Nous suivons la même démarche que précédemment. Il ressort du vote, par considération des indices de Dunn et Silhouette, un nombre optimal de trois (03) classes, ce qui est conforme à la connaissance à priori que nous avons dudit corpus. Pour le corpus II, nous expérimenterons un découpage en 6 classes.

			Nombre de classes			Choix
			2	3	4	
CORPUS I	hierarchical					3
		dunn	0,9499	**1,0602**	1,0025	
		silhouette	0,4409	**0,5062**	0,5042	
	pam					
		dunn	0,9499	**1,0602**	1,0025	
		silhouette	0,4409	0,5062	**0,5262**	
	som					
		dunn	**0,9499**	0,5869	0,6794	
		silhouette	0,4409	0,4405	**0,5201**	

Table 5.11: Choix du nombre de classes pour le corpus I (Dunn et Silhouette) 'tous itemsets fréquents'

			Nombre de classes					Choix 1	Choix 2
			3	4	5	6	7		
CORPUS II	hierarchical							6	na
		dunn	0,6475	0,7774	0,7774	**0,9344**	0,8344		
		silhouette	0,2083	0,2727	**0,2919**	0,2588	0,2852		
	pam								
		dunn	0,6905	**0,7774**	0,6906	0,6906	0,6906		
		silhouette	0,2467	0,2727	0,2650	0,2761	**0,3188**		
	som								
		dunn	0,6475	0,6707	0,6733	**0,6906**	0,6616		
		silhouette	0,2393	0,1891	0,2722	**0,2945**	0,2732		

Table 5.12: Choix du nombre de classes pour le corpus II (Dunn et Silhouette) 'tous itemsets fréquents'

5.6.2 corpus I (NBA, Microsoft, Mozart)

- **Tous les itemsets (support >=0.5)**

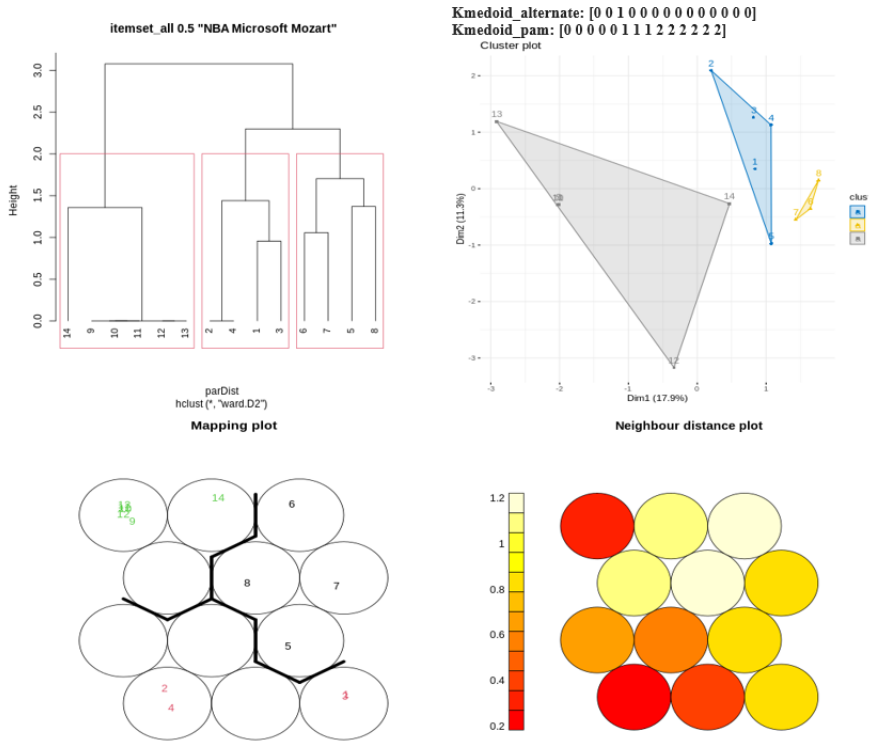


Figure 5.19: Résultats CAH, K-médoïde et SOM, corpus I, tous les itemsets fréquents, support minimum= 0,5

- **1-itemsets (support >=0.5)**

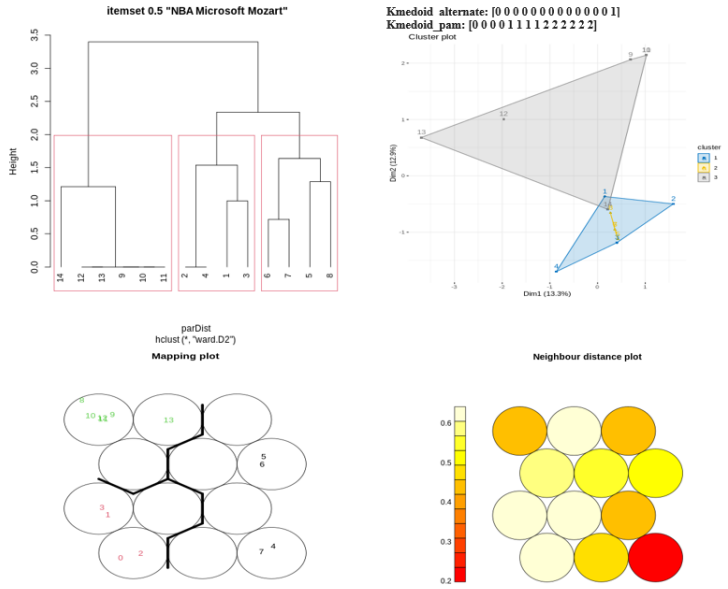


Figure 5.20: Résultats CAH, K-médoïde et SOM, corpus I, 1-itemsets, support minimum=0,5

