UNIVERSITÉ DE SAAD DAHLEB -1-
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
SPÉCIALITÉ : INGÉNIERIE DES LOGICIELS

MÉMOIRE DE MASTER

Fouille automatique des publications dans les réseaux sociaux : Application sur Twitter

Réalisé par

Reguieg Faïrouz

Devant le jury composé de :

M.CHIKHI Nacim	Université de Blida	*Président*
M.BALA Mahfoud	Université de Blida	*Examinateur*
M.CHEMCHEM Med Lamine	Université de Blida	*Promoteur*
MME.OUKID Lamia	Université de Blida	*Co-Promotrice*

Soutenu le 27/06/2018, Blida

UNIVERSITÉ DE SAAD DAHLEB -1-
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
SPÉCIALITÉ : INGÉNIERIE DES LOGICIELS

MÉMOIRE DE MASTER

Fouille automatique des publications dans les réseaux sociaux : Application sur Twitter

Réalisé par

Reguieg Faïrouz

Devant le jury composé de :

M.CHIKHI Nacim	Université de Blida	*Président*
M.BALA Mahfoud	Université de Blida	*Examinateur*
M.CHEMCHEM Med Lamine	Université de Blida	*Promoteur*
MME.OUKID Lamia	Université de Blida	*Co-Promotrice*

Soutenu le 27/06/2018, Blida

RÉSUMÉ

Ces dernières années, les réseaux sociaux sont devenus le moyen le plus utilisé pour le partage des informations entre les internautes. Dans ce projet, nous présentons des approches d'exploration et d'apprentissage automatique pour la catégorisation des textes courts publiés sur les réseaux sociaux (Twitter), dans le but de réaliser un système de classification selon les thèmes et/ou selon les centres d'intérêt. Une autre application fort intéressante de cette étude est la découverte des propos raciste, menaçants, et les communautés cachées.
Pour la réalisation de ce projet, des méthodes de classification classiques, et des approches d'apprentissage automatique ont été implémentées et adaptées pour traiter les textes de tweeter, afin de comparer leurs performances sur plusieurs jeux de données publiques. De plus, une nouvelle hybridation entre les approches promettantes a été proposée.
Les expériences menées sur différents corpus ont abouti à des résultats satisfaisants dans la majorité des cas.

Mots clés : Text mining, Twitter, Apprentissage automatique, Segmentation, Topic modeling, Classification.

ABSTRACT

THIS last years, social networks have become the most used means for sharing information between Internet users. In this project, we present approaches of exploration and machine-learning for short text categorization published on social networks (Twitter), with the aim of achieving a classification system according topics (themes) and/or according to centers of interest. Another very interesting application of this study is the discovery of racist, threatening, and hidden communities. For the realization of this project, classical classification methods, and automatic learning approaches were implemented and adapted to process on tweets texts, in order to compare their performances on several public datasets. In addition, a new hybridization between promising approaches has been proposed.
The experiments carried out on different corpus have led to satisfying results in most cases.

Keywords : Text mining, Twitter, Machine learning, Clustering, Topic modeling, Classification.

ملخص

في السنوات الأخيرة ، أصبحت الشبكات الاجتماعية أكثر الوسائل المستخدمة لتبادل المعلومات بين مستخدمي الإنترنت. في هذا المشروع ، نقدم أساليب الاستكشاف والتعلم الآلي لتصنيف النصوص القصيرة المنشورة على الشبكات الاجتماعية (ةوِتتر) ، بهدف تحقيق نظام تصنيف وفقاًا للمواضيع و\أو وفقاًا للاهتمام. تطبيق آخر مثير للاهتمام للغاية من هذه الدراسة هو اكتشاف عنصري، تهديد، والمجتمعات المخفية.لتحقيق هذا المشروع ، تم تطبيق أساليب التصنيف الكلاسيكية ، لتعلم التلقائي وتكييفها لمعالجة نصوص التويتر ، من أجل مقارنة أدائها في العديد من مجموعات البيانات العامة. بالإضافة إلى ذلك ، تم اقتراح تهجين جديد بين الأساليب الواعدة.

وقد أدت التجارب التي أجريت على البيانات المختلفة إلى نتائج مرضية في أغلب الحالات.

الكلمات الرئيسية : تويتر ، التعليم الالي ، التجميع ،التصنيف.

DÉDICACES

À mes parents...

Remerciments

Mes remerciements sincères et profonds s'adressent à mon promoteur M. CHEMCHEM Med Lamine. Qu'il trouve ici l'expression de ma parfaite gratitude pour sa patience, son soutien et surtout pour ses conseils et orientations qui m'ont été précieux afin de mener ce travail à bon port.

Je tiens à exprimer ma reconnaissance à Mme. OUKID Lamia d'avoir acceptée de co-diriger ce travail.

Mes vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à cette recherche en acceptant d'examiner mon travail et de l'enrichir par leurs propositions.

Que mes parents et mes amis trouvent ici l'expression de mes sincères remerciements pour leurs encouragements et leur support.

Enfin, je tiens également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

TABLE DES MATIÈRES

Liste des tableaux

Table des figures

INTRODUCTION GÉNÉRALE

Avec l'évolution des technologies du Web, des capacités de stockage et d'échanges sur Internet durant cette dernière décennie, les réseaux sociaux sont devenus un élément majeur des internautes. Ce qui a poussé à une explosion en termes de volume de données.

En effet, l'utilisation quotidienne des réseaux comme Twitter ou Facebook a changé l'image du web et lui a donné une nouvelle dimension.

Twitter est un réseau social de type microblogage géré par l'entreprise Twitter Inc. Créé en mars 2006 et lancé en juillet de la même année avec un slogan « *What are you doing ?* » (Qu'est-ce que vous faites ?), ensuite changé vers « *What's happening ?* » (Quoi de neuf ? ou encore Que se passe-t-il ?). Sa mascotte est un oiseau stylisé, nommé Larry en hommage au basketteur américain "Larry Bird".
Il permet aux utilisateurs de raconter leurs vies, partager leurs expériences, leur émotions, leurs avis de manière brefs. Ces courts messages sont appelés tweets.

Ces messages sont limités à 280 caractères. Les utilisateurs sont dotés d'un identifiant et comme dans les autres réseaux sociaux il est possible d'utiliser un mot-dièse (hashtag) pour annoter le tweet avec un mot-clé de contexte.

Selon les statistiques officielles faites par l'entreprise Twitter, ce service web compte actuellement 330 millions d'utilisateurs actifs par mois avec 500 millions de tweets envoyés par jour. Environs 6.000 Tweets par seconde et est disponible en plus de 40 langues.

Avec cet ampleur importante de données, Twitter est devenu une cible de recherche très attractive pour les agences de presse, les entreprises et notamment les chercheurs en informatique et science de l'information. Dont leur principal but est l'étude du marché, l'analyse des tendances et comportement humain, l'identification des propos racistes, des personnes influentes, ... etc.

Les posts publiés sur Twitter reflètent l'interaction des utilisateurs avec les évènements réels qui se déroule dans le monde, tel que les élections, les évènements culturels, les catastrophes naturelles... etc.
Les humains ont la capacité, en s'aidant du contexte, à identifier et désambiguïser sans trop d'efforts ces posts publiés en des langues naturelles.
Toutefois, pour le traitement automatique des langues naturelles, cette ambiguïté pose problème, et il est fondamental de trouver des méthodes pour affecter aux mots les sens corrects vis-à-vis du contexte.

L'obstacle majeur à faire face dans ce projet, c'est de pouvoir remédier à cette ambiguïté, et de s'inspirer de la robustesse des méthode de fouille de données et d'apprentissage automatique pour la réalisation d'un système performant de classification automatique des tweets.
Un autre problème traité par cette étude est le passage à l'échelle. En effet, comme évoqué précédemment, le nombre de tweets ne cessent d'augmenter, et les approches présentées par la littérature ont montré leur inefficacité lors du traitement de grande base de données.

Dans cette étude, nous passerons en revue les méthodes existantes de fouille et d'analyse des tweets. Nous réaliserons par la suite, un système de classification selon les thèmes et/ou selon les centres d'intérêt. Cela, en hybridant certaines approches existantes et/ou en proposant des nouvelles. Ces approches sont enfin implémentées et comparées selon leurs performances. L'objectif principal de cette étude est de classifier/segmenter les tweets de manière efficace et rapide. Une application forte intéressante de ce projet, est la détection des propos racistes, détection de communautés cachées... etc.

Dans le cadre de ce projet, nous avons choisi de travailler sur des tweets en langue anglaise.

La suite du projet est organisée comme suit : Tout d'abord, dans le premier chapitre nous passerons en revue les travaux connexes à notre projet/problématique.

Puis, dans le deuxième chapitre nous présentons les procédés de pré-traitement du langage naturel pour nettoyer les tweets. Suivi par la conversion du texte des tweets vers des vecteurs numériques via plusieurs approches.

Par la suite, dans le troisième chapitre, nous exposons nos contributions concernant les méthodes non-supervisées (clustering/topic modeling) et les approches supervisées sur des collections de données réelles de grande taille.

En dernier lieu, dans le quatrième chapitre, nous avons réaliser une étude comparatives de ces approches selon les deux critères « temps d'exécution » et « le taux de réussite » de la classification.

Chapitre 1

ÉTUDES ET TRAVAUX CONNEXES

Introduction

Dans ce chapitre nous allons passer en revue quelques travaux qui se rapproche à notre projet/problématique. Par la suite, nous exposons les différentes stratégies et techniques de représentation du texte dans la littérature et nous allons conclure ce chapitre par une étude comparative entre les algorithmes de classification.

1.1 Travaux connexes

Publications mining, à savoir l'analyse et exploitation des posts dans les réseaux sociaux est une discipline qui fait la une en informatique. Cela dit, Twitter représente une importante et une vraie mine d'or de source d'informations.

Les recherches les plus populaires dans ce cadre de cette problématique sont généralement consacrées à l'analyse des sentiments [Anumol and P., 2016] la détection des catastrophes naturelles, tel que l'ouragan de Sandy , évènements sociopolitiques à titre d'exemple : the Arab Spring [Kumar et al., 2014]...etc.

Différentes méthodes pour l'analyse des posts des réseaux sociaux et l'extraction des informations pertinentes ont été proposées dans la littérature. Ces méthodes se divisent en deux catégories principales : d'une part les approches supervisées, nécessitant des corpus d'entraînement étiquetés et, d'autre part, des approches non-supervisées qui exploitent des données non annotées.

1.1.1 Approches non-supervisées

Dans [Popovici et al., 2014], [Ifrim et al., 2014], les auteurs ont suivi un processus très similaire, qui commencent par l'étape de pré-traitement des publications afin de les nettoyer comme la suppression de la ponctuation,etc pour ensuite effectuer l'étape de segmentation (clustering).

Pour le ciblage publicitaire [Friedemann, 2015] propose une technique pour l'extraction des caractéristiques (features) qui se basent non seulement sur le tweet (texte) lui même mais aussi, sur le nombre des abonnées (followers), puis faire un regroupement en utilisant « la distance Euclidienne » avec l'algorithme « K-means » [Forgy, 1965].

Dans [Soni and Mathai, 2015], un modèle « cluster-then-predict » a été proposé pour l'amélioration de la prédiction des sentiments (Twitter sentiment) via une composition des deux approches supervisée et non-supervisée.
Leur contribution se résume à extraire les caractéristiques des tweets dans un premier temps, ensuite appliquer l'algorithme « K-means » sur l'ensemble de données tel que les tweets avec des mots similaires sont regroupés. Après cette phase non-supervisée, une classification supervisée avec les algorithmes « Arbre de décision » et « Machine à vecteurs de support » a été effectué sur les mêmes données. D'après les résultats obtenus, le modèle proposé a pu améliorer la prédiction des sentiments sur Twitter.

L'étude menait par [Ifrim et al., 2014], pour la détection des thématiques « Topics detection » sur Twitter, se base sur une méthode de filtrage strict sur les tweets/termes avec l'utilisation du « Clustering hiérarchique » (CHA) [Székely and Rizzo, 2005]. Leur contribution est comme suit ; d'abord calculer la distance entre les tweets en utilisant la mesure de similarité « cosine ». Ensuite, appliquer l'algorithme hiérarchique de tel sorte que les tweets appartenant à la même thématique (topic) seront regrouper dans le même cluster.

En outre chaque cluster est considéré comme une thématique et/ou un sujet détecter. Le point faible de ce travail est la phase de filtrage strict, car ce dernier peut conduire à une perte de valeur d'une grande partie de données qui peuvent s'avérer être utiles.

Un travail a été réalisé par [Purwitasari et al., 2015], dont le but de résumé les nouveautés sur Twitter (News summary). Les auteurs ont choisis de négliger les hashtags et de garder les retweets. Ensuite, ils ont calculé la fréquence des termes dans un tweet puis appliquer l'algorithme « K-medoids » [Kaufman and Rousseeuw, 1987] pour le regroupement (clustering). Les résultats obtenus n'étaient pas satisfaisant et cela à cause de l'inclusion des retweets qui ont affecté la qualité du clustering.

Ici, on peut déduire qu'il faut éliminer les retweets [1] avant la phase d'extraction des features (caractéristiques) et par conséquent avant le clustering pour ne pas avoir des clusters contenant des tweets redondants.

Un autre travail a été réalisé par [Boom et al., 2015] dans le but de la détection des évènements et cela en se basant seulement sur les hashtags dans les tweets. Les auteurs ont regroupé les co-occurrence des hashtags en utilisant l'algorithme de densité « DBSCAN » (Density-Based Spatial Clustering of Applications with Noise) [Ester and P., 1996].
Leur méthode requit deux paramètres : un nombre minimal des hashtags par cluster et une distance de similarité minime entre deux hashtags. Cette dernière est calculée comme suit : $\Sigma(\textit{des occurrences de deux hashtags par jour})/2$. D'après les résultats obtenus, il y a eu un progrès lors de la détection des événements.

Une étude récente [Anumol and P., 2016], ils proposent l'application de « DBSCAN » (Density-Based Spatial Clustering of Applications with Noise) pour la représentation des segments significatifs des tweets en « batch mode » (par paquet/lot) lors de l'analyse des sentiments. Dans cette approche, leur ultime but de clustering est d'intégré « DBSCAN » avec « Jaccard» comme mesure de similarité.
Les résultats ont indiqué une amélioration du regroupement à la suite de l'intégration de l'algorithme « DBSCAN ».

1. Republication d'un tweet d'une personne x sur son propre compte Twitter.

Le point fort de ces deux travaux est l'utilisation des algorithmes à base de densité, qui peuvent être particulièrement adaptés pour regrouper des données non structurées comme les tweets. Et ne nécessitent pas une initial identification du nombre de clusters.

Cependant, leur point faible est qu'ils nécessitent un grand espace mémoire lors du traitement d'un ensemble de données volumineux.

Topic modeling

Dans [REN et al., 2016], les auteurs proposent une méthode pour la classification des sentiments sur Twitter en se basant sur l'algorithme du topic modeling « LDA » (Latent Dirichlet Allocation) [Blei et al., 2003] pour générer une distribution thématique (topic) des tweets et « SVM » (Support Vector Machine) pour la classification.

Leur expérimentations ont montré que le modèle proposé a pu atteindre un taux de réussite de 81.02% sur l'ensemble de données : « SemEval-2014 from Twitter Sentiment Analysis ».

Une approche pour l'extraction des thématiques (topics) basée sur « LDA » a été proposé par [TAN et al., 2014], ils se sont focalisés sur le suivi et la modélisation (topic modeling) des sentiments sur Twitter. Ils ont notamment proposé une autre méthode pour classer un ensemble de candidats nommé « Reason Candidate and Background LDA » (RCB-LDA).

Leurs résultats ont montré que leurs modèles peuvent être utilisés pour identifier des sujets spéciaux et trouver différents aspects.

Les auteurs de [WANG et al., 2012], avaient pour but de prédire les incidents criminels sur Twitter. Ils ont suggéré une approche pour l'analyse et la compréhension des tweets en se basant sur un modèle probabiliste « LDA » ainsi qu'un modèle de régression linéaire.

Leur évaluation a montré que cette approche à la capacité de prédire les délits (hit-and-run crimes) seulement en utilisant les informations existantes dans leur jeu d'entrainement.

Une autre étude intéressante de [WANG et al., 2016], cette fois, ils ont examiné les caractéristiques des abonnées de Trump (président actuel des États-Unis) sur Twitter, ensuite ils ont proposé un model de régression pour modéliser les "likes" (j'aime) et extraire les tweets de Trump via « LDA ».

Leur évaluation a été effectuer sur l'ensemble de données « US2016 » (Twitter) qui contenait le nombre des abonnées pour tous les candidats à l'élection présidentiel des États-Unis en 2016 . Les auteurs ont démontré que l'intégration de la notion de thématiques (topics) (via LDA) est très impressionnante pour la classification des sentiments sur Twitter.

Dans [CHEN et al., 2015], ils ont présenté un système d'alertes précoce pour détecter les intentions des activités criminelles en se basant sur l'algorithme « LDA » et «CRC» (Collaborative Representation Classifier).

Leur système inclut deux phases : La première consiste a utilisé « LDA » pour l'apprentissage des caractéristiques (features) et extraire celles qui sont pertinentes depuis leur dataset qui est un ensemble d'articles de chez Yahoo. Et pour la deuxième phase, ils ont utilisé « CRC » pour classifier de nouveaux documents en se basant bien-entendu sur les caractéristiques extraits via « LDA ».

Un modèle statistique basé sur « LDA » a été proposé par [GERBER, 2014], dont le but d'identifier les thématiques des discussions dans la ville de Chicago, ainsi que l'utilisation des techniques « KDE » (Kernel Density Estimation) pour la détection de crimes.

Un autre travail réalisé par [SHARMA et al., 2015], où les auteurs ont présenté une approche basée sur le modèle géographique des intensités de criminalité pour détecter le chemin le plus sûr entre deux endroits et cela en appliquant le classificateur « Naïve de Bayes » sur les caractéristiques dérivées d'un modèle «LDA».

D'après les travaux cités sur le topic modeling, on peut en déduire que la modélisation thématique est l'une des techniques les plus puissantes dans le text mining (exploration de texte) et par conséquent pour le data mining (exploration des données). Ses méthodes et algorithmes aident a la découverte des connaissances cachées, trouver des relations parmi les données, documents ...etc.

1.1.2 Approches supervisées

D'autres travaux ont été réaliser par l'emploi des réseaux de neurones.

Les auteurs dans [DeMello. et al., 2005] ont présenté une méthode pour la classification automatique du texte en utilisant un réseau de neurones de type « ART-2A » (Adaptative Resonance Theory), un algorithme pour l'apprentissage rapide des catégories et reconnaissance.

Leur contribution se résume en trois phases : La première étape consiste à l'extraction des caractéristiques via le calcul de la fréquence du radical d'un mot dans un document. En deuxième étape ; utiliser une architecture réseau de neurones de type « ART-2A » pour classifier les vecteurs de fréquence des mots pour chaque document. En dernière étape, pour chaque groupe généré par le réseau de neurones, extraire les mots signifiants et cela avec une mesure de distance proposé pour automatiser le paramètre de vigilance, qui est responsable de la qualité de la classification.

Le point fort de cette approche est l'élimination de l'intervention humaine pour le processus du paramétrage et cela via la mesure qui a été suggérer.

Une classification des phrases via « CNN » (Convolutional Neural Networks) a été faite par [Yoon, 2014]. Il a suggéré un model « CNN » basé sur « Word2Vec » ce dernier est un groupe de modèles, des réseaux de neurones artificiels à deux couches développés par (Mikolov et al, 2013). Le modèle proposé par Kim Yoon, a été testé sur plusieurs benchmark dont la tâche était de détecter la présence des sentiments dans un tweet (positive/négative).

Les résultats de nombreux tests présentés dans le document montrent que, après un petit réglage des hyper-paramètres, le modèle donne d'excellents résultats suggérant que les vecteurs pré-entrainer (Word2Vec) sont des extracteurs de caractéristiques universels et peuvent être utilisés pour diverses tâches de classification.

Dans l'article [Siwei et al., 2015] les auteurs ont proposé un modèle qui regroupe «CNN » et « RNN », respectivement (Convolutional Neural Networks) et (Recurent Neural Networks) pour la classification textuelle.

Leur modèle a pu capturer des informations contextuelles avec la structure récurrente et construire la représentation de texte en utilisant un réseau de neurones convolutionnels. Après l'avoir tester sur quatre ensemble de données différents, les résultats obtenus ont démontré que leur modèle surpasse « CNN » et « RNN ».

1.2 Pré-traitement

Les auteurs dans [Ozdikis et al., 2012] ont proposé une méthode de détection d'évènements en utilisant une approche basée sur une expansion lexico-sémantique (lexicale et sémantique) sur le contenu des tweets.
Cette technique d'expansion sémantique qu'ils ont mit en œuvre est basée sur des approches syntagmatiques et relations paradigmatiques entre les mots extraites de leurs statistiques de co-occurrence. Elle est intégrée dans la phase de pré-traitement et permet à des mots comme "alexsandra" et "alexandra" d'être considérés comme un mot identique.
Les résultats expérimentaux décrit dans l'article montrent que l'utilisation de cette technique apporte plus de précision.

D'autres techniques simples et intuitives ont été évoquées dans les articles [Gao et al., 2014] et [Han et al., 2015] pour l'étape de nettoyage tel que, l'enlèvement des points et virgules, la suppression des identifiant Twitter et notamment celle des liens.

Au final, nous avons retenu l'utilisation des types de nettoyage simple mentionnés précédemment, ainsi qu'ajouter d'autres filtrages qu'on expliquera en détails dans le chapitre suivant.

1.3 Représentation du texte

La majorité des algorithmes prennent comme entrée des vecteurs numériques et non pas un texte brut. De ce fait, il est primordial de trouver une transformation adéquate et représentative qui convertit le texte des tweets vers des vecteurs numériques.

L'une des méthodes les plus connues est la transformation « Bag-Of-Words » (sac à mots). Dans la littérature les approches qui permettent ce passage du texte vers des « Bag-Of-Words » se divisent en trois catégories.

La première catégorie est une approche statistique basée sur l'occurrence des mots/termes à titre d'exemples : « TF » (Term Frequency) et « TF-IDF » (Term Frequency-Inverse Document Frequency) [Jones and K., 1972].

La deuxième est une approche lexicale et similitude basée sur des ressources lexicales telle que « WordNet ».

La troisième catégorie est l'ensemble des méthodes de la famille « N-gram » pour désigner des séquences de mots.

L'utilisation de l'approche statistique comme le « TF-IDF » a été appliqué sur des tweets comme mentionné dans [Kathy et al., 2011], pour sa simplicité mais aussi convient parfaitement au type de données tel que Twitter où l'importance du mot vient de sa fréquence d'emploi.

Une méthode pour combiner l'approche statique et connaissances lexicale a été proposé par [Fodeh et al., 2009] dans le but du regroupement (clustering) des documents.
Cette méthode fusionne deux sources d'informations : Des informations statiques dérivées depuis l'ensemble de données et le sens des mots retiré via « WordNet » pour la construction d'une matrice (modèle) sémantique binaire. Par la suite, l'algorithme « K-means » est appliqué pour le clustering.

Des tests expérimentaux ont été effectués sur deux benchmarks (20 NewsGroups) et (Routers), les auteurs ont conclus que le remplacement des mots par leur sens n'improuve pas forcément le résultat (qualité) du clustering.

Le point fort de cette contribution est que la méthode qu'ils ont proposé peut être efficace sur certains ensembles de données.

Comme points faibles, on déduit que l'augmentation du nombre des caractéristiques lors du remplacement des mots par leur sens mène à une dégradation sur les résultats obtenus. La transformation des mots en leur sens ne suffit pas à améliorer le regroupement, en réalité, ça peut empirer à cause de l'augmentation de la dimension des données, ou quand le mauvais sens est choisi pour remplacer certains mots.

Selon ce qui a été évoqué précédemment, nous avons décidé d'utiliser les méthodes des deux approches, statistique et lexicale. Dans le chapitre suivant nous exposons en détails les méthodes choisies.

1.4 Clustering/Classification

D'après les articles cités au début de ce chapitre, les algorithmes utilisés par les chercheurs dans les travaux connexes sont des algorithmes de regroupement (clustering), plus précisément, des algorithmes de type partitionnement comme : « K-means » et « K-medoid », d'autres basé sur la densité tel que : « DBSCAN » ainsi que ceux pour le Topic modeling, par exemple : « LDA ». D'autres ont préféré d'employer des algorithmes de classification à titre d'exemple : « SVM » et « Les réseaux de neurones artificiels ».

Le Tableau 1.1, représente les avantages et faiblesses des réseaux de neurones artificiels (approche supervisée) que nous utiliserons lors de nos expérimentations.

Pour le deuxième Tableau 1.2, il représente une étude comparative entre les algorithmes de clustering (approche non-supervisée) dont nous allons nous en servir.

Tableau 1.1 – Forces et Faiblesses des réseaux de neurones artificiels.

	Forces	Faiblesses/Limites
Réseaux de neurones	• Relativement facile à utiliser. • Peut approximer n'importe quelle fonction, indépendamment de sa linéarité. • Idéal pour les problèmes complexes/ abstraits comme la reconnaissance d'images, sons et texte. • Imite le cerveau (à un certain degré).	• Souvent abusé dans les cas où des solutions plus simples comme la régression linéaire serait la meilleure. • Nécessitent une grande base d'entrainement et différents cas. • Peuvent être difficiles à régler pour s'assurer qu'ils apprennent bien, et donc difficile à débugger. • Difficile à interpréter, ils se transforment en boite noire une fois entrainé.

Tableau 1.2 – Comparaison des méthodes non-supervisées.

	Forces	Faiblesses/Limites
K-means	• Relativement extensible pour traitement de grands ensemble de données. • Relativement efficace: O(tkn), où n objets, k clusters et t itérations. • Relativement rapide par rapport au clustering hiérarchique et a base de densité.	• Spécification du nombre de clusters. • Les clusters sont construits par rapport à des centres (objets) inexistants dans les données. • Si on enlève des données ou on rajoute d'autres il faut refaire tout le calcul.
LDA	• Utilise la loi de Dirichlet priors pour la distribution document-thème(sujet) et thème(sujet)-mot. • Empêche l'over-fitting (sur-ajustement).	• Il devient incapable de modéliser les relations entre les thèmes (sujets). • Spécification du nombre de topics (clusters).
LSI	• Peut gérer le problème de la synonymie. • Mappe les documents dans un espace de faible dimension. • Décompose la matrice document-termes ce qui le rend plus rapide par rapport à d'autre modèles de réduction dimensionnelle.	• Ne gère pas de manière efficace le problème de la polysémie. • Spécification du nombre de topics (clusters). • Dépend fortement de SVD (décomposition en valeurs singulières) qui s'avère difficile à mettre à jour lors de l'apparition d'un nouveau document.

Nous avons donc opté pour l'utilisation de chacun des algorithmes suivants :
« K-means », « LDA », « Réseaux de neurones » et en rajoutant l'algorithme « LSI » qui appartient à l'approche non-supervisée, plus exactement le Topic modeling.
Ce choix à pour objectif d'exploiter les deux approches non-supervisée et supervisée de l'apprentissage automatique afin d'avoir une vision concis sur leur points forts et faiblesses en les appliquant sur notre flux de travail.

Conclusion

Dans ce chapitre nous avons présenté plusieurs travaux connexes à notre thème qui est *la fouille automatique des publications dans les réseaux sociaux.* Suivie par des techniques de conversion de texte et un comparative sur les algorithmes à utiliser.

Le chapitre suivant va traiter en détails les deux principales étapes a effectué avant chaque segmentation/classification qui sont le pré-traitement et la conversion du texte.

CHAPITRE 2

PRÉ-TRAITEMENT ET CONVERSION DES TWEETS

Introduction

Pour pouvoir appliquer les méthodes de la classification non-supervisée et/ou supervisée sur un flux textuel, il faut que ce dernier soit bien nettoyé et représenté sous une forme la plus optimale, afin de l'exploiter et en extraire le maximum de caractéristiques.

2.1 Pré-traitement

L'analyse des messages publiés sur Twitter représente un vrai challenge à cause de leur nature qui n'est pas structurée orthographiquement et grammaticalement. Les tweets sont totalement différents par rapport à d'autres documents comme les articles des journaux, les discours officiels, les pages web, etc où le lexique et la syntaxe sont contrôlés.

Parmi les particularités qu'on peut trouver dans les tweets, nous citons :

- Le langage utilisé n'est pas formel, et un mélange entre l'argot, abréviations et plusieurs langues dans le même tweet.
- L'existence des liens et des identifiants compliquent l'opération d'analyse.
- Les tweets sont pleins d'erreurs d'orthographe, lexicales et syntaxiques.

Pour ces raisons, nous avons décidé de nettoyer et d'appliquer un filtrage sur les tweets comme suit :

- **Supprimer les liens-hypertexte :** Dans ce contexte, les liens n'ont pas d'importance car un lien dans un tweet n'a pas un poids sémantique.
- **Supprimer les identifiants :** Un identifiant Twitter est précédé par un arobase "@"et n'est pas un mot-clé qui concerne un thème (topic). Il représente juste l'utilisateur sur la plateforme. Exemple : *@thedeatheaterr.*
- **Suppression des retweets :** On rappelle, un retweet est l'action de publier un tweet existant, souvent précédé par un mot réserver "RT". Leur suppression est inévitable pour ne pas avoir de redondance.
- **Suppression des emoji :** Les emoji, émoticônes en français, sont des images miniatures qui permettent de communiquer brièvement, à l'écrit, une information comparable à une expression faciale. Dans ce contexte, on travaille sur du texte, donc il faut enlever ces images.
- **Hashtags :** Pour les hashtags, on supprime le signe dièse "#" tout en gardant le mot qui a été annoté, car ce dernier fait toujours partis du champs lexical d'un thème X.
  Exemple : $\#basketball \rightarrow basketball \Rightarrow$ *thème sport.*
- **Suppression des chiffres :** Le nettoyage des chiffres n'est pas aussi simple, leur utilités varie selon les contextes à traiter. Dans cette étude, on a opté pour leur élimination et de travailler uniquement sur le langage naturel.
- **Ponctuations :** Un autre type de traitement qui consiste à nettoyer le texte par la suppression des guillemets, des points, des virgules, des points d'exclamation, etc.

- **Supprimer les Stop words :** Les mots vides (Stop words) sont des mots qui n'apportent pas sens qu'il est inutile de les indexer ou de les utiliser dans une recherche. Exemples de mots vide en anglais : *« And », « That », « Was », etc.*

- **Supprimer les mots courts :** Éliminer les mots qui sont de taille inférieur ou égale à deux caractères. Exemples : *« Ah », « Oh », etc.*

- **Lemmatisation :** Ce traitement consiste à trouver la forme neutre canonique. Exemple : *car, cars, car's, cars'* $\Rightarrow$ *car.*

- **Majuscule/Minuscule :** Pour résoudre le problème des majuscules et des minuscules on a décidé de convertir tous les mots en minuscule. Et cela pour éviter qu'un mot comme "Home" et "home" soient considérés comme des mots différents.

L'entrée de cette étape est la liste des listes des mots bruts extraits de chaque tweet. En sortie de cette étape nous avons, une liste des listes des mots nettoyés. Si le nettoyage supprime tous les mots d'un tweet, celui-la sera supprimé.

2.2 Représentation du texte

D'après les travaux existants et ce dont nous avons évoqués dans la partie d'état d'art, nous rappelons que les algorithmes des deux approches non-supervisée et supervisée ne prennent pas directement du texte mais des vecteurs numériques. C'est pour cela qu'il faut effectuer une transformation du texte.

Cette étape prends comme entrée :

- La liste des mots uniques existant dans le flux à analyser après pré-traitement.
- La liste des listes des mots de chaque tweet après le pré-traitement.

En sortie, elle fournit des vecteurs numériques correspondants à chaque tweet.
Nous avons employé différentes méthodes pour créer ces vecteurs et cela sur la base de la technique « Bag-Of-Words », plus précisément en utilisant les deux approches statistique et lexicale.

2.2.1 Approche statistique et/ou terminologique

➢ **Occurrence :** Il s'agit de la formule la plus simple et intuitive, qui se base sur le nombre de fois où apparaît le terme. Le vecteur en sortie a comme taille le nombre de mots uniques du flux étudié. La valeur de chaque élément correspond au nombre d'occurrences du mot dans le tweet, pour les mots qui ne sont pas présents dans le tweet leur valeur est égale à 0. Ce qui nous pousse à utiliser une représentation éparse (représenter que les valeurs non nulles) de ces vecteurs pour pouvoir les stocker en mémoire et accélérer les opérations algébriques matrice/vecteur.

➢ **TF-IDF :** La formule TF-IDF (Term Frequency-Inverse Document Frequency) est une méthode de pondération souvent utilisé dans la recherche d'information et en particulier dans la fouille et l'exploration de texte (Text Mining). Cette mesure statistique permet d'évaluer l'importance d'un mot pour un document dans une collection ou un corpus. L'importance augmente proportionnellement au nombre de fois qu'un mot apparaît dans le document. Elle varie également en fonction de la fréquence du mot dans le corpus.

Autrement dit, cette formule reflète l'importance du mot dans le document avec le TF et son importance dans le corpus par l'IDF. La valeur finale présente le rapport entre les deux. Il existe plusieurs variation de cette formule, nous avons opté pour l'utilisation de la suivante [Jones and K., 1972] :

La formule TF-IDF d'un terme t est :

$$TF - IDF(t) = TF(t) \times IDF(t) \tag{2.1}$$

$$\equiv$$

$$TF - IDF(t) = \left(\frac{a}{b}\right) \times \left(1 + \log\left(\frac{c}{d}\right)\right) \tag{2.2}$$

Où :

- a : Le nombre d'occurrences du terme t dans le tweet.
- b : La taille de tweet (en nombre de mots).
- c : Le nombre total des tweets.
- d : Le nombre de tweets où ce terme t est présent.
- L'ajout du "1" empêche les divisions nulles.

Ensuite, on a appliqué la normalisation *L2-norm* connue aussi sous le nom de la *normalisation Euclidienne.* Chaque vecteur est normalisé de tel sorte que : Si on met au carrée chaque élément du vecteur résultat (v_{norm}) et on les additionne on obtiendra un 1.

La formule de normalisation d'un vecteur v est :

$$v_{norm} = \frac{v}{||v||^2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}} \tag{2.3}$$

Où :

- v : Le vecteur calculer via TF-IDF.

➢ **Similarité cosinus :** La similarité cosinus (ou mesure cosinus) permet de calculer la similarité entre deux vecteurs à n dimensions en déterminant le cosinus de l'angle entre eux. Cette métrique est fréquemment utilisée en fouille de textes.

La similarité résultante est comprise dans l'intervalle [-1,1], tel que la valeur -1 indique des vecteurs opposés, 1 des vecteurs similaires, 0 signifie des vecteurs indépendants (dé-corrélation), et les valeurs intermédiaires indiquent le degré de similarité ou dis-similarité.

L'angle θ s'obtient par le produit scalaire et la norme des vecteurs comme suit [Giller, 2012] :

$$similarité\ cosinus = \cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.4}$$

Où :

- A_i *et* B_i : Des éléments du vecteur A et B respectivement. Les valeurs de ces vecteurs sont le nombre d'occurrences d'un terme t dans le tweet.