- **Supersets de longueur maximal (support >=0.5)**

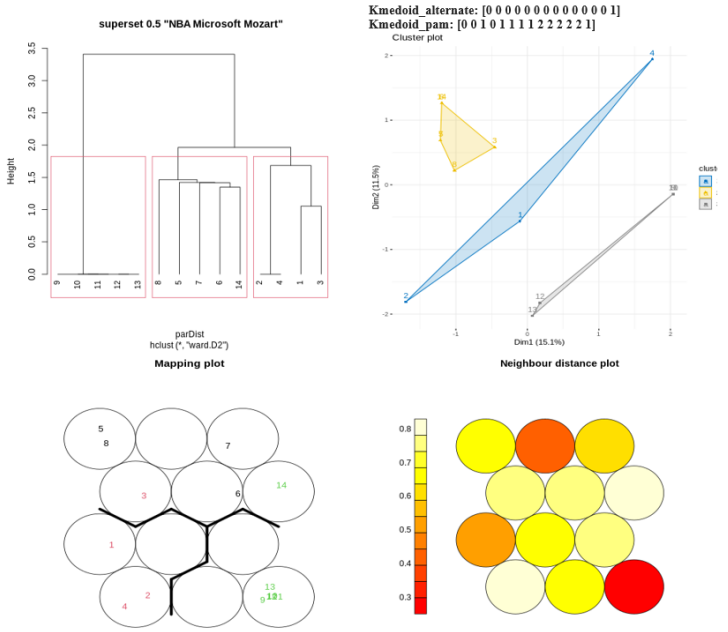


Figure 5.21: Résultats CAH, K-médoïde et SOM, corpus I, supersets de longueur maximale, support minimum=0,4

- **Itemsets partagés (support >=0.5)**

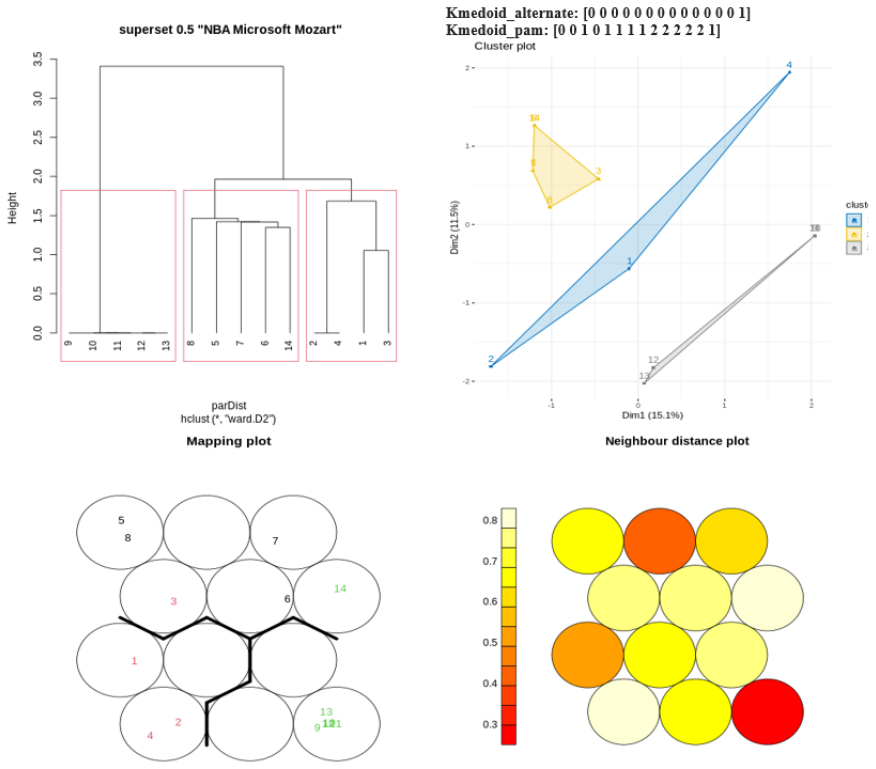


Figure 5.22: Résultats CAH, K-médoïde et SOM, corpus I, itemsets partagés, support minimum=0,5

- **Récapitulatif**

Utiliser les 1-itemsets fréquents plutôt que tous les itemsets fréquents donnent de meilleurs résultats comme le montre le tableau récapitulatif ci-dessous.

	CAH	K-MEDOID PAM	SOM
1-itemsets	✓	✓	✓
Tous itemsets	✓	×	✓
Supersets L. max	×	×	×
Itemsets partagés	✓	×	✓

Table 5.13: Récapitulatif des performances des 4 sous-descripteurs liés à l'itemset fréquent sur le corpus I

[×] : Trois classes non retrouvées [✓] : Trois classes retrouvées

5.6.3 Corpus 2 (livre 1 et 2 de la civilisation des arabes)

- **Tous les itemsets (support >=0.2)**

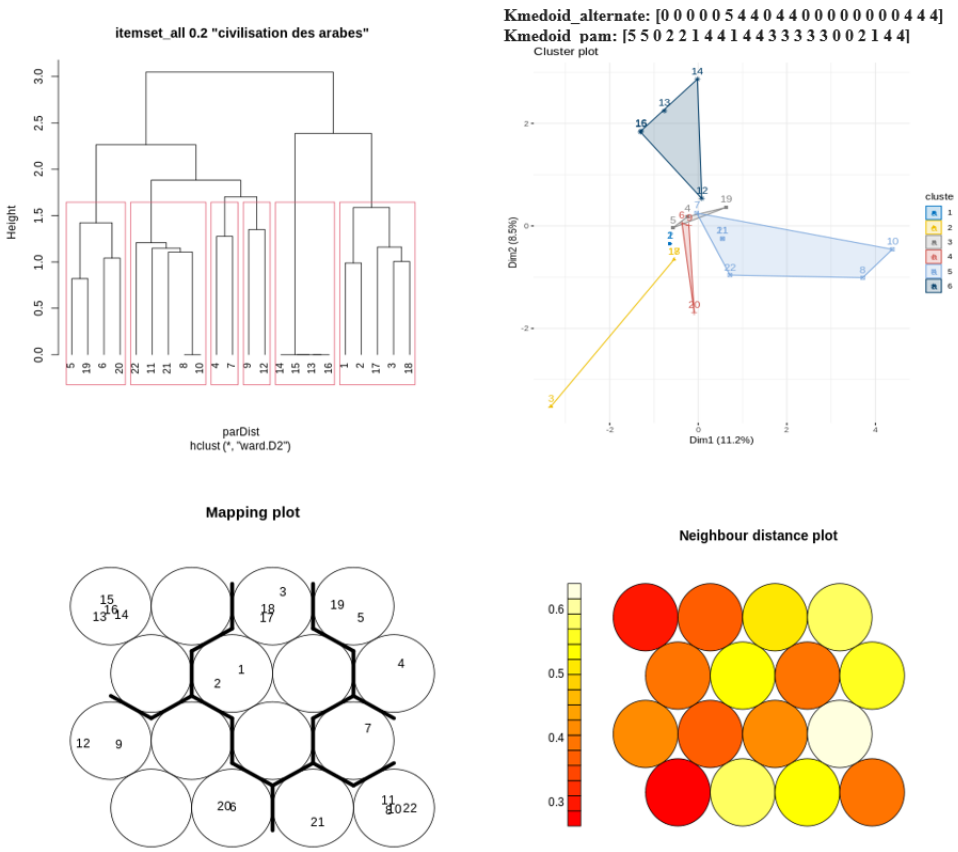


Figure 5.23: Résultats CAH, K-médoïde et SOM, corpus II, tous les itemsets fréquents, support minimum= 0,2

Le tableau suivant se veut un récapitulatif de cette sous partie et sera commenté dans la suite.

Formes faibles	Formes fortes
(13,14,15,16) (17,18)	(13,14,16) (1,2) (21,22)

Table 5.14: Corpus II, formes fortes et formes faibles pour le TF-IDF

- **1-itemsets (support >=0.2)**

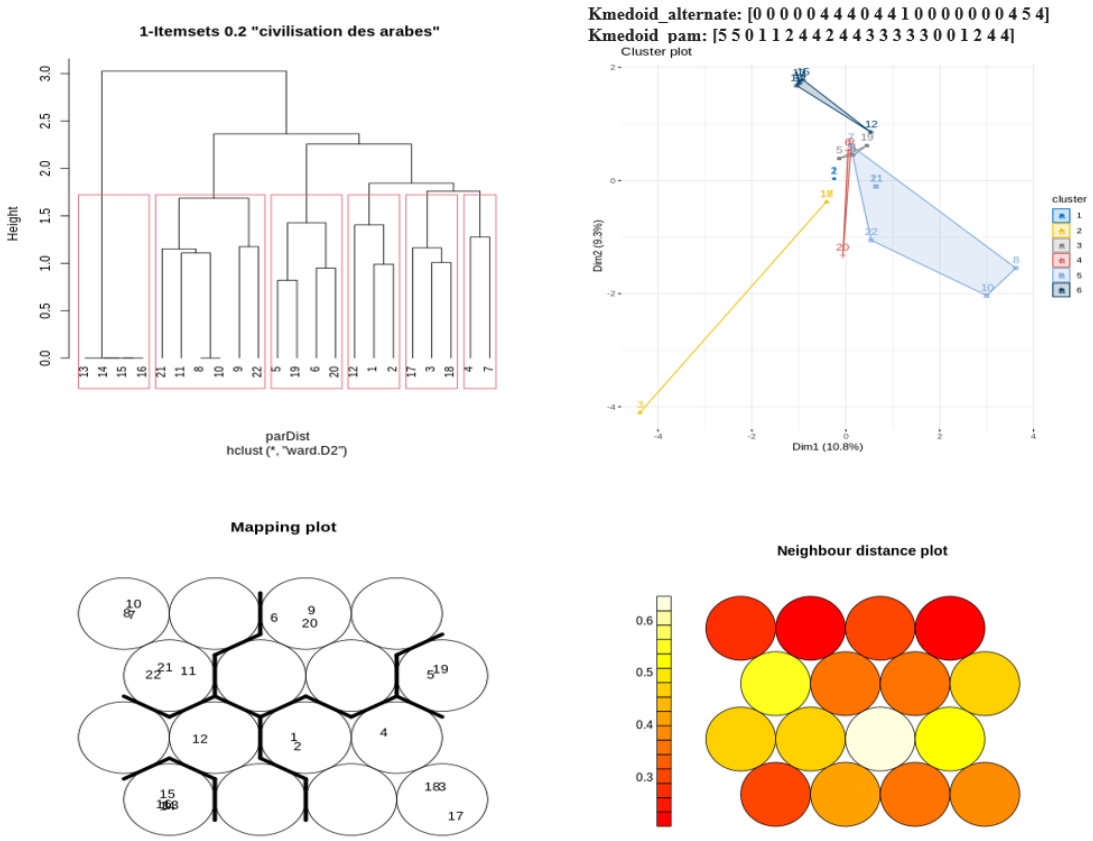


Figure 5.24: Résultats CAH, K-médoïde et SOM, corpus II, 1- itemsets, support minimum= 0,2

Le tableau suivant se veut un récapitulatif de cette sous partie et sera commenté dans la suite.