➢ **Hybridation TF-IDF x Cosinus :** La combinaison de TF-IDF avec Cosinus, consiste à calculer les éléments des vecteurs avec la formule *TF-IDF (2.2)* ensuite appliquer la *similarité cosinus (2.4)* sur ces vecteurs (comme une méthode de normalisation).

La similarité résultante, ici, sera comprise dans l'intervalle [0,1], puisque la fréquence du terme (poids TF-IDF) ne peut pas être négatif. L'angle entre deux vecteurs fréquentiels à terme ne peut pas être supérieur à 90°.

2.2.2 Approche lexicale et/ou structurelle

Un grand nombre de techniques de désambiguïsation lexicale ont été présentées dans la littérature. Par ailleurs, des ressources lexicales telles que *WordNet* ([MILLER, 1995]) sont structurées et jouent le rôle d'inventaires de sens et de dictionnaires, mais donnent également accès à une hiérarchie de sens. Autrement dit, un thésaurus structuré.

➢ **WordNet :** Wordnet est une ressource lexicale de la langue anglaise. Il est structuré autour de la notion de *synsets*, c'est-à-dire il regroupe des termes (noms, verbes, adjectifs et adverbes) en un ensemble de synonymes qui forment et dénotent un concept.

Les **synsets** sont reliés entre eux par des relations, soit lexicales (*antonymie*, *synonymie* par exemple), relation est-un (*is-a*), ou taxonomiques (*hyperonymie/hyponymie*, *holonyme/méronyme*, etc).

La Figure (2.1) ci-dessous montre un sous-graphe de WordNet.

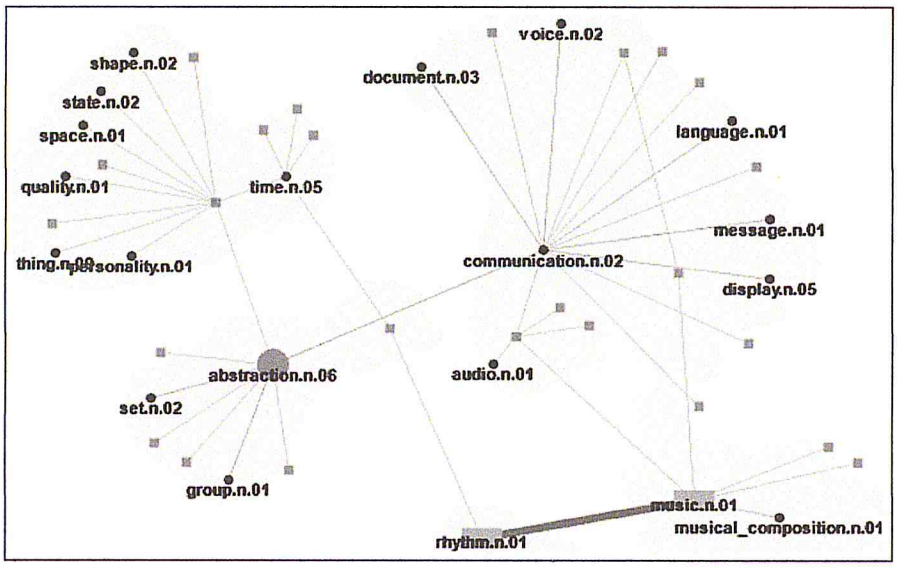


FIGURE 2.1 – Sous-graphe de Wordnet.
Source : ([Wang. et al., 2015])

La **taxonomie** est un ensemble de classes muni d'une relation de généralité. Une opération courante pour la réalisation d'une classification hiérarchique.

La Figure (2.2) ci-dessous montre une partie d'une taxonomie.

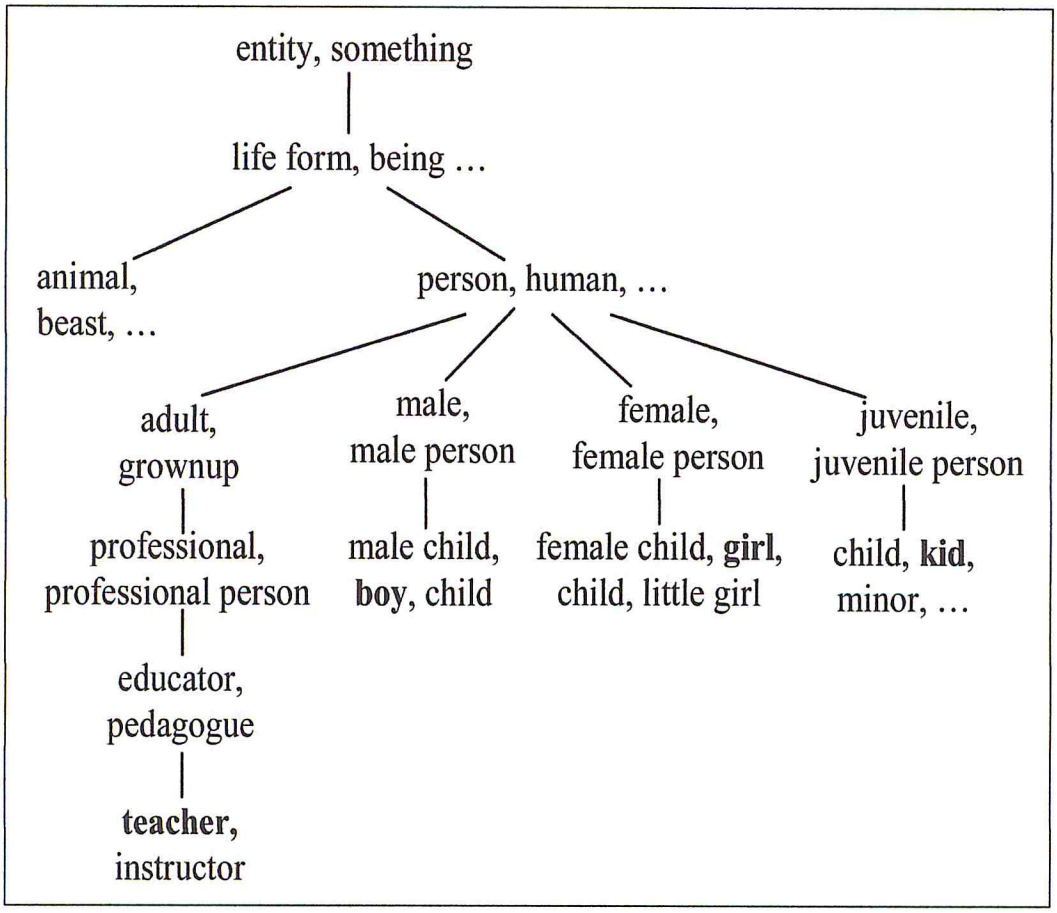


FIGURE 2.2 – Exemple d'une taxonomie.
Source : ([Yuhua Li and Crockett, 2006])

L'**hyperonymie** est la relation sémantique hiérarchique d'une unité lexicale à une autre selon laquelle l'extension du premier terme, plus général, englobe l'extension du second, plus spécifique. Le premier terme est dit *hyperonyme* de l'autre, ou super-ordonné par rapport à l'autre. C'est le contraire de l'*hyponymie.*
D'une manière générale et simplifiée, un *hyperonyme* est une catégorie générale regroupant des sous-catégories.

La Figure (2.3) ci-dessous illustre la notion d'hyperonymie/hyponymie.

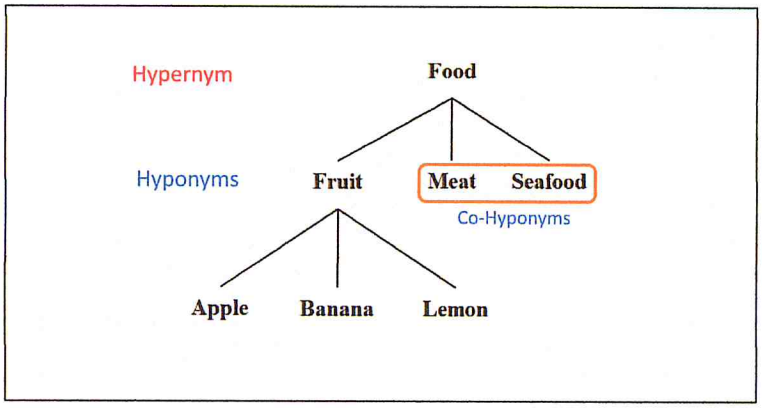


FIGURE 2.3 – Exemple d'hyperonymie/hyponymie.

Parmi les mesures de similarité sémantique utilisant WordNet, nous retrouvons les mesures basées sur la distance taxonomique. Leur principe est de compter le nombre d'arcs qui séparent deux sens dans une taxonomie. Dans ce cadre, nous choisissons la mesure définie par ([Wu and Palmer, 1994]).

• **Mesure WUP :** La mesure de similarité Wu & Palmer (wup) est définie selon la distance qui sépare deux concepts par rapport à leur sens commun le plus spécifique LCS (Least Common Subsumer) que la racine de la taxonomie.

Le score de similarité résultante est compris dans l'intervalle]0,1]. Tel qu'il ne peut jamais être nul car la profondeur du LCS n'est jamais nulle, 1 indique que les deux synsets sont identiques, et les valeurs intermédiaires indiquent le degré de similitude ou dis-similitude.

La Figure 2.4 représente la relation de deux sens quelconques S_1 et S_2 dans une taxonomie par rapport à leur sens commun le plus spécifique (LCS) S_3 et par rapport à la racine de la taxonomie.

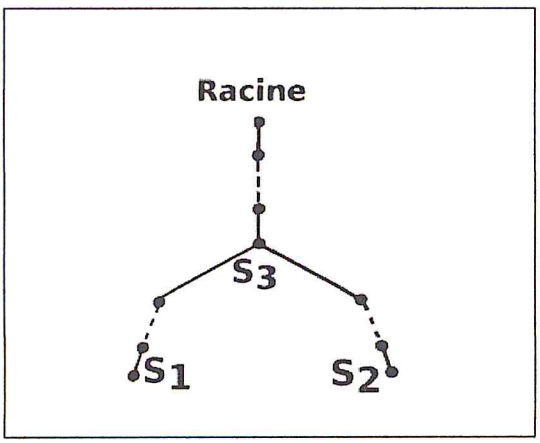


FIGURE 2.4 – Deux sens et leur sens commun le plus spécifique dans une taxonomie. Source : ([Wu and Palmer, 1994])

La formule de similarité wup entre deux sens S_1 et S_2 est définit comme suit :

$$Sim_{wup}(s_1, s_2) = \frac{2 \times depth(LCS)}{depth(s_1) + depth(s_2)} \tag{2.5}$$

Où :

– depth(LCS) : Le nombre d'arcs qui séparent LCS de la racine.

– $depth(s_1)$: Le nombre d'arcs qui séparent s_1 de la racine en passant par LCS.

– $depth(s_2)$: Le nombre d'arcs qui séparent s_2 de la racine en passant par LCS.

La Figure 2.5 illustre les différentes méthodes existantes pour le calcul de similarité.

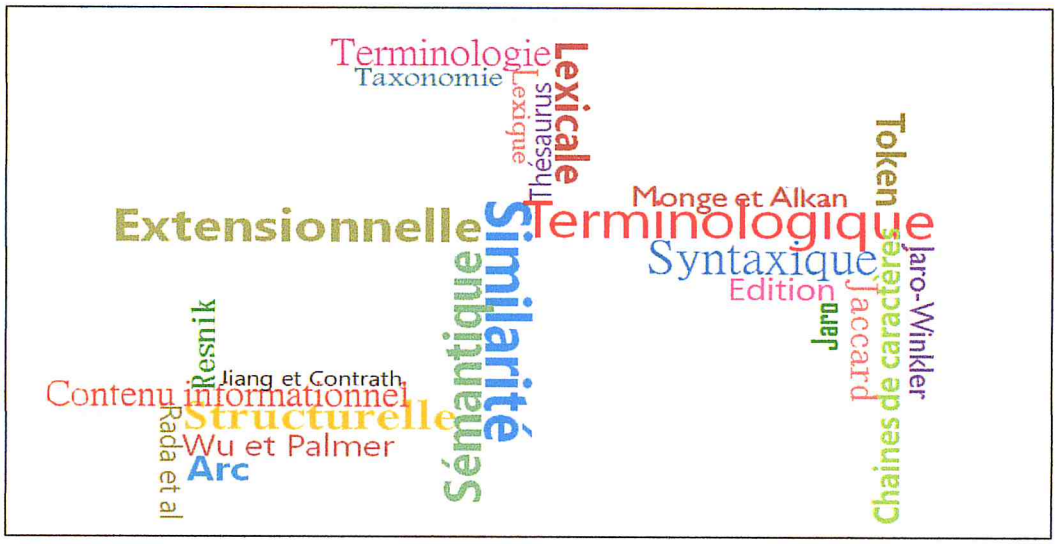


FIGURE 2.5 – Les différentes méthodes pour le calcul de similarité.
Source : ([MmeFareh, 2017])

Conclusion

Dans ce chapitre nous avons présenté les techniques de pré-traitement et les mesures de similarité que nous avons choisi d'appliquer lors de nos expérimentations.

A la fin de cette étape, les tweets convertis en des vecteurs numériques seront passer vers les algorithmes de clustering et classification que nous allons voir dans le chapitre suivant.

CHAPITRE 3

LES APPROCHES PROPOSÉES POUR LA FOUILLE DES TWEETS

Introduction

Dans ce chapitre, nous allons présenter en détails les algorithmes que nous avons choisi à utiliser sur notre flux de travail.

Ces algorithmes sont séparés en deux catégories, des algorithmes pour l'apprentissage non-supervisé et d'autres pour l'apprentissage supervisé.

3.1 Apprentissage non-supervisé

L'apprentissage non-supervisé fait partis des techniques de l'apprentissage automatique, qui consiste à apprendre sans superviseur. Il s'agit d'extraire des classes ou groupes de tweets présentant des caractéristiques communes à partir de données non étiquetées (absence d'annotation).

Comme a été mentionné dans l'état de l'art, on va s'intéresser à l'algorithme « K-means » ainsi qu'aux algorithmes du topic modeling, « LDA » et « LSI ».

➢ **Pseudo-algorithme K-means**

Entrée : k (nombre de cluster) ;

Ensemble de tweets ;

Sortie : Ensemble de k clusters ;

1. Initialisation des centres $\mu_1, \ldots, \mu_k$;
2. Répéter :
   - Affectation de chaque tweet à son cluster le plus proche ;

     $C_l \longleftarrow x_i$ tel que $l = min\ d(x_i, \mu_k)$; d est la distance Euclidienne où

     $$d(x_i, \mu_k) = \sqrt{\sum_{j=1}^{n} (x_{i_j} - \mu_{k_j})^2}$$

   - Recalculer le centre μ_k de chaque cluster ;

     $$\mu_k = \frac{1}{N_k} \sum_{i \in C_k} x_i \text{ avec } N_k = card(C_k)$$

3. Jusqu'à ce qu'il n'y aura aucun changement ;

Pour l'initialisation des centres on a procédé comme suit :

1. Sélectionner le premier centre aléatoirement.
2. Trouver les tweets les plus éloignés du(des) centre(s) (en calculant la distance entre le(s) centre(s) et les x_i).
3. Affecter le centre du nouveau cluster à proximité de ces tweets éloignés.
4. Revenir à 2 jusqu'à l'initialisation de tous les centres.

La Figure 3.1 schématise les étapes du pseudo-algorithme cité ci-dessus.

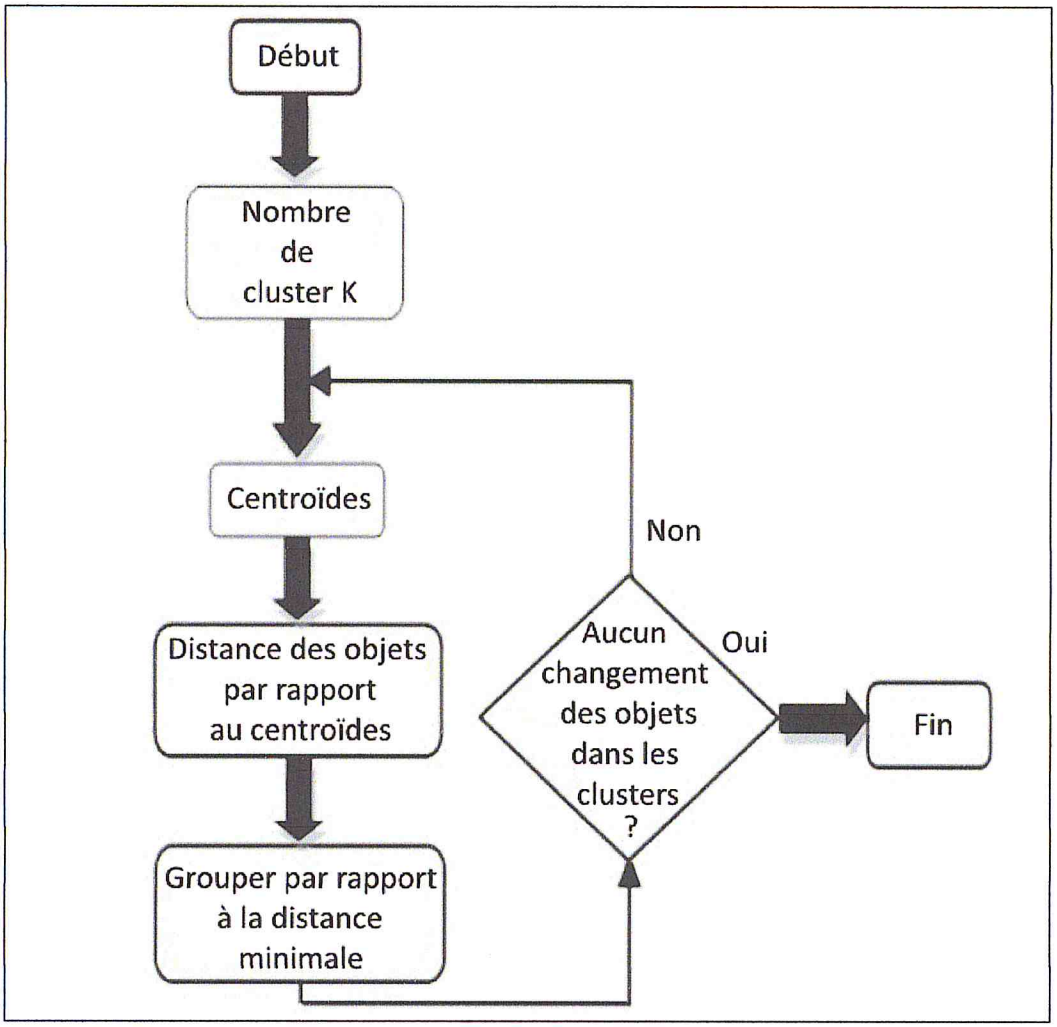


FIGURE 3.1 – Architecture générale de l'algorithme K-means.

Le point en plus que nous avons introduit est l'utilisation des centres de gravité finaux obtenu via K-means pour classifier de nouvelles données. Donc, ces centres joueront le rôle de "classe".

3.1.2 Topic modeling

Topic modeling est une technique de modélisation des documents textuels qui a été largement utilisée dans l'exploration de texte, l'analyse de réseaux et la génétique, et bien d'autres domaines. Les modèles thématiques sont importants pour démontrer des données discrètes ; aussi, donnez une approche productive pour trouver des structures/sémantiques cachées dans des informations gigantesques.

Le topic modeling peut être facilement comparée au clustering. Comme dans le cas du clustering, le nombre de topics, comme le nombre de clusters, est un hyperparamètre. En faisant du topic modeling, nous construisons des clusters (groupes) de mots plutôt que des clusters de textes. Un texte est donc un mélange de tous les topics (thèmes), chacun ayant un certain poids.

Il existe plusieurs scénarios dans lesquels le topic modeling peut s'avérer utile. En voici quelques uns :

- **Classification textuelle :** Le topic modeling peut améliorer la classification en regroupant des mots similaires dans des sujets plutôt qu'en utilisant chaque mot comme caractéristique (feature).
- **Systèmes de recommandation :** En utilisant une mesure de similarité, nous pouvons créer des systèmes de recommandation. Le système recommandera des articles avec une structure de thème (topic) similaire aux articles que l'utilisateur a déjà lus.
- **Découverte des thématiques dans les textes :** Utile pour détecter les tendances dans les publications en ligne par exemple.

FIGURE 3.2 – Une vision sur l'application du topic modeling dans diverses sciences.

Dans ce qui suit, on va s'intéresser aux deux algorithmes **LDA** (Latent Dirichlet Allocation) et **LSI** (Latent Semantic Indexing) connue aussi sous le nom d'**LSA** (Latent Semantic Analysis).

3.1.2.1 Hybridation LDA_LSI

3.1.2.1.1 LDA (Latent Dirichlet Allocation)

L'**allocation de Dirichlet Latent** (ou **LDA**, de l'anglais : **Latent Dirichlet Allocation**) est une approche générative dans la classification des textes. Il s'agit d'un modèle Bayésien (basé sur les statistiques bayésiennes) hiérarchique à trois niveaux où il crée des probabilités au niveau du mot, au niveau du document et au niveau du corpus [1].

D'une manière simple, l'idée de base du processus est, chaque document est modélisé comme un mélange de sujets (topics), et chaque sujet est une distribution de probabilité discrète qui définit la probabilité d'appartenance d'un mot dans un sujet donné. Ces probabilités de sujet fournissent une représentation concise d'un document.

Supposons qu'on définit le nombre de topics (sujets/thèmes) à 4 et on a un tweet A, ce dernier est caractérisé par un ou plusieurs sujets. Par exemple, 60% du 1er topic, 30% du 2ème topic, 10% du 3ème topic et n'appartient pas au 4ème topic.

Pour M documents, on a K topics et Nm mots par document. La fonction objectif semble assez complexe, mais c'est juste une façon de décrire la probabilité d'un corpus :

$$p(D|\alpha,\eta) = \prod m = 1M \int p(\theta m|\alpha) = \prod n = 1Nm \sum_{i=1}^{kp} (z_{i_{mn}}|\theta m) p(w_{mn}|z_{i_{mn}},\eta) d\theta m \qquad (3.2)$$

Où :

– nm : Représente le mot n dans un document m.

1. Ensemble de tous les documents. Exemple : articles, tweets, etc.

Le modèle bayésien hiérarchique à trois niveaux peut maintenant être résumé comme suit :

– α et η : Des paramètres au niveau du corpus. Tel que, alpha (α) plus elle est grande, plus le document sera assigner à plusieurs topics et vise-vers-sa. Eta (η), quant à elle plus elle est élevée, plus le topic contiendra beaucoup de mots.

– $\theta_1 \dots \theta_m$: Des variables au niveau du document, la distribution d'un topic pour un document m.

– $z_{i_{mn}}$ et w_{mn} : Des variables au niveau du mot. Tel que $z_{i_{mn}}$ est le sujet du n-ème mot dans le document m et w_{mn} est le mot spécifié.

➢ **Pseudo-algorithme LDA**

Entrée : K, le nombre de thèmes (topics) ;

Objectif : Apprendre les thèmes représentés dans chaque tweet et les mots associés à ces thèmes ;

1. Initialisation (génération du premier "topic model") :

   Attribuer un thème à chaque mot de chaque tweet, selon une distribution de Dirichlet sur un ensemble de K thèmes :

   $\theta_i \sim Dir(\alpha)$, avec $i \in \{1, \dots, M\}$ et $Dir(\alpha)$ est une distribution de Dirichlet avec un paramètre symétrique α creux ($\alpha < 1$).

2. Répéter :

   Pour chaque mot w de chaque tweet d, on calcule deux choses pour chaque thème t :

   - $p(t/d)$: La probabilité que le tweet d soit assigné au thème t ;
   - $p(w/t)$: La probabilité que le thème t dans le corpus soit assigné au mot w ;

   On choisit alors le nouveau thème t avec la probabilité $p(t/d) \times p(w/t)$.

   Ceci correspond à la probabilité que le thème t génère le mot w dans le tweet d.

3. Jusqu'à la stabilisation des assignations ;

En utilisant la *notation plate*,la Figure 3.3 schématise le processus d'LDA.

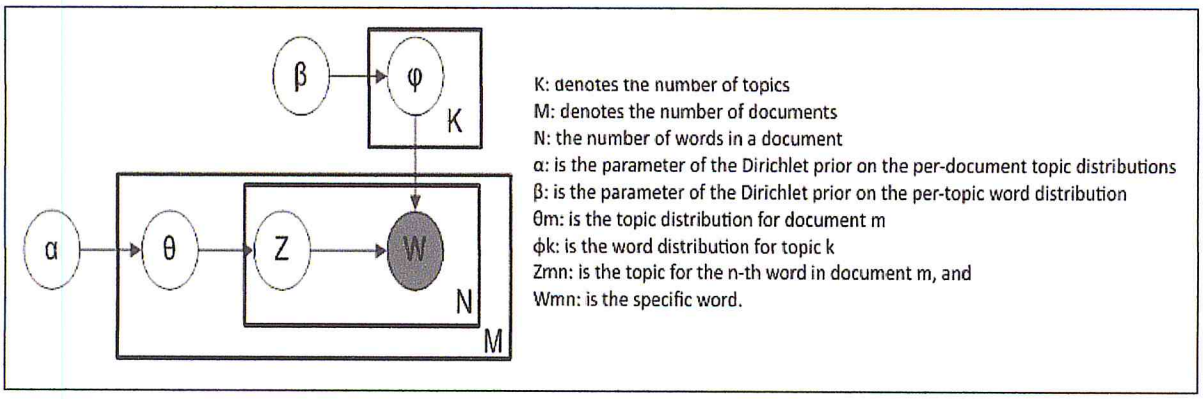


FIGURE 3.3 – Modèle graphique d'LDA.
Source : ([Blei et al., 2003]).

3.1.2.1.2 LSI (Latent Semantic Indexing)

Indexation sémantique latente (ou **LSI**, de l'anglais : Latent semantic indexation) ou l'**analyse sémantique latente** (**LSA**, de l'anglais : Latent semantic analysis) est une technique de traitement du langage naturel, en particulier la distribution sémantique, qui consiste à analyser les relations entre un ensemble de documents et les termes qu'ils contiennent en produisant un ensemble de concepts liés aux documents et aux termes.

LSI suppose que les mots qui ont une signification proche se produiront dans des parties de texte similaires (l'hypothèse distributionnelle). Elle utilise une matrice qui décrit l'occurrence de certains termes dans les documents. C'est une matrice creuse dont les lignes correspondent aux « termes » et dont les colonnes correspondent aux « documents ».

Une technique mathématique appelée **décomposition en valeurs singulières** (ou **SVD**, de l'anglais : singular value decomposition) est utilisée pour réduire le nombre de lignes tout en préservant la structure de similarité entre les colonnes.

Les mots sont ensuite comparés en prenant le cosinus de l'angle entre les deux vecteurs (ou le produit scalaire entre les normalisations des deux vecteurs) formés par deux rangées quelconques. Les valeurs proches de "1" représentent des mots très similaires alors que les valeurs proches de "0" représentent des mots très dissemblables.

Donc, LSI repose entièrement sur SVD pour identifier les patterns dans les relations entre les termes et les concepts contenus dans une collection de texte non structurée.

Pour une matrice de terme de document X, se décompose en U et V, qui sont des matrices orthogonales et S est une matrice diagonale. La formule de la décomposition de valeur singulière (SVD) est comme suit :

$$X = USV^T \tag{3.3}$$

Où :

– V^T : La transposée de la matrice V.

➢ Pseudo-algorithme LSI

Entrée : K, le nombre de concepts (topics) ;

1. Construction de la matrice des fréquences :

   X la matrice où l'élément *(i,j)* décrit les fréquences du terme i dans le tweet j.

2. Corrélation (produit scalaire) :

   Le produit matricielle XX^T contient tous les produits scalaires entre les vecteurs « termes », qui donnent la corrélation entre deux termes sur l'ensemble du corpus.

   De même, le produit X^TX contient tous les produits scalaires entre les vecteurs « tweets », qui donnent leurs corrélations sur l'ensemble du lexique.

3. Décomposition en valeurs singulières (SVD) :

   Effectuer une décomposition en valeurs singulières sur X, qui donne deux matrices orthonormales U et V et une matrice diagonale S : $X = USV^T$

4. Espace des concepts :

   Sélectionner les K plus grandes valeurs singulières, ainsi que les vecteurs singuliers correspondants dans U et V.

   Les vecteurs « termes » possèdent alors K composantes, qui chacune donne l'importance du terme i dans chacun des K différents « concepts » (topics).
   De même, les vecteurs « tweets » donnent l'intensité des relations entre le tweet j et chaque concept.

   On écrit cette approximation sous la forme suivante : $X_k = U_k S_k V_k^T$

La Figure 3.4 schématise le processus d'SVD utilisé par la technique LSI.

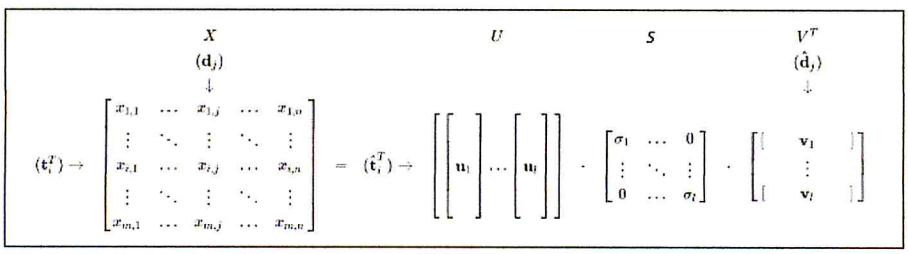
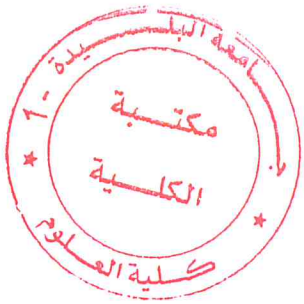


FIGURE 3.4 – Décomposition SVD.

3.1.2.1.3 Hybridation LDA_LSI

L'hybridation entre LDA et LSI consiste à créer un nouveau modèle qui se base sur ces deux approches pour la classification des tweets.

➢ **Procédure**



Entrée : Les modèles LDA et LSI entrainés ;

1. Extraction des caractéristiques avec LDA ;
2. Extraction des caractéristiques avec LSI ;
3. Concaténation des deux matrices "caractéristiques" ;
4. Utilisation d'une grille de recherche à cadre de validation croisé (GridSearch Cross Validation) sur la matrice résultante ;
5. Sauvegarder le modèle avec une meilleure régression logistique ;

Où :

– **Extraction des caractéristiques** : Caractéristiques (ou features, en anglais), c-à-d, extraire la proportion qu'un topic x soit présent dans un tweet.

– **Grille de recherche à cadre de validation croisé** : En anglais, GridSearch Cross Validation est une méthode de recherche exhaustive sur des valeurs de paramètres spécifiées pour un estimateur donnée. Les paramètres de l'estimateur utilisé sont optimisés par une grille de validation croisée sur une grille de paramètres.

L'estimateur choisi est la *régression logistique*, qui a pour but de modéliser au mieux un modèle mathématique simple à des observations réelles nombreuses. En d'autres termes d'associer à un vecteur de variables aléatoires $(x_1, \ldots, x_k)$ à une variable aléatoire binomiale (ou multinomiale) génériquement notée y.

Les paramètres spécifiées pour l'estimateur sont les deux normes *L1* et *L2* pour le calcul de distance. Tel que :

On note $\vec{x}$ *un vecteur* $(x_1, \ldots, x_n)$

- *La norme 1*, dite aussi *distance de Manhattan* est donnée par la somme des valeurs absolues des coefficients :

$$L1 = ||\vec{x}||_1 = |x_1| + \cdots + |x_n| \tag{3.4}$$

- Plus généralement, pour tout p supérieur ou égal à 1, *la norme p* est donnée par la formule suivante :

$$Lp = ||\vec{x}||_p = (|x_1^p| + \cdots + |x_n^p|)^{\frac{1}{p}} \tag{3.5}$$

Elle identifie donc la norme euclidienne avec la norme 2 ($p = 2$).

La validation croisée est une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage. Il existe plusieurs techniques de validation croisée, la technique choisie est « k-fold cross-validation » qui consiste à :

1. Diviser l'échantillon original en k échantillons.
2. Sélectionner un des k échantillons comme ensemble de validation et les $k - 1$ autres échantillons constitueront l'ensemble d'apprentissage.
3. Calculer le score de performance avec la méthode *Logistic Loss* selon la formule suivante :

$$V(f(\vec{x}), y) = \frac{1}{\ln 2} \ln(1 + e^{-yf(\vec{x})}) \tag{3.6}$$

4. Répète l'opération (3) en sélectionnant un autre échantillon de validation parmi les $k-1$ échantillons qui n'ont pas encore été utilisés pour la validation du modèle.

   (Note : l'opération se répète ainsi k fois pour que chaque sous-échantillon soit utiliser exactement une fois comme ensemble de validation.)

5. Calculer La moyenne des k scores pour estimer l'erreur de prédiction.

6. Le modèle avec une erreur minime sera sauvegarder pour ensuite être utiliser lors de la phase de classification.

La Figure 3.5 met en évidence la méthode de validation croisée pour un modèle. Chaque point appartient à 1 des 5 ensemble de validation (en blanc) et aux 4 autres l'ensemble d'apprentissage (en orange).

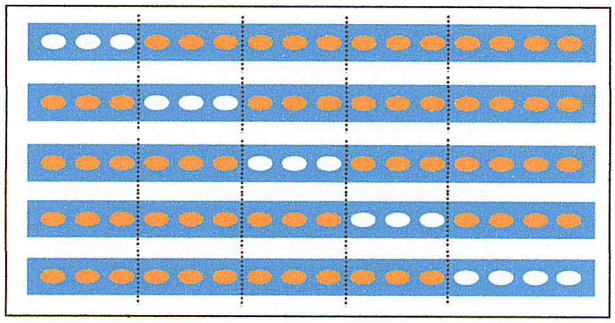


FIGURE 3.5 – Une validation croisée à 5 folds.
Source : ([Azencott, 2018])

3.2 Apprentissage supervisé

Apprentissage supervisé ou une classification supervisée est une technique d'apprentissage automatique qui consiste à regrouper divers objets (les individus) en sous-ensembles d'objets (les classes) à partir de données étiquetées.

Un scénario optimal pour un algorithme d'apprentissage supervisé est de pouvoir déterminer correctement les classes pour les instances inconnus (nouvelles données).

Parmi les algorithmes de classification supervisée :

- Arbre de décision.
- Méthode des K plus proches voisins (Kpp ou Knn en anglais).
- Réseau de neurones artificiel.
- Machine à vecteurs de support (SVM).

Dans les sections suivantes on s'intéresse aux réseaux de neurones artificiels.

3.2.1 Réseau de neurones artificiel

Un **réseau neuronal artificiel** (ou **ANN**, de l'anglais : Artificial neural network) est un modèle dont la conception est basée sur la structure et fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

Les réseaux de neurones sont capables d'apprendre à effectuer des tâches comme la classification, la prédiction, la prise de décision, la visualisation et d'autres, simplement en considérant des exemples.