Formes faibles	Formes fortes
(7,8)	(13,14,15,16) (8,10,11,21,22) (17,18,3) (1,2)

Table 5.15: Corpus II, formes fortes et formes faibles pour tous les 1-itemsets

- **Supersets de longueur maximale (support >=0.2)**

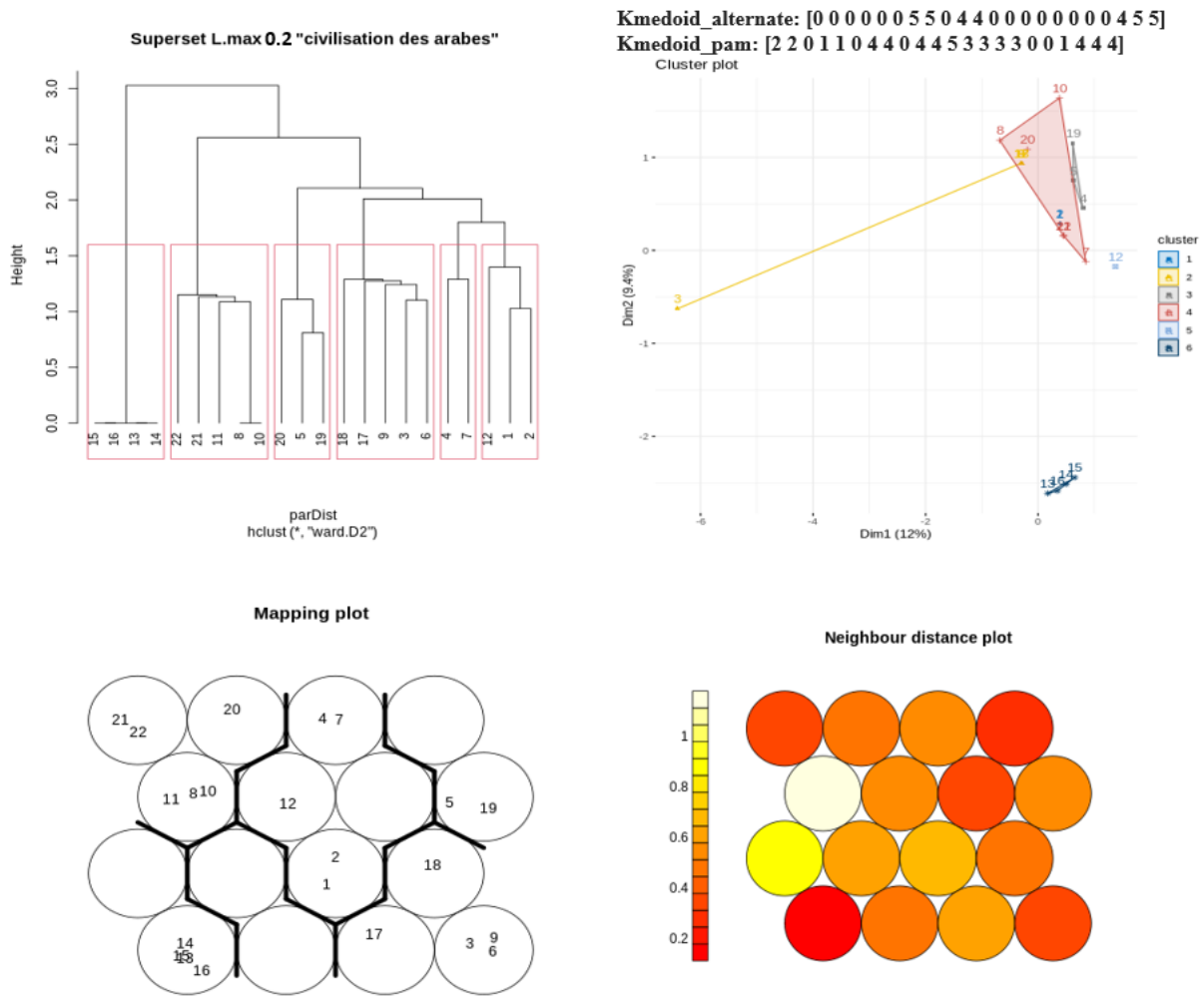


Figure 5.25: Résultats CAH, K-médoïde et SOM, corpus II, itemsets de longueur maximale, support minimum= 0,2

Le tableau suivant se veut un récapitulatif de cette sous partie et sera commenté dans la suite.

Formes faibles	Formes fortes
	(3,6,9,17,18) (13,14,15,16) (21,22) (4,7) (1,2)

Table 5.16: Corpus II, formes fortes et formes faibles pour les supersets de longueur maximale

- **Itemsets partagés (support >=0.2)**

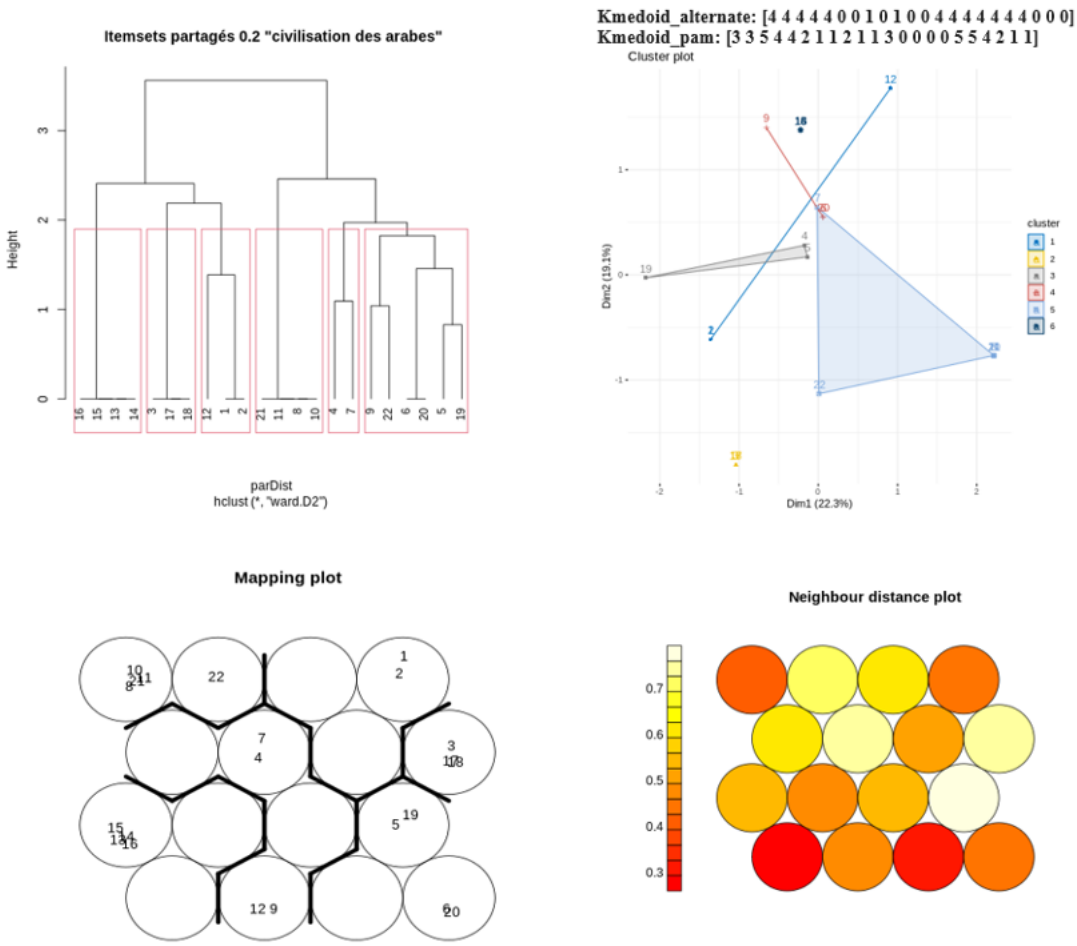


Figure 5.26: Résultats CAH, K-médoïde et SOM, corpus II, tous les itemsets partagés, support minimum= 0,2

Le tableau suivant se veut un récapitulatif de cette sous partie et sera commenté dans la suite.

Formes faibles	Formes fortes
(4,7)	(10,11) (1,2) (3,17,18)

Table 5.17: Corpus II, formes fortes et formes faibles pour les supersets de longueur maximale

- **Récapitulatif**

De toutes les formes d'Itemsets fréquents expérimentées, le 1-Itemset est sans doute le plus performant. Comme le montre le tableau suivant, il permet la formation de trois (03) formes fortes d'au moins 3 individus alors qu'avec tous les itemsets fréquents et les itemsets partagés, il y en a qu'une (01).

	1-itemsets	**Supersets L.max**	**Tous Itemsets**	**Itemsets partagés**
Nombre de formes fortes de 3 individus ou plus	**3**	**2**	**1**	**1**

Table 5.18: Nombre de formes fortes d'au moins 3 textes par sous-descripteurs liés à l'itemset fréquent

5.7 Résultats SVM sur Corpus I

En utilisant les fonctions de transformation linéaire, sigmoïde et polynomiale, tous les descripteurs sauf le K-skip-N-gram de mots, parviennent à classer correctement les textes de corpus I dans les trois (03) classes. Les résultats sont moins bons avec la fonction polynomiale qui s'avère produire un résultat moins mauvais avec les 1-itemsets. Le tableau suivant, présentent les erreurs de classement par descripteurs et type de fonction de transformation.

	linéaire	sigmoïde	polynomiale
5-skip-2-gram de caractères	0	0	2
5-skip-2-gram de mots	2	2	2
Sac de mots	0	0	2
TF-IDF	0	0	2
1-itemsets	0	0	1

Table 5.19: Nombre d'erreurs de classement par descripteur, corpus I, SVM

5.8 Synthèse des résultats

Le tableau synthétique des formes fortes partagées montre à quel point les résultats varient d'un descripteur à l'autre. Sur sept (07) formes fortes partagées, seul une (01) est commune à trois (03) des cinq (05) descripteurs principaux retenus.

	(1, 2)	(4, 5)	(4, 5, 6)	(10, 11)	(12, 13)	(16, 17)	(17, 18)
Mots		■	■	■	■	■	
1-Itemsets	■			■			■
TF-IDF	■	■	■		■		
5-skip-3-gram de caractères		■				■	
5-skip-2-gram de mots							■

Légende
■ Présence
Absence

Figure 5.27: Formes fortes partagées par les principaux descripteurs, corpus II

De manière générale, les formes fortes obtenues avec le TF-IDF et les 1-Itemsets sont plus imposantes (nombre de textes les composant) et plus nombreuses (nombre de formes fortes). Elles sont suivies, sous cet angle d'appréciation, par respectivement, les mots ou sac de mots, le 5-skip-3-gram de caractères et le 5-skip-2-gram de mots.