3.2.1.1 Perceptron

Le perceptron peut être vu comme le type de réseau de neurones le plus simple. C'est un classifieur linéaire. Ce type de réseau neuronal ne contient aucun cycle et est doté d'une fonction de transfert qui transforme ses entrées en sortie selon des règles précises. Par exemple, un neurone somme ses entrées, compare la somme résultante à une valeur seuil, et répond en émettant un signal si cette somme est supérieure ou égale à ce seuil.

➢ **Modélisation et principe**

- Le neurone reçoit les entrées $x_1, \ldots, x_n$.
- Le potentiel d'activation du neurone p est défini comme la somme pondérée (les poids sont les coefficients synaptiques w_i de type réel) des entrées.

$$p = x.w = x_1.w_1 + \cdots + x_n.w_n \tag{3.7}$$

- La sortie est calculée en fonction du seuil θ.

$$1 \; si \; p > \theta \; ; \; 0 \; si \; p \leq \theta \tag{3.8}$$

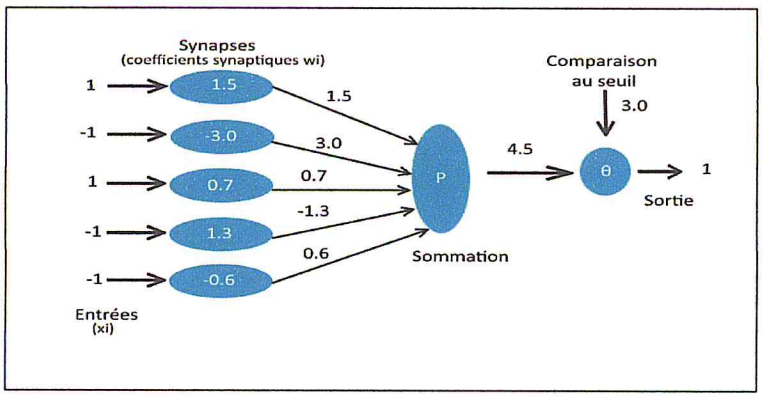


FIGURE 3.6 – Exemple schématisant le principe d'un réseau de neurones.

3.2.1.2 Perceptron multicouches

Le **perceptron multicouche** (ou **MLP**, de l'anglais : Multilayer perceptron) est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente.

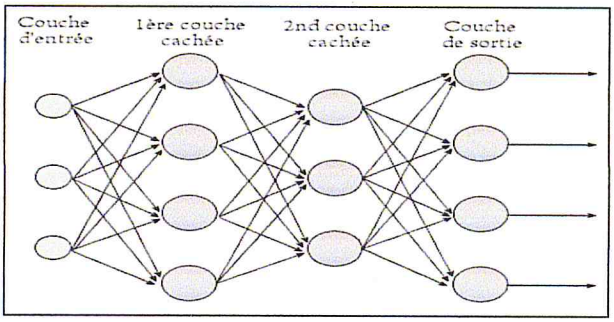


FIGURE 3.7 – Perceptron multicouche.
Source : ([HRcommons, 2009])

– Chaque couche (i) est composée de N_i neurones, prenant leurs entrées sur les N_{i-1} neurones de la couche précédente.

– À chaque synapse est associé un poids synaptique, de sorte que les N_{i-1} sont multipliés par ce poids, puis additionnés par les neurones de niveau i, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation.

La Figure 3.8 représente la structure général d'un neurone artificiel. Tel que, le neurone calcule la somme de ses entrées puis cette valeur passe à travers la fonction d'activation pour produire sa sortie.

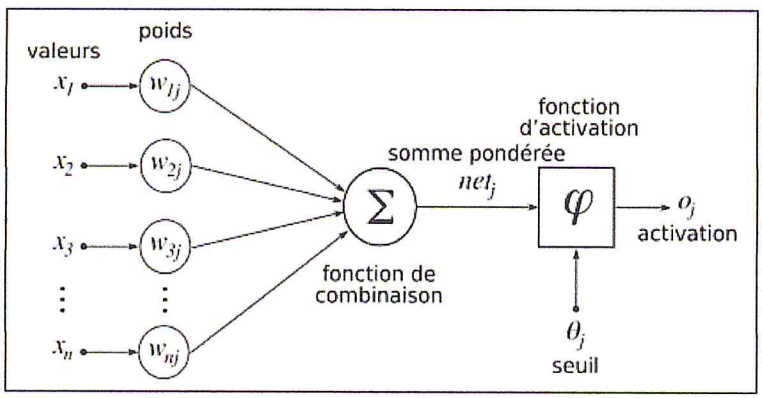


FIGURE 3.8 – Structure d'un neurone artificiel.
Source : ([Chrislb, 2005])

➢ **Fonction de combinaison**

La **fonction de combinaison** est une fonction *vecteur à scalaire* utilisé par le neurone lorsqu'il reçoit un certain nombre de valeurs via ses connexions synaptiques avec les neurones en amont, alors il renvoie le produit scalaire entre le vecteur des entrées et le vecteur des poids synaptique.

➢ **Fonction d'activation**

La **fonction d'activation** (ou **fonction de seuillage**, ou encore **fonction de transfert**) sert à introduire une non-linéarité dans le fonctionnement du neurone. Cela lui permet de généraliser ou de s'adapter à une variété de données et de différencier les résultats. La fonction de seuillage est donc utilisée pour déterminer la sortie du réseau de neurones comme "oui" ou "non". Tout dépend la fonction choisie, les valeurs résultantes sont compris dans l'intervalle [0,1] ou [-1,1], ou encore $[0,\infty[$
Tel que :

- La valeur 0 ou -1 signifie que le neurone est inactif (en dessous du seuil).
- La valeur 1 signifie que le neurone est actif (au dessus du seuil).

Les fonctions d'activation non linéaires sont principalement divisées en fonction de leur intervalles ou de leurs courbes. Nous nous intéressons aux fonctions suivantes : Sigmoïd, Softmax et ReLu.

1. **Sigmoïd ou fonction d'activation logistique :**
   La fonction sigmoïde (dite aussi courbe en S) est définie par :

$$f(x) = \frac{1}{1+e^{-x}} \quad \textit{pour tout réel } x \tag{3.9}$$

   La raison principale pour laquelle la fonction sigmoïde est utilisée c'est parce qu'elle existe entre [0,1]. Par conséquent, elle est particulièrement utilisé pour les modèles où nous devons prédire la probabilité en tant que sortie.

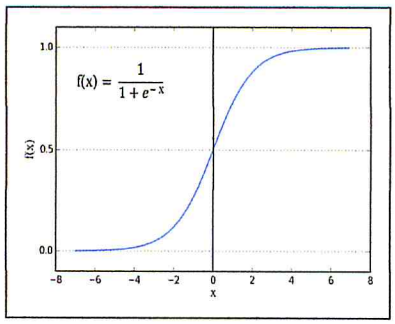


FIGURE 3.9 – Fonction Sigmoïd.

2. **Softmax ou fonction exponentielle normalisée :**

La fonction softmax est une généralisation de la fonction logistique et est définie par :

$$f(x)_j = \frac{e^{x_j}}{\sum_{i=1}^{K} e^{x_j}} \quad pour\ tout\ j \in \{1,\ldots,k\},\ x = (x_1,\ldots,x_k)\ de\ k\ nombres\ réels. \tag{3.10}$$

Le vecteur $f(x)$ en sortie est de K nombres réels strictement positifs et de somme *1*.

3. **ReLu ou fonction d'unité de rectification linéaire :**

La fonction d'unité de rectification linéaire (ReLu) est la fonction d'activation non linéaire la plus simple. Sa sortie est compris dans l'intervalle [0,∞[, elle vaut *0* si l'entrée est inférieure à *0* et une sortie brute dans le cas contraire. C'est-à-dire que si l'entrée est supérieure à *0*, la sortie est égale à l'entrée. Sa formule est :

$$f(x) = \begin{cases} 0 & pour \quad x < 0 \\ x & pour \quad x \geq 0 \end{cases} \tag{3.11}$$

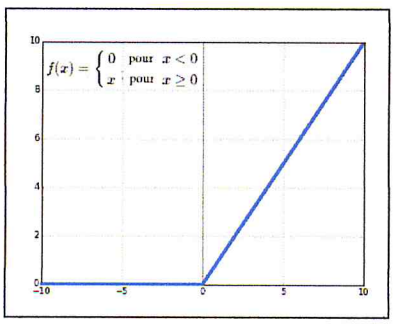


FIGURE 3.10 – Fonction ReLu.

➢ Fonctions de perte

Les fonctions de perte (ou de l'anglais : Loss functions) sont utiles pour entrainer un réseau de neurones. Avec une entrée et une cible, elles calculent la perte, c'est-à-dire la différence entre la sortie et la variable cible. Différentes fonctions de perte donneront différentes erreurs pour la même prédiction, et auront donc un effet considérable sur la performance du modèle. Ces fonctions se séparent en trois catégories :

- Fonctions dans le cas d'une régression.
- Fonctions dans le cas d'une perte d'intégration.
- Fonctions dans le cas d'une classification.

On s'intéresse à la troisième catégorie qui regroupe les fonctions de perte dans le cas d'une classification.

Fonctions de perte pour un problème de classification :

La variable de sortie dans le problème de classification est souvent une valeur de probabilité $f(x)$, appelée score pour l'entrée x. Généralement, l'ampleur du score représente la confiance de notre prédiction dont la variable cible y est une variable binaire 1 pour "vrai" et -1 pour "faux". Cette catégorie regroupe plusieurs méthodes tel que :

- Cross Entropy.
- Negative Log Likelihood.
- Margin Classifier.
- Soft Margin Classifier.

On s'intéresse à la méthode "cross entropy".

Cross Entropy : La perte d'entropie croisée, ou perte de log, mesure la performance d'un modèle de classification dont la sortie est une valeur de probabilité entre 0 et 1. La perte d'entropie croisée augmente lorsque la probabilité prédite diverge de l'étiquette réelle.

Sa formule est définie comme suit :

$$-\sum_{c=1}^{k} y_{o,c} \log(p_{o,c}) \tag{3.12}$$

Où :

- k : Le nombre de classes.
- y : Indicateur binaire (0 ou 1) si l'étiquette de classe c est la classification correcte pour l'observation o.
- p : La probabilité prédit pour l'observation o si est de classe c.

➢ **Fonctions d'optimisation**

Les fonctions d'optimisation calculent habituellement le gradient, c'est-à-dire la dérivée partielle de la fonction de perte par rapport aux poids, et les poids sont modifiés dans la direction opposée du gradient calculé. Ce cycle est répété jusqu'à ce qu'on atteint le minimum de la fonction de perte. On s'intéresse aux deux fonctions suivantes : RMSProp et Adam.

1. **RMSProp :**

   Root Mean Square Prop (RMSProp) fonctionne en conservant une moyenne pondérée exponentiellement des carrés des gradients passé. RMSProp divise ensuite le taux d'apprentissage par cette moyenne pour accélérer la convergence.

2. **Adam :**

   La fonction Adam calcule le taux d'apprentissage adaptatif pour chaque paramètre et fonctionne comme suit :

   - Premièrement, elle calcule la moyenne exponentiellement pondérée des gradients passés.
   - Deuxièmement, calcule la moyenne exponentiellement pondérée des carrés des gradients passés.
   - Troisièmement, ces moyennes ont un biais vers zéro et pour contrer cela, une petite valeur ϵ est ajoutée.
   - Enfin, les paramètres sont mis à jour en utilisant les informations des moyennes calculées.

➢ **Apprentissage par descente de gradient**

- Soit le vecteur des entrées x et le vecteur des coefficients synaptiques w.
- La sortie vaut alors : $o = f(x.w) = f(x_0.w_0 + \cdots + x_{n.wn})$, f étant une fonction de transfert (activation) continue et dérivable.
- Posons $y = x.w$.
- Soit S la base d'apprentissage composée de couples (x, c), où c est la sortie attendue pour x.
- On définit l'erreur sur le réseau pour la base S : $E(w) = fonction\ de\ perte$.
- But : Trouver w qui minimise $E(w)$.

Pseudo-algorithme

1. Initialiser aléatoirement les coefficients w_i ;
2. Répéter :
   - Pour tout i : $\Delta w_i = 0$;
   - Pour tout exemple (x, c) *dans* S :
     - Calculer la sortie o du réseau pour l'entrée x ;
     - Pour tout i : $\Delta w_i = \Delta w_i + \epsilon \times (c - o) \times x_i \times f'(x.w)$;
   - Pour tout i : $w_i = w_i + \Delta w_i$;
3. Jusqu'à convergence ;

Ainsi, les composants d'un modèle de réseau neuronal, c'est-à-dire la fonction d'activation, la fonction de perte et ceux d'optimisation jouent un rôle très important lors de l'entrainement pour obtenir un modèle efficace qui produit des résultats précis. Différentes tâches nécessitent un ensemble différent de telles fonctions pour donner les résultats les plus optimaux.

Pour les réseaux de neurones, notre contribution à pour but de prouver qu'on peut atteindre un résultat acceptable en utilisant le modèle MLP mais seulement avec une seule couche cachée (Figure 3.11) et jouer sur le nombre de neurones et les différents paramètres.

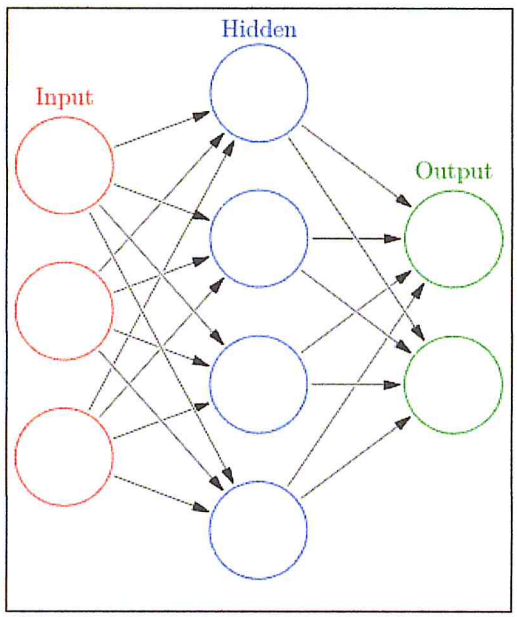


FIGURE 3.11 – Schéma d'un réseau de neurones artificiel avec une seule couche caché. Source : (Glosser.ca Cburnett, neural network)

Conclusion

Après avoir présenter et expliquer le principe de chaque approche proposée pour la fouille des tweets, dans le chapitre suivant nous allons exposer les résultats de nos tests obtenus lors de la segmentation(clustering) et classification des tweets.

CHAPITRE 4

EXPÉRIMENTATIONS ET COMPARAISON DES RÉSULTATS

Introduction

Dans ce chapitre nous allons exposer les résultats de différents tests qui ont été effectué sur de multiples dataset [1].

La Figure 4.1 schématise et résume les différentes phases de notre contribution.

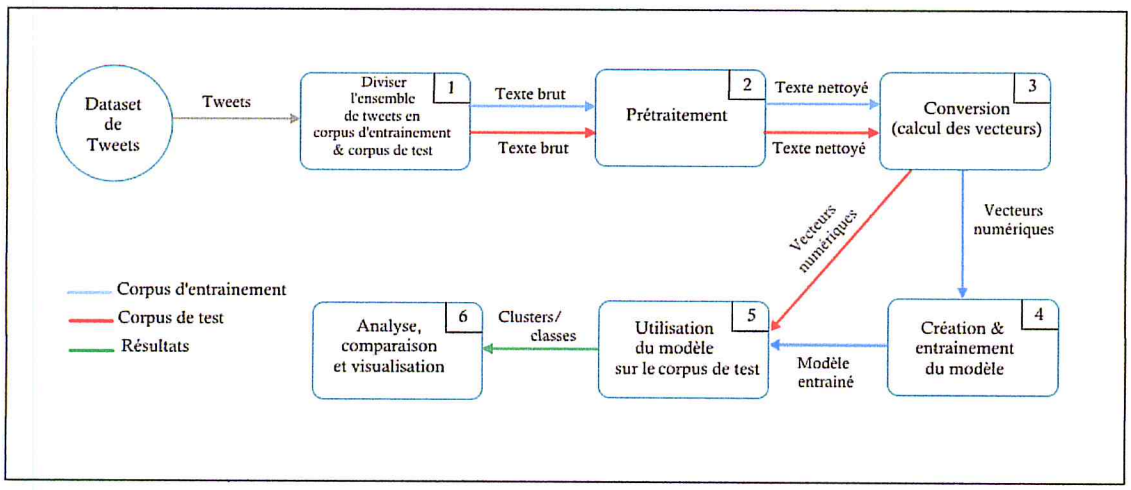


FIGURE 4.1 – Architecture expérimentale.

1. Dataset : Un jeu de données, corpus ou encore un ensemble de données.

➢ **Dataset**

Les datasets utilisé pour les expérimentations ont été téléchargé depuis [Crowd:Flower, 2018], qui est un site officiel de partage de datasets publiques issus de Twitter ou autres source.

Les dataset utilisé sont : Airline, Coachella, Emotions_v1, Emotions_v2 et Weather.

➢ **Phase 1**

La première phase consiste à diviser de manière aléatoire l'ensemble de tweets en deux parties. 90%/80% pour l'entrainement et 10%/20% pour les tests.

➢ **Phase 2, 3, 4 et 5**

Pour la deuxième et troisième phases elles ont été abordé précédemment dans le chapitre 3. La quatrième phase quant à elle a été expliqué dans le chapitre 4, on précise que les modèles issus de cette phase on été créé et entrainé sur l'ensemble d'entrainement en appliquant les différents algorithmes d'apprentissage automatique.

La cinquième phase a pour but d'utiliser les modèles entrainés pour une meilleure segmentation/classification de l'ensemble de données de test.

➢ **Phase 6**

Dans cette phase, qui est l'objectif de ce chapitre, on va analyser, visualiser et comparer en temps d'exécution et taux de réussite les résultats issus de la 5ème phase. La mesure calculant le taux de réussite est la suivante :

• **Mesure d'évaluation de la qualité de segmentation/classification**

Pour calculer le taux de réussite de la segmentation (clustering)/classification du jeu de test on a utilisé la formule suivante :

$$Taux\ de\ réussite = \sum_{i=1}^{k} \frac{Cc_i}{Tc_i} \tag{4.1}$$

Où :

- Cc_i : Le nombre de tweet classifié correctement du cluster i. Tel que,
  $Cc_i = MAX(c_i \cap Classej) \quad \forall j \in \{1, \ldots, k\}, \quad k = nombre\ de\ classe.$
- Tc_i : Le nombre total de tweets du cluster i.
- c_i : Les tweets classifié dans le cluster i (en sortie).

➢ **Langage et modules utilisés**

Le langage de programmation que nous avons utilisé est Python, nous avons notamment utilisé les deux librairies Scikit-learn et TensorFlow.

Pour le traçage et visualisation des données en 2D nous avons utilisé la librairie "*Matplotlib*" ainsi que le module "*pyLDAvis*" pour la visualisation interactive des topics.

4.1 Notes (pilote)

Avant d'entamer la suite du chapitre nous allons expliquer les notations utilisées.
Pour certaine méthode on a utilisé la notation "standard" pour référencer les paramètres qu'on a fixé, sinon, on mentionne la valeur du paramètre qui a changé.

1. **LDA (paramètres standard) :**
   - Num_topics : Nombre de topics (K).
   - Chunksize : 2000. Taille du bloc.
   - Passes : 1. Nombre de passes.
   - Itération : 50.

   Exemple illustrant la notion de chunksize et passes. Si le corpus d'entrainement à 50 000 tweet, on met chunksize = 10 000 et passes = 2, donc l'apprentissage est effectuée en 10 mises à jour :

   # 1 tweets 0-9.999 # 2 tweets 10.000-19 999
   # 3 tweets 20.000-29.999 # 4 tweets 30.000-39.999
   # 5 tweets 40.000-49.999 (Fin passe 1). # 6 tweets 0-9.999
   # 7 tweets 10.000-19.999 # 8 tweets 20.000-29.999
   # 9 tweets 30.000-39.999 # 10 tweets 40.000-49.999 (Fin passe 2).

2. **LSI (paramètres standard) :**
   - Num_topics : Nombre de topics (K).
   - Chunksize : 20 000. Taille du bloc.
   - Passes : 1.
   - Itération : 2.

3. **K-means (paramètres standard) :**
   - Num_clusters : Nombre de clusters (K).
   - Max_iter : 300.
   - Random_state : None. Random_state est la valeur entière "seed" utilisée pour la génération aléatoire d'un nombre.

   Remarque : On va utiliser la notion "*entrainement*" pour k-means pour exprimer le fait suivant : Appliquer k-means sur l'ensemble d'entrainement pour avoir en sortis K centres de gravité (K clusters), sauvegarder le modèle avec les centres de gravité finaux.
   En ce qui concerne l'ensemble de test il va être classifier via ce modèle dont ces centres joueront le rôle de "classe", autrement dit on recalculera pas les centres de gravité.

4. **Réseau de neurones (paramètres standard) :**
   - Activation pour la couche d'entrée (Input layer) : ReLu.
   - Nombre de neurones de la couche de sortie (Output layer) : Nombre de classes (K).
   - Batch_size : Le nombre d'échantillons qui vont être propagés à travers le réseau (même chose que chunksize).
   - Epoch : Nombre d'itération (même chose que passes).
   - Validation_split : 0.1. Diviser l'ensemble d'entrainement en 90% entrainement et 10% pour valider l'entrainement.

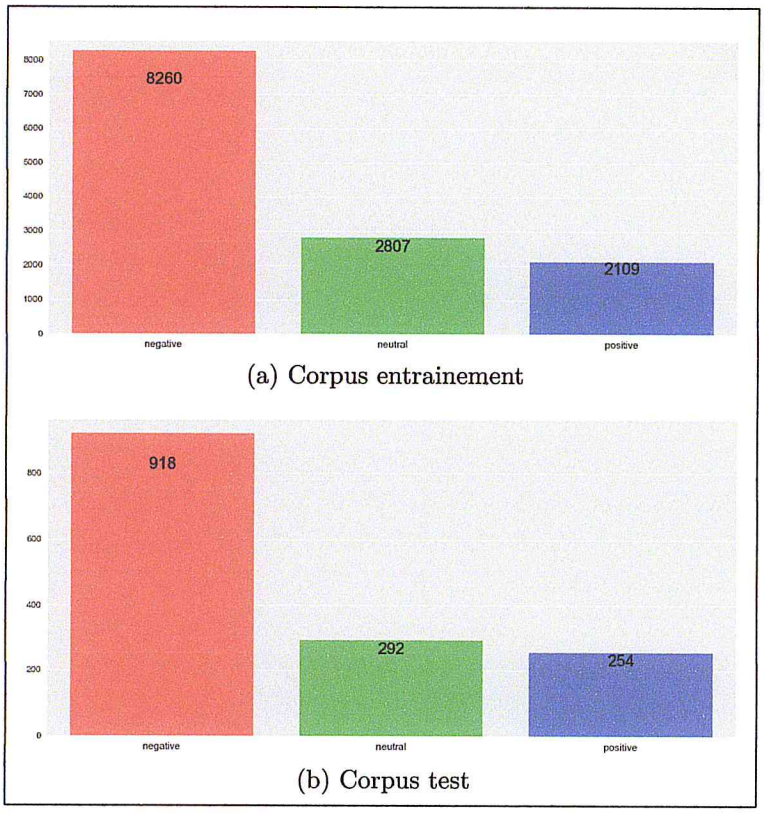
5. **TF-IDF (paramètres standard) :**
   - Norm : L2. Formule de normalisation.
   - Max_df : 0.9. La proportion des termes a ignorer qui ont une fréquence > max_df.

4.2 Dataset Airline

L'ensemble de données "*Airline Twitter sentiment*" contient des tweets sur les problèmes rencontrés dans les compagnies aériennes américaine.

- Un total de 14 640 tweets.
- Un total de 3 classes, positive, negative et neutral. Donc, pour chaque algorithme la variable K prendra la valeur 3.

Séparation de l'ensemble de données en : 13 176 tweets pour jeu de données d'entrainement et 1 464 tweets pour jeu de données de test.



(a) Corpus entrainement

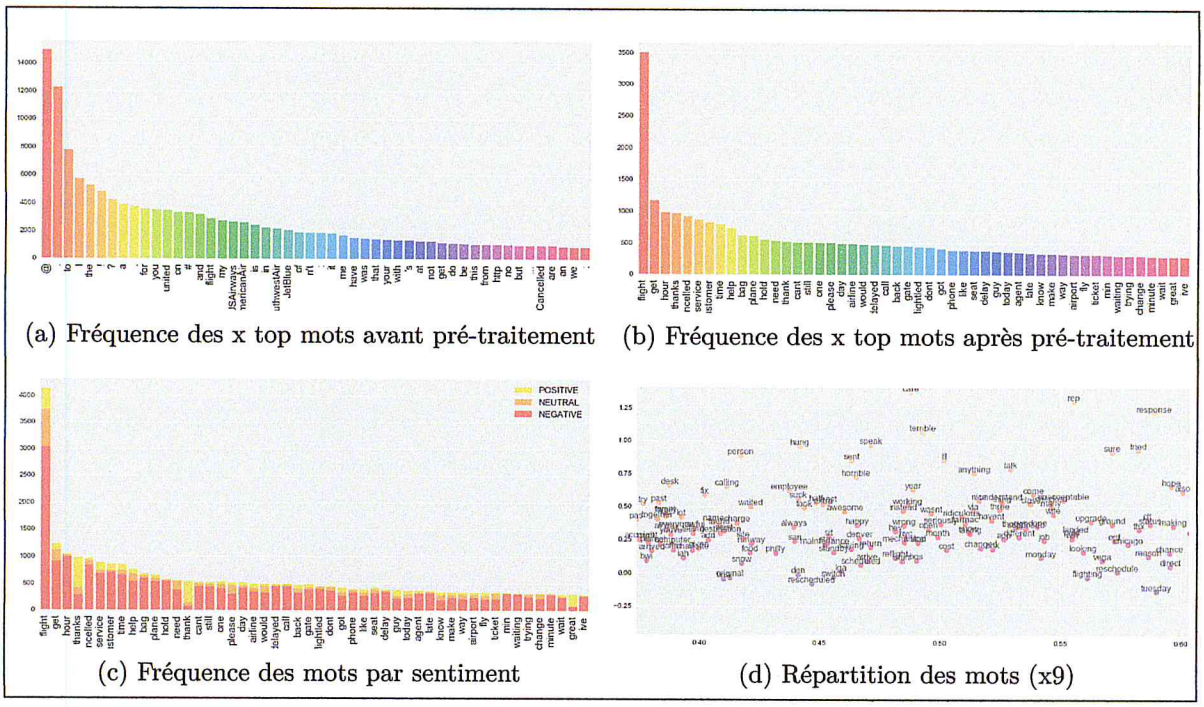
(b) Corpus test

FIGURE 4.2 – Séparation du dataset en jeu de données d'entrainement et de test.

4.2.1 Exposition des résultats

➢ **Pré-traitement**

1. **Jeu de données d'entrainement**

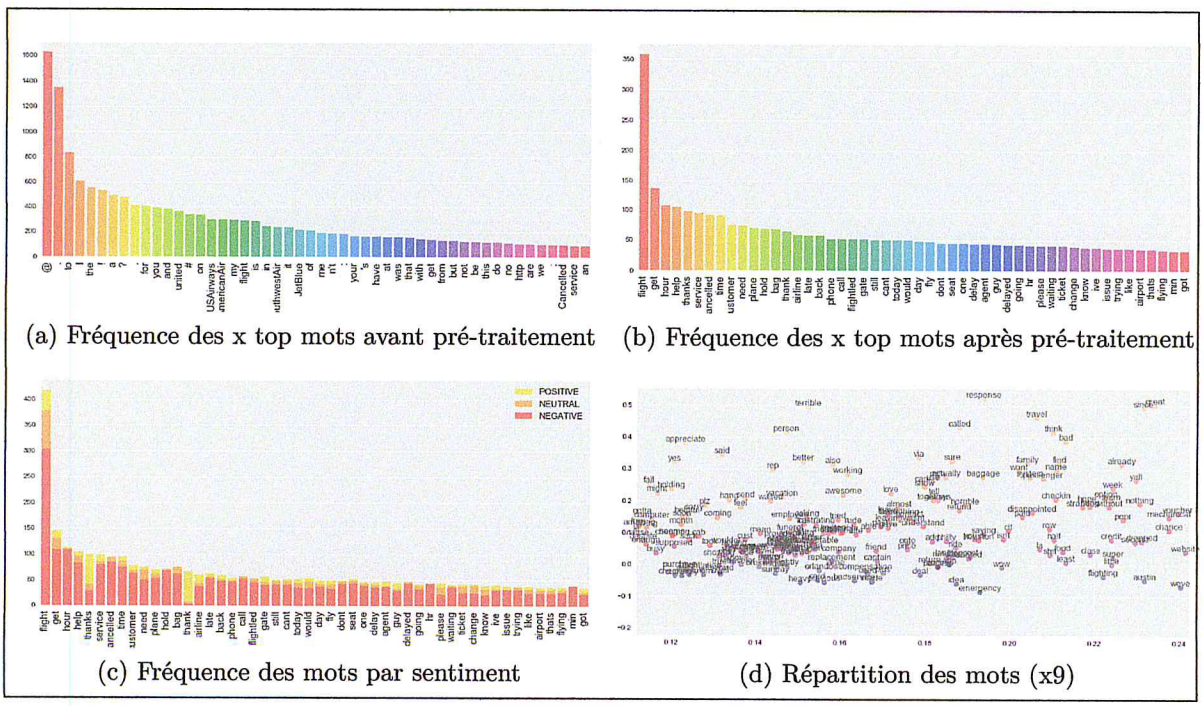


(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.3 – Airline Twitter sentiment. -corpus d'entrainement-

2. **Jeu de données de test**



(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

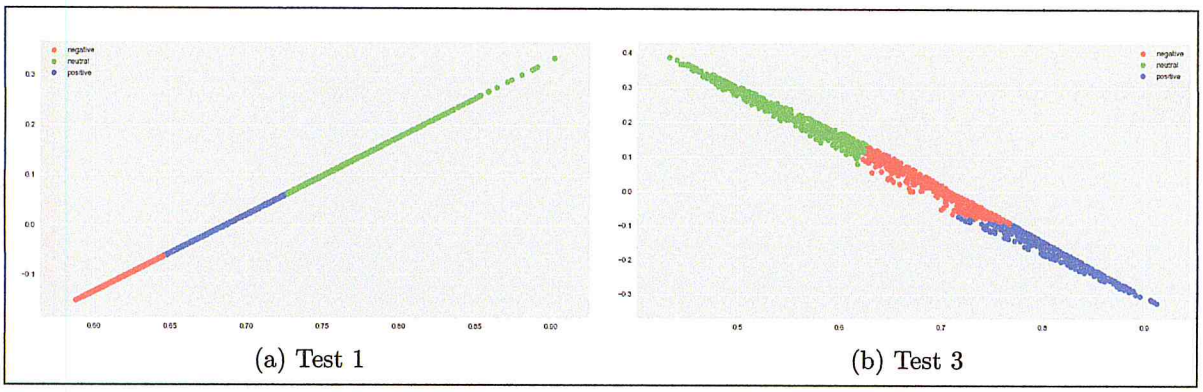
FIGURE 4.4 – Airline Twitter sentiment. -corpus de test-

➢ Clustering/Classification

1. Hybridation LDA_LSI

• Résultats :

La Figure 4.5 est une représentation graphique des résultats du clustering pour les trois classes à savoir la classe négative,neutre et positive. Tel que, *Test 1* est le taux de réussite le plus bas et *Test 3*, celui avec le meilleur taux de réussite.



(a) Test 1 (b) Test 3

FIGURE 4.5 – Représentation graphique des clusters.

Tableau 4.1 – Résultats du clustering avec LDAxLSI.

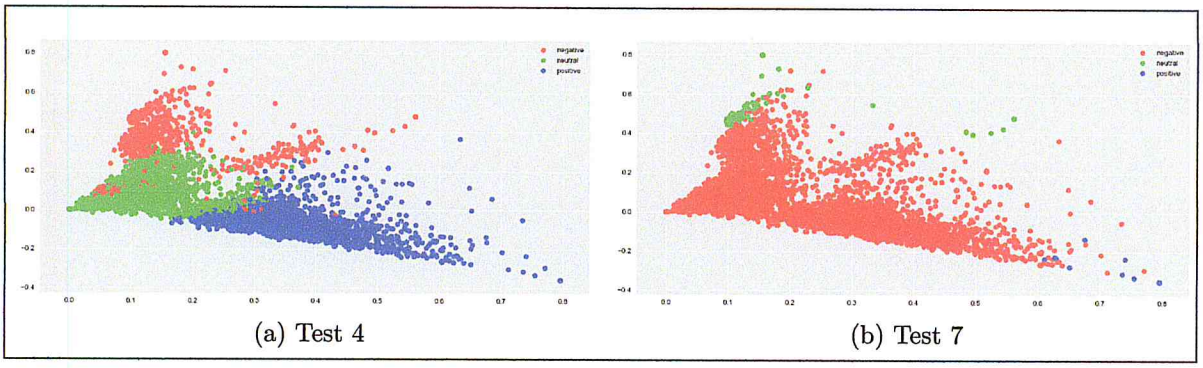
	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	LDA : standard --- LSI : standard	83.04 s	20.9 %	3.99 s
Test 2	LDA : chunksize = 1500 passes = 5 --- LSI : standard	166.15 s	50.68 %	2.84 s
Test 3	LDA : chunksize = 1500 passes = 7 --- LSI : standard	204.67 s	60.75 %	2.71 s
Test 4	LDA : chunksize = 1500 passes = 15 --- LSI : standard	382.72 s	52.56 %	2.68 s
Test 5	LDA : chunksize = 1000 passes = 7 --- LSI : standard	205.11 s	59.14 %	2.69 s
Test 6	LDA : chunksize = 1000 passes = 5 --- LSI : standard	150.23 s	58.33 %	2.74 s
Test 7	LDA : chunksize = 1500 passes = 7 --- LSI : chunksize = 1500	171.98 s	50.95 %	2.35 s

D'après les résultats du Tableau 4.1 on constate une augmentation ou diminution du temps d'entrainement en changeant la valeur des deux paramètres *chunksize* et *passes*. Pour la segmentation (clustering) des données de test via le modèle entrainé, le processus s'est effectué dans un intervalle de 2.35(s) à 3.99(s).

2. **K-means**

- **Résultats :**

Sur la Figure 4.6 ci-dessous, on peut visualiser les résultats du clustring sur le jeu de données de test.



(a) Test 4 (b) Test 7

FIGURE 4.6 – Représentation graphique des clusters.