	Tf-idf	1-itemsets	mots	5-skip-3-gram de caractères	5-skip-2-gram de mots
Nombre de formes fortes pour le corpus II	5	5	4	3	1
Nombre de formes fortes de 3 individus ou plus pour le corpus II	4	3	1	0	0
Nombre de liaison (formes fortes à Chapitres) pour le corpus II	4	2	2	2	1

Table 5.20: Tableau synthétique global des résultats pour le corpus II

Ces regroupements ont-ils du sens ? Sur la figure suivante, nous faisons l'exercice de relier chaque forme forte à une partie (chapitre) des livres I et II du livre la « civilisation des arabes ». Les chapitre I et II sont ici considérés comme une même partie conformément à des travaux précédents [Bokhabrine et al., 2019]. Le principe graphique est le suivant :

– Si la forme forte correspond entièrement à une partie (chapitre) du livre, nous lui

donnons la couleur de fond de la partie concernée et la lui relions par un trait de même couleur ;

– Dans le cas contraire, la couleur de fond reste neutre et les sous-regroupements la composant et correspondant entièrement à une partie (chapitre) du texte sont mis en gras et couleur foncée.

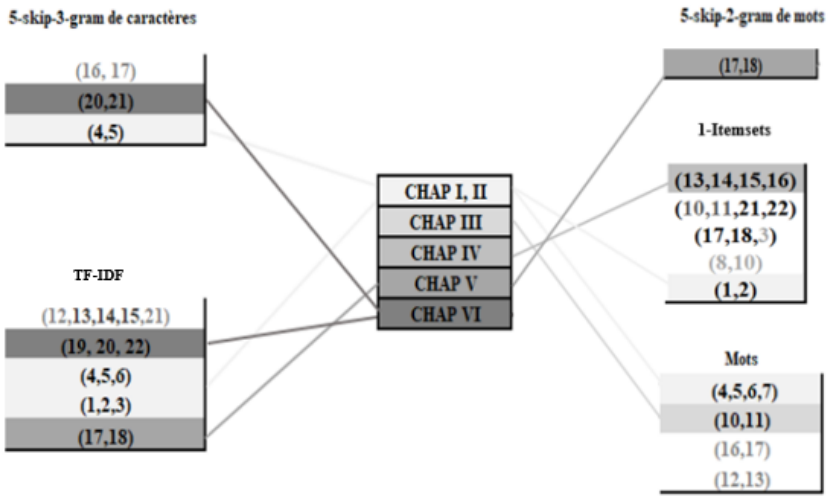


Figure 5.28: Exercice de liaison des formes fortes aux parties des livres I et II de la 'civilisation des arabes' pour les descripteurs principaux

Les formes fortes obtenues avec le TF-IDF peuvent, dans leur quasi-totalité (4/5 ou 80 %), être entièrement associées à des parties du corpus II (chapitres) développant chacune une thématique bien précise. Elles sont suivies dans cette logique par celles formées avec, respectivement, les 1-Itemsets, les mots, les 5-skip-3-gram de caractères et les 5-skip-2-gram de mots. Les textes 13,14 et 15 appartiennent au chapitre IV (Les anciennes religions de l'Arabie). Le 1-itemset est le seul descripteur avec lequel, est formée la forme forte (13,14,15,16), correspondant à l'entièreté du chapitre IV. Il est suivi du TF-IDF qui conduit à une forme forte avec 75 % des textes du chapitre IV (13,14,15).

Les formes (16,17) et (12,13) ont la particularité d'être composés d'un texte marquant la fin d'un chapitre et d'un autre par lequel commence le chapitre qui suit. Par exemple, le texte 12 parle des « anciennes religions de l'Arabie » et clos le chapitre III tandis que le texte 13 entame le chapitre IV et a pour titre « Mahomet – Naissance de l'empire arabe ». Le texte 21 parle du 'résumé de l'histoire des arabes', il appartient au chapitre VI (Les conquêtes des Arabes). A

l'instar des textes de son chapitre, le texte 21 a tendance à s'associer avec les textes de tous les autres chapitres.

Dans le cadre d'une analyse thématique, respectivement, par potentiel décroissant, le TF-IDF, le 1-Itemset et le sac de mot semblent être les descripteurs les plus pertinents. Le K-skip-N-gram de caractères a été concluant pour des thématiques très distinctes mais n'a produit aucune forme forte de plus de 3 textes pour des thématiques liées, ce qui met en question son adéquation à une telle tâche dans des conditions réelles (expérimentation avec le Corpus II). Le K-skip-N-gram de mots a produit les résultats les moins bons. Il a conduit néanmoins à quelques rares regroupements intéressants mais sans rien de vraiment concluant.

CHAPITRE 6: Conclusion et perspectives

Au terme de cette étude, nous avons comparé plusieurs techniques de vectorisation de texte (sac de mots, Itemsets fréquents, Tf-idf, K-skip-N-gram de mots et respectivement de caractères). Ceci, dans un contexte de classification de documents et à travers quatres méthodes: la classification ascendante hiérarchique (CAH), le K-médoïde, les cartes auto-organisatrices (SOM) et le support vector machine (SVM). Les expérimentations ont été menées sur un petit corpus avec une connaissance a priori des classes existantes, puis sur un corpus plus grand, abordant des thématiques assez voisines.