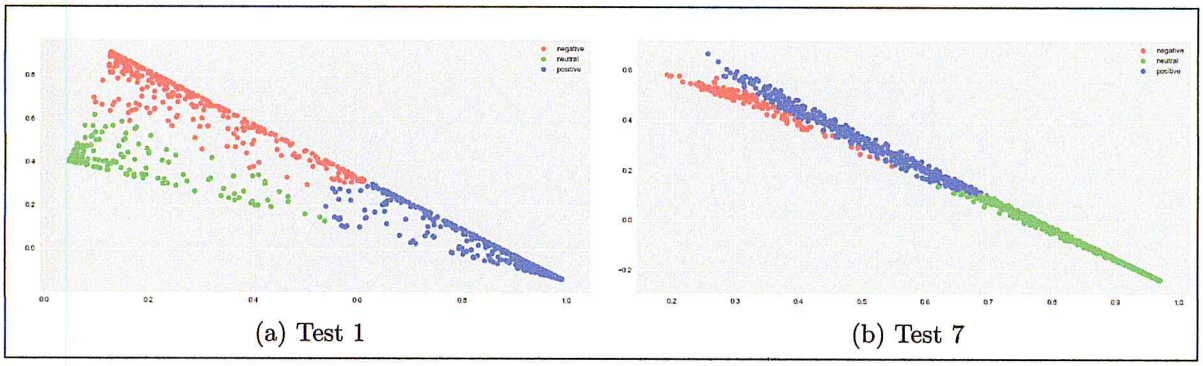
Tableau 4.2 – Résultats du clustering avec K-means.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	TF-IDF : standard --- Kmeans : standard	7.73 s	32.97 %	0.07 s
Test 2	TF-IDF : standard --- Kmeans : random_state = 0	7.35 s	32.97 %	0.04 s
Test 3	TF-IDF : standard --- Kmeans : random_state = 100	6.56 s	32.68 %	0.04 s
Test 4	TF-IDF : standard --- Kmeans : random_state = 125	7.63 s	34.67 %	0.03 s
Test 5	TF-IDF : max_df = 0.5 --- Kmeans : random_state = 125	7.17 s	34.67 %	0.05 s
Test 6	TF-IDF : standard --- Kmeans : random_state = 155	7.11 s	32.01 %	0.05 s
Test 7	TF-IDF : norm = L1 --- Kmeans : random_state = 125	9.32 s	29.18 %	0.08 s

3. Réseau de neurones

- **Résultats :**

La Figure 4.7 schématise le résultat de classification pour le 1er et 7ème test sur le corpus de test.



(a) Test 1 (b) Test 7

FIGURE 4.7 – Représentation graphique des classes.

Tableau 4.3 – Résultats de la classification avec réseau de neurones.

	Paramètres	Temps d'entrainement en secondes	Val_acc en pourcentage %	Temps de classification en secondes	Taux de réussite en pourcentage %
Test 1	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 32 Epochs = 5	47 s	78.07 %	2.85 s	74.37 %
Test 2	Input layer : input_dim = 10000 outputs = 512 ---------- Output layer : activation = sigmoid ---------- Loss = categorical Optimizer = adam Batch_size = 32 Epochs = 2	211.8 s	78.07 %	16.91 s	75.48 %
Test 3	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = binary Optimizer = adam Batch_size = 15 Epochs = 2	36.3 s	86.06 %	2.94 s	75.56 %
Test 4	Input layer : input_dim = 10000 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = adam Batch_size = 15 Epochs = 1	57.5 s	86.65 %	5.67 s	76.74 %
Test 5	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = rmsprop Batch_size = 32 Epochs = 5	46 s	79.59 %	3.78 s	76.15 %
Test 6	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = binary Optimizer = rmsprop Batch_size = 32 Epochs = 5	47 s	86.92 %	3.52 s	76.07 %
Test 7	Input layer : input_dim = 10000 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = binary Optimizer = rmsprop Batch_size = 132 Epochs = 5	23 s	85.81 %	3.72 s	78.81 %

Pour les tests de classification supervisée avec les réseaux de neurones (Tableau 4.3), on remarque qu'il y a une colonne en plus *val_acc*, c'est le taux de validation de l'entrainement. Donc, *validation_accuracy* est basse au début de l'entrainement et augmente au fur et à mesure. La valeur doit augmenter au moins un peu à travers les epochs, sinon si elle commence à diminuer cela voudra dire un sur-apprentissage (overfitting).

4.2.2 Comparaison en temps d'exécution et taux de réussite

D'après le tableau récapitulatif ci-dessous (Tableau 4.4), l'approche des réseaux de neurones à atteint le meilleur taux de réussite 78.81% suivi par le modèle hybridé LDA_LSI avec 60.75% et en dernière place l'algorithme K-means avec 34.67%

Tableau 4.4 – Comparaison des approche en temps d'exécution et taux de réussite.

	Temps d'entrainement en secondes	Temps de clustering/classification en seconde	Taux de réussite en pourcentage %
LDA x LSI	204.67 s	2.71 s	60.75 %
Kmeans	7.63 s	0.03 s	34.67 %
Réseau de neurones	23 s	3.72 s	78.81 %

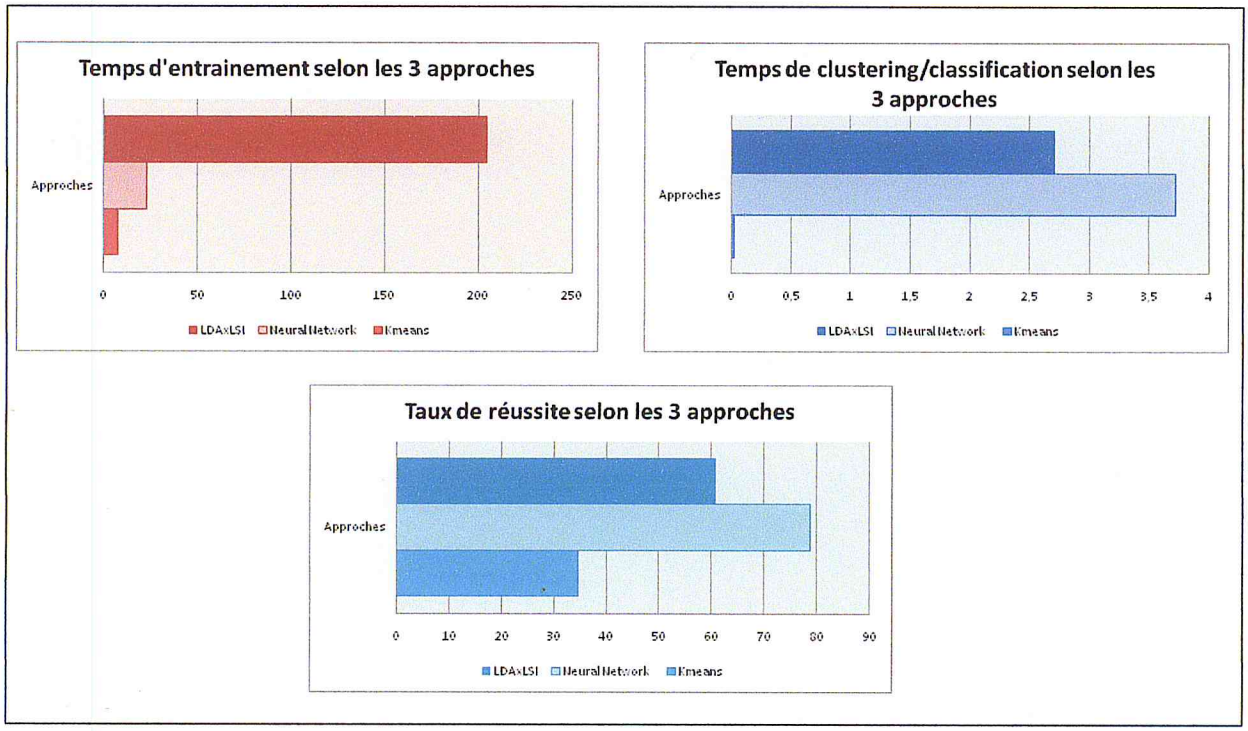


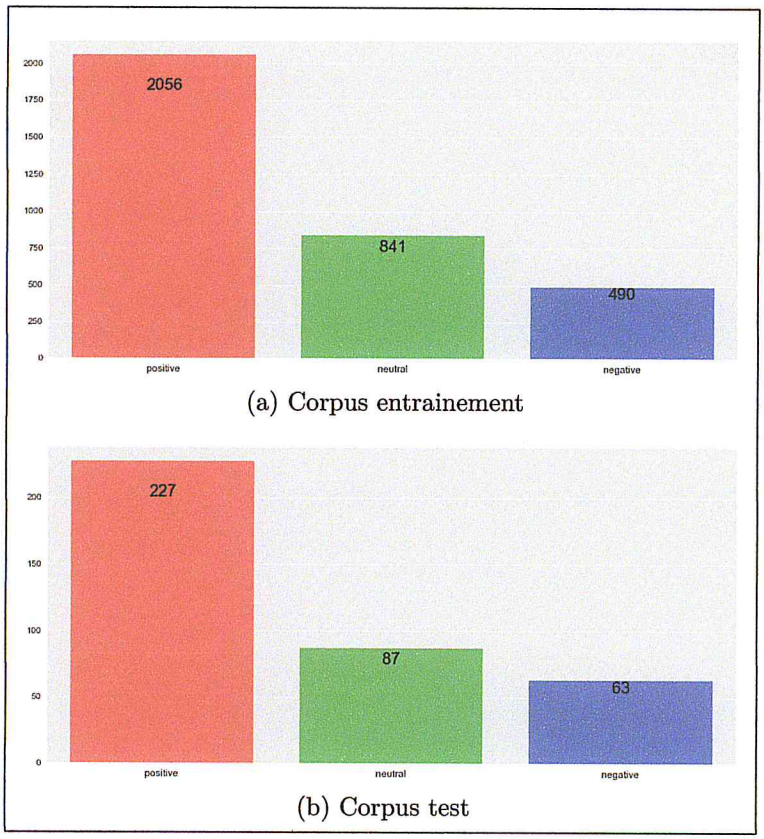
FIGURE 4.8 – Comparaison des approches en temps d'entrainement, segmentation/classification et taux de réussite

4.3 Dataset Coachella

L'ensemble de données "*Coachella Twitter sentiment*" contient des tweets au sujet de la programmation et l'organisation de l'évènement musical Coachella.

- Un total de 3 764 tweets.
- Un total de 3 classes ; positive, negative et neutral. Donc, pour chaque algorithme la variable K prendra la valeur 3.

Séparation de l'ensemble de données en : 3 387 tweets pour jeu de données d'entrainement et 377 tweets pour jeu de données de test



(a) Corpus entrainement

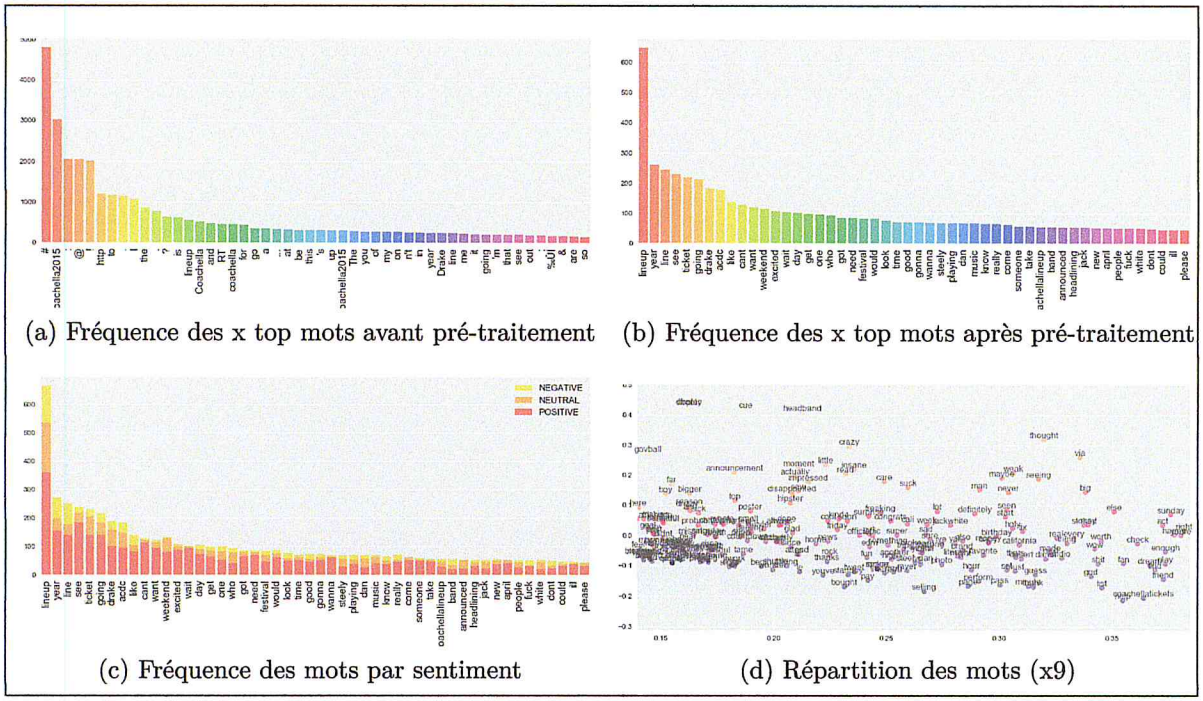
(b) Corpus test

FIGURE 4.9 – Séparation du dataset en jeu de données d'entrainement et de test.

4.3.1 Exposition des résultats

➢ **Pré-traitement**

1. **Jeu de données d'entrainement**

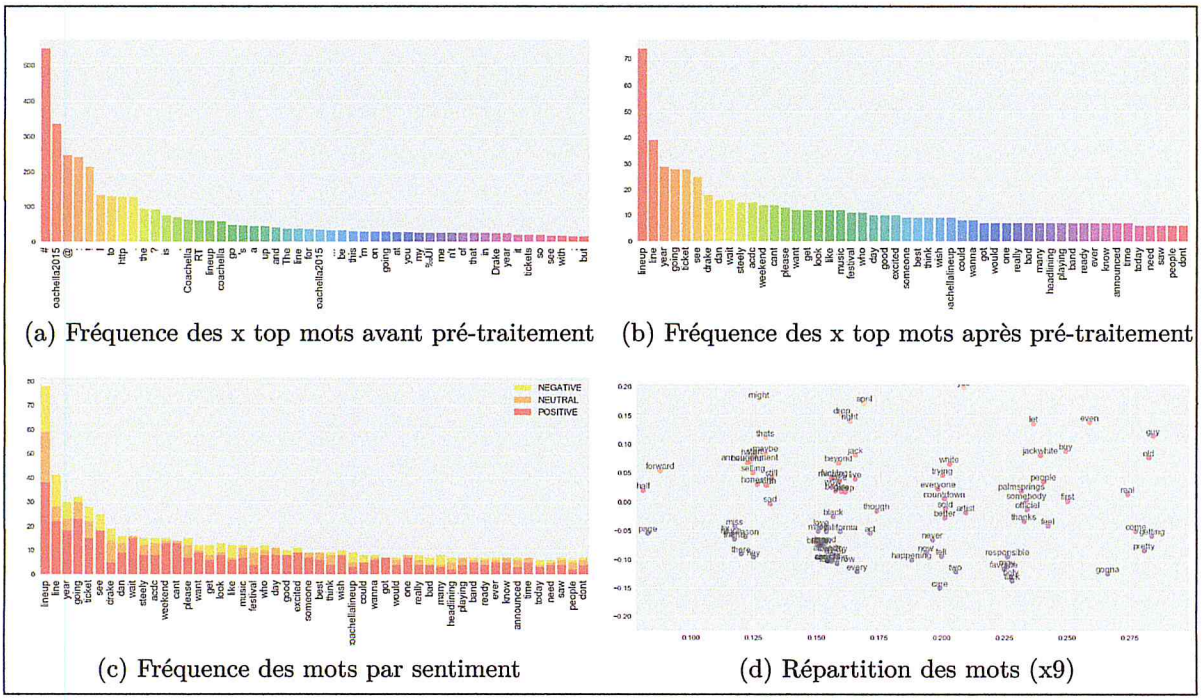


(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.10 – Coachella Twitter sentiment. -corpus d'entrainement-

2. **Jeu de données de test**



(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

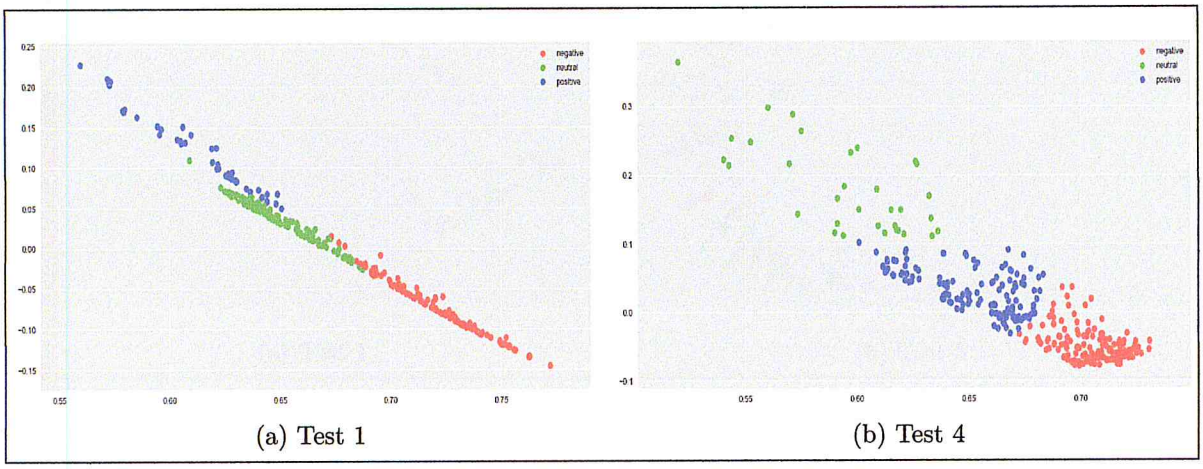
(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.11 – Coachella Twitter sentiment. -corpus de test-

➢ Clustering/Classification

1. Hybridation LDA_LSI

- **Résultats :**



(a) Test 1 (b) Test 4

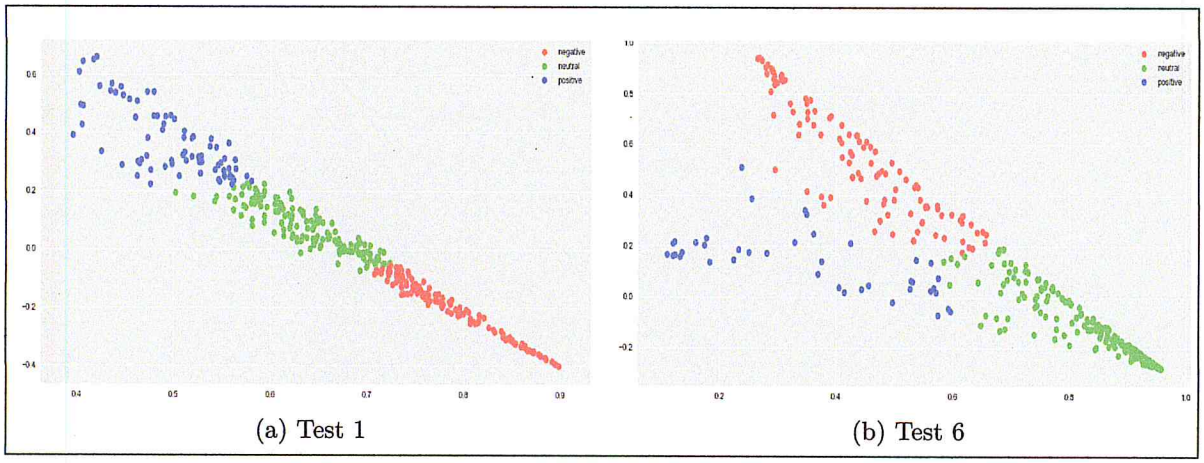
FIGURE 4.12 – Représentation graphique des clusters.