En évaluant le potentiel des différents descripteurs, et cela, en perspective d'une analyse thématique de documents, il est ressorti, l'importance des descripteurs étudiés dans leur capacité à extraire de l'information pertinente du texte. Les performances diffèrent néanmoins d'un descripteur à l'autre. Le Tf-idf et le 1-itemset fréquent semblent plus prometteurs tandis que le K-skip-N-gram de mots a produit les résultats les moins convainquants. Un peu plus dans le détail, le 1-itemset fréquent semble plus performant que l'itemset en général, le superset de longueur maximale et l'itemset partagé. La CAH et le SOM quant à eux semblent donner de meilleurs résultats que le k-médoïde. Aussi les K-skip-N-gram de caractères se sont avérés plus efficaces que ceux au niveau des mots. Ce dernier résultat va dans le sens des conclusions d'une étude comparative précédente [Miao et al., 2005]. La justification évoquée à ce résultat était que le N-gram de caractères occasionne moins de problèmes de données rares (data sparcity) et arrive à trouver des caractéristiques communes entre les mots partageant les mêmes racines, ceci, même avec des morphologies différentes (exemple: "finance" et "financière").

Comme contribution importante, nous avons donné suite à deux (02) perspectives d'une étude précédente [Bokhabrine et al., 2019, Bokhabrine et al., 2020] menée au sein du département, où l'auteur introduit les itemsets fréquents comme descripteurs de texte dans un contexte de classification de documents. Il s'est agi premièrement, de l'utilisation d'une autre méthode de classification, le support vector machine "SVM". Deuxièmement, il a été question d'élargir l'étude à d'autres textes. Nous avons utilisé les mêmes corpus mais pas les mêmes segments de textes. Les segments utilisés dans cette étude sont de plus grande taille : 12 contre

20 segments pour le corpus I et 22 contre 42 segments pour le corpus II.

En perspectives, nous proposons que la même étude soit conduite avec des corpus d'une autre langue (anglais, arabe, etc.). Un support de 0,5 a été utilisé pour le corpus I et 0,2 pour le corpus II. Une autre étude pourrait se pencher sur la fixation de ce seuil dont le choix reste jusque-là quasi-arbitraire. Aussi, une étude pourrait expérimenter les K-skip-N-gram de mots et de caractères pour le sentiment analysis ou d'autres tâches du traitement du language naturel. De plus, un exercice intéressant consisterait à determiner les facteurs de performance des 1-itemsets par rapport aux autres itemsets fréquents et aux sacs de mots.

RÉFÉRENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. Proc Int Conf on Very Large Databases VLDB-94, page 487–499.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bokhabrine, A., Biskri, I., and Ghazzali, N. (2019). Frequent itemsets as descriptors of textual records. In International Conference on Computational Collective Intelligence, page 35–45.

Bokhabrine, A., Biskri, I., and Ghazzali, N. (2020). Textual clustering: Towards a more efficient descriptors of texts. Communications in Computer and Information Science, 1287:801–810.

Bon, G. L. (1884). La civilisation des arabes.

Britz, D., Guan, M. Y., and Luong, M.-T. (2017). Efficient attention using a fixed-size memory representation. arXiv preprint arXiv:1707.00110.

Cervantes, J., Li, X., and Yu, W. (2007). Svm classification for large data sets by considering models of classes distribution. In 2007 Sixth Mexican International Conference on Artificial Intelligence, Special Session (MICAI), page 51–60.

Charrad, M., Ghazzali, N., Boiteau, V., and Niknafs, A. (2014). Nbclust: An r package for determining the relevant number of clusters in a data set. Journal of Statistical Software, 61:1–36.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR, abs/1406.1078.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. N. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Dongare, A., Kharde, R., Kachare, A. D., et al. (2012). Introduction to artificial neural network. International Journal of Engineering and Innovative Technology (IJEIT), 2(1):189–194.

Friedman, N. and Halpern, J. Y. (2013). A qualitative markov assumption and its implications for belief change. CoRR, abs/1302.3578.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G. J., Dunson, D. B., and Dudík, M., editors, AISTATS, volume 15 of JMLR Proceedings, pages 315–323. JMLR.org.

Harris, Z. S. (1954). Distributional structure. WORD, 10(2-3):146–162.

Indolia, S., Goswami, A. K., Mishra, S., and Asopa, P. (2018). Conceptual understanding of convolutional neural network- a deep learning approach. Procedia Computer Science, 132:679–688. International Conference on Computational Intelligence and Data Science.

Kim, G., Fukui, K., and Shimodaira, H. (2018). Segmentation-free compositional n-gram embedding. CoRR, abs/1809.00918.

Krasnoshtan, D. (2019). On the number of k-skip-n-grams. CoRR, abs/1905.05407.

Labiad, A. (2017). Sélection des mots clés basée sur la classification et l'extraction des règles d'association ». mémoire de maitrise en informatique, université du québec à trois-rivières.

Listiawan, T. and Hudha, M. N. (2021). Design and implementation software for mining association rules (market basket analysis) to design product layout desicions. volume 1869 of Journal of Physics Conference Series, page 012121.

Miao, Y., Keselj, V., and Milios, E. (2005). Document clustering using character n-grams: A comparative evaluation with term-based and word-based clustering. pages 357–358.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Milligan, G. and Cooper, M. (1985). An examination of procedures for determining the number of clusters in a data set. Psychometrika, 50(2):159–179.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In Proceedings of the 24th international conference on Machine learning, pages 641–648.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In EMNLP, volume 14, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.

R Core Team (2013). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Rehman, Z., Anwar, W., Bajwa, U., Xuan, W., and Chaoying, Z. (2013). Morpheme matching based text tokenization for a scarce resourced language. PLoS ONE, 8(8):68178.

Saif, H., Fernandez, M., and Alani, H. (2014). On stopwords, filtering and data sparsity for sentiment analysis of twitter. Proceedings of the 9th International Language Resources and Evaluation Conference (LREC), pages 810–817.

Schmidhuber, Hochreiter, J., and S. (1997). Mémoire longue à court terme. calcul neuronal, 9:1735–1780.

Staudemeyer, R. C. and Morris, E. R. (2019). Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. CoRR, abs/1909.09586.

Tenney, I., Das, D., and Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline.

Turian, J., Ratinov, L.-A., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 384–394, Uppsala, Sweden. Association for

Computational Linguistics.

Van Rossum, G. and Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, and Illia (2017). Attention is all you need. In Advances in neural information processing systems, pages 5998–6008.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). Glue: A multi-task benchmark and analysis platform for natural language understanding.

Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network.

Zhang, X., Zhao, J. J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. CoRR, abs/1509.01626.

ANNEXES

Code de transformation du texte en forme de transaction pour itemsets fréquents

```
def text_itemsetfreq (donnee):
  if donnee=='livre.txt':
    OP2='long'
  else:
    OP2='court'
  text=[]
  with open(donnee, 'r', encoding='cp1252') as f:
    text.append(f.read())
  f.close()
 #



# un peu de nettoyage
  text[0]=text[0].replace('S\x01',"oe")
  text[0]=text[0].replace("\x19 ","'")
  text[0]=text[0].replace("\x00\\'","'")
  text[0]=text[0].replace("\x00","")
  text[0]=text[0].replace("ÿþ","")
  text[0]=text[0].replace('\\', "")
  text[0]=text[0].replace("\'", " ")
  text[0]=text[0].replace('"L', 'L')
  for k in (['1','2','3','4','5', '6','7','8','9','0','-','(',')','"','[',']']):
    text[0]=text[0].replace(k,'')

# découpage du corpus
  if OP2=="court":
    text_facile=text[0].split('\n\n\n\n')
  if OP2=="long":
    text_facile=text[0].split('\n\n')

 # indexation des différents textes
  namespace=globals()
```

```
33   n=len(text_facile)
34   for i in range (n):
35     namespace['paragraphe_%s' % i]=text_facile[i]
36   for i in range (n):
37     namespace['paragraph_%s' % i]=sent_tokenize(namespace['paragraphe_%s' %
       i])
38
39
40 #definition de ma liste de stopwords personnalisée
41   SW=list(stopwords.words('french'))+list(stopwords.words('english'))+list(
       punctuation)
42   for i in ['[]',',','a','et','entre','qu','b','c','d','e','f','g','h','i',
       'j','k','l','m','n','o','p','q','r','s','t','u', 'v','w','x','y','z','.'
       ,'1','2','3','4','5', '6','7','8','9','0','-','','un',
43           'tant','ainsi','alors','puisque', 'donc','ou','si','lors','[]']:
44     SW.append(i)
45
46 #transformation des textes en groupe de token et lemmatisation
47
48   lemmatizer= WordNetLemmatizer()
49   nlp = spacy.load('fr_core_news_md')
50
51   namespace=globals()
52
53   for p in range (n):
54     namespace['par_%s' % p]=[]
55     for i in range(len(sent_tokenize(namespace['paragraphe_%s' % p]))):
56       namespace['par_%s' % p].append([])
57       namespace['paragraph_%s' % p]=sent_tokenize(str(namespace['
       paragraphe_%s' % p]))
58       for j in namespace['paragraph_%s' % p][i].split():
59         j=j.replace('"','')
60         j=j.replace('.','')
61         if not j.lower() in SW:
62           y=nlp(j)
63           for it in y:
64             d=it.lemma_
```

```
            lexeme = nlp.vocab[d]
            if not d in SW:
              if lexeme.is_stop == False:
                namespace['par_%s' % p][i].append(d)
textes={}

for i in range(n):
  te = TransactionEncoder()
  te_ary = te.fit(namespace['par_%s' % i]).transform(namespace['par_%s' %
  i])
  df = pd.DataFrame(te_ary, columns=te.columns_)
  s="texte_{v1}".format(v1=str(i))
  if i==0:
    textes = {s:df}
  else:
    textes[s] = df
return textes
```

Code de vectorisation pour itemsets fréquents à partir des transactions

```
namespace=globals()
def apriori_all(sup):
  for l in range (len(data)):
    for j in range (l,l+1):
      a=apriori(data["texte_{vl}".format(vl=str(j))], use_colnames=True,
    n_jobs=-1)
      a['length'] = a['itemsets'].apply(lambda x: len(x))
      a['length']=pd.DataFrame(a['length'])
      a=  a[ (a['support']>=sup)]
      items_dict = {}
      for i in range(len(a["itemsets"])):
        items_dict[a["itemsets"][i]]=1
      namespace['df_%s' % j] = pd.DataFrame(items_dict, index=[0])
Data_item = df_0.append(df_1, sort=False)
for i in range(2,len(data)):
  Data_item = Data_item.append(namespace['df_%s' % i], sort=False)
Data_item.index = pd.RangeIndex(len(Data_item.index))
Data=Data_item.fillna(0)
Data = Data.astype(int)
```