Tableau 4.5 – Résultats du clustering avec LDAxLSI.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	LDA : standard --- LSI : standard	35.9 s	20.07 %	0.91 s
Test 2	LDA : chunksize = 100 passes = 2 --- LSI : chunksize = 100	23.28 s	20.07 %	0.53 s
Test 3	LDA : chunksize = 1500 passes = 10 --- LSI : standard	58.65 s	20.07 %	0.65 s
Test 4	LDA : chunksize = 5 passes = 10 --- LSI : standard	107.14 s	20.04 %	0.53 s
Test 5	LDA : chunksize = 5 passes = 20 --- LSI : chunksize = 5	295.75 s	20.07 %	0.60 s
Test 6	LDA : chunksize = 2 passes = 20 --- LSI : standard	422.16 s	20.07 %	0.54 s
Test 7	LDA : chunksize = 3000 passes = 7 --- LSI : standard	49.97 s	20.07 %	0.70 s

3. Réseau de neurones

- **Résultats :**



(a) Test 1 (b) Test 6

FIGURE 4.14 – Représentation graphique des classes.

Tableau 4.7 – Résultats de la classification avec réseau de neurones.

	Paramètres	Temps d'entrainement en secondes	Val_acc en pourcentage %	Temps de classification en secondes	Taux de réussite en pourcentage %
Test 1	Input layer : input_dim = 3000 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 32 Epochs = 5	5 s	67.55 %	0.52 s	56.46 %
Test 2	Input layer : input_dim = 3000 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 5 Epochs = 5	10 s	70.21 %	0.56 s	59.45 %
Test 3	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 5 Epochs = 5	66 s	66.67 %	0.92 s	61.19 %
Test 4	Input layer : input_dim = 3000 outputs = 256 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 5 Epochs = 3	70 s	69.91 %	1.23 s	58.54 %
Test 5	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = rmsprop Batch_size = 5 Epochs = 3	33 s	73.92 %	1.09 s	57.09 %
Test 6	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = binary Optimizer = adam Batch_size = 5 Epochs = 3	38 s	80.14 %	1.14 s	64.87 %
Test 7	Input layer : input_dim = 3000 outputs = 256 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = adam Batch_size = 5 Epochs = 3	71 s	78.56 %	1.48 s	61.06 %

4.3.2 Comparaison en temps d'exécution et taux de réussite

Tableau 4.8 – Comparaison des approche en temps d'exécution et taux de réussite.

	Temps d'entrainement en secondes	Temps de clustering/classification en seconde	Taux de réussite en pourcentage %
LDA x LSI	147.61 s	0.65 s	20.07 %
Kmeans	0.59 s	0.05 s	38.59 %
Réseau de neurones	38 s	1.14 s	64.87 %

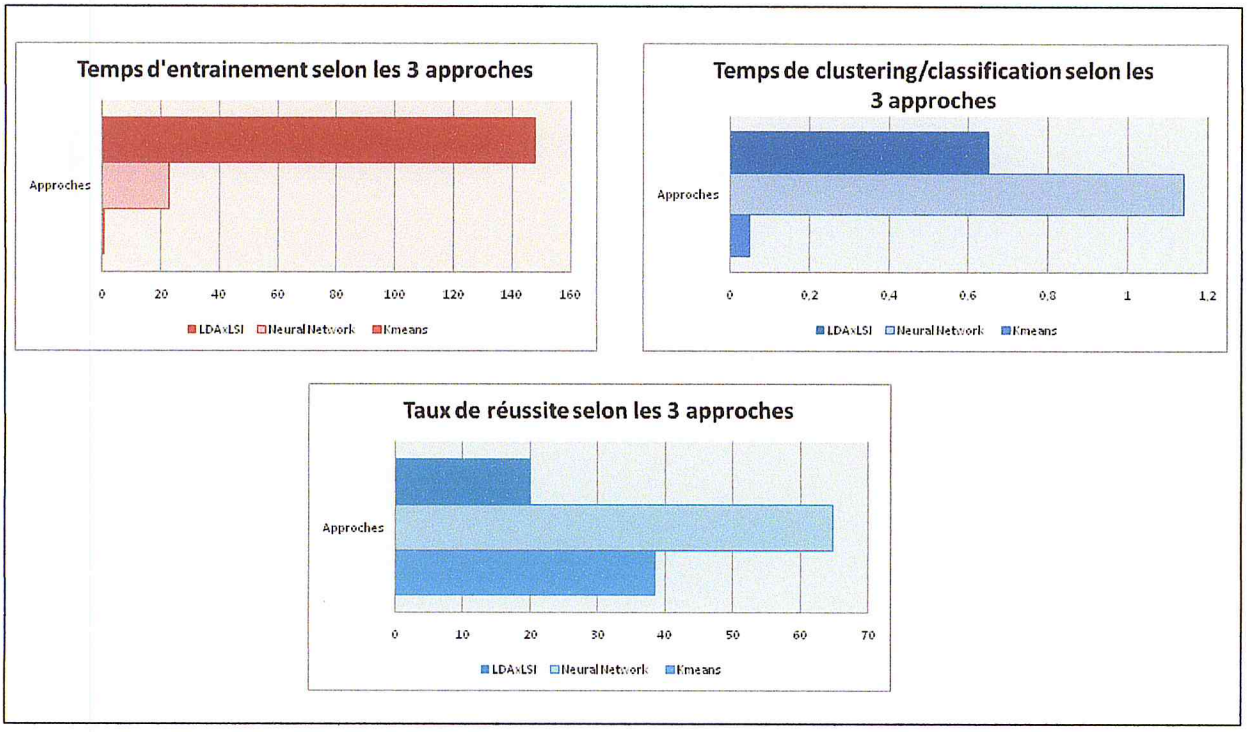


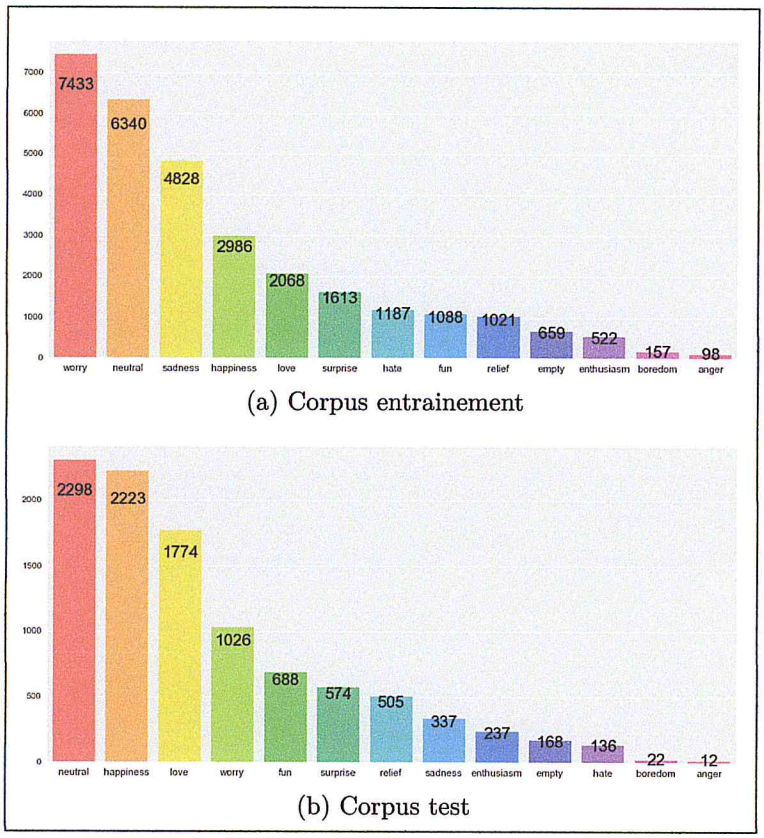
FIGURE 4.15 – Comparaison des approches en temps d'entrainement, segmentation/classification et taux de réussite

4.4 Dataset Emotions_v1

L'ensemble de données "*Emotions in text*" contient des tweets qui sont étiquetés selon l'émotion exprimée via le texte.

- Un total de 40 000 tweets.
- Un total de 13 classes ; anger, boredom, empty, enthusiasm, fun, happiness,hate, love, neutral, relief, sadness, surprise et worry. Donc, pour chaque algorithme la variable K prendra la valeur 13.

Séparation de l'ensemble de données en : 30 000 tweets pour jeu de données d'entrainement et 10 000 tweets pour jeu de données de test.



(a) Corpus entrainement

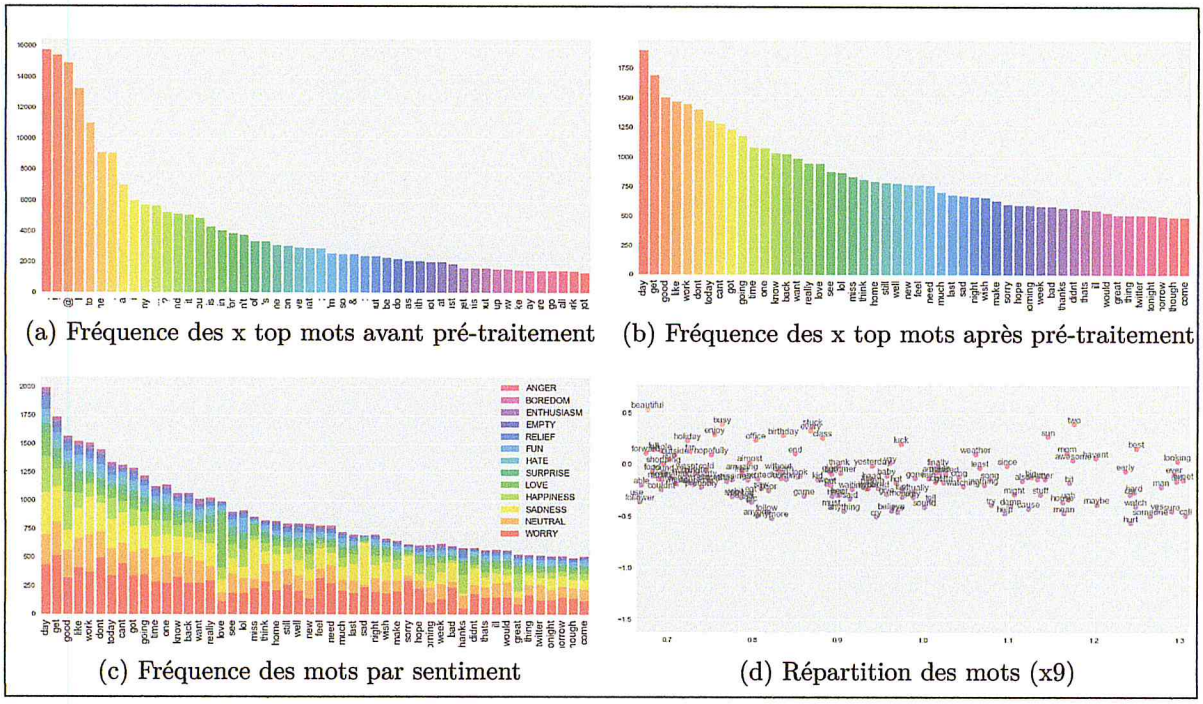
(b) Corpus test

FIGURE 4.16 – Séparation du dataset en jeu de données d'entrainement et de test.

4.4.1 Exposition des résultats

➢ **Pré-traitement**

1. **Jeu de données d'entrainement**

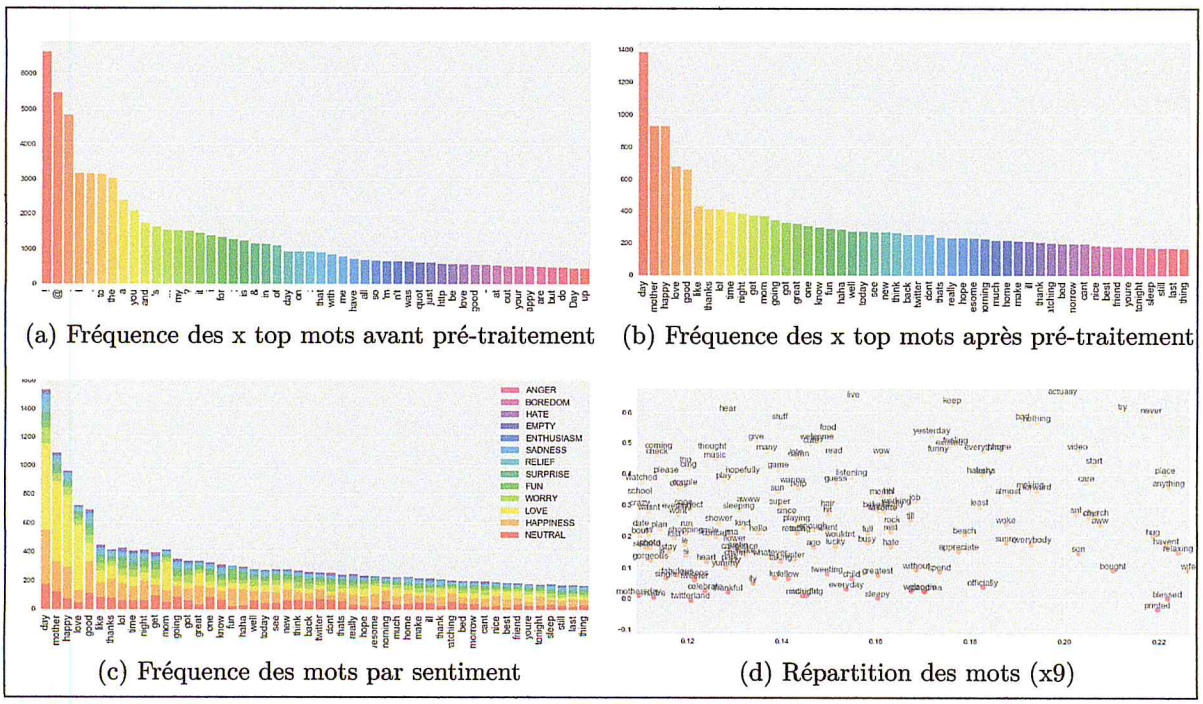


(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.17 – Emotions_v1 Twitter. -corpus d'entrainement-

2. **Jeu de données de test**



(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

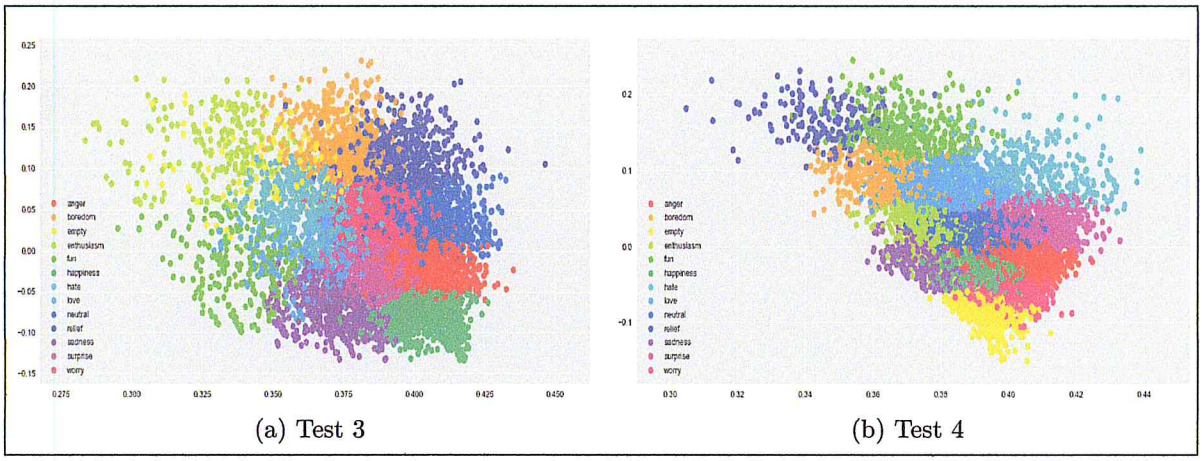
(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.18 – Emotions_v1 Twitter. -corpus de test-

➢ Clustering/Classification

1. Hybridation LDA_LSI

- **Résultats :**



(a) Test 3 (b) Test 4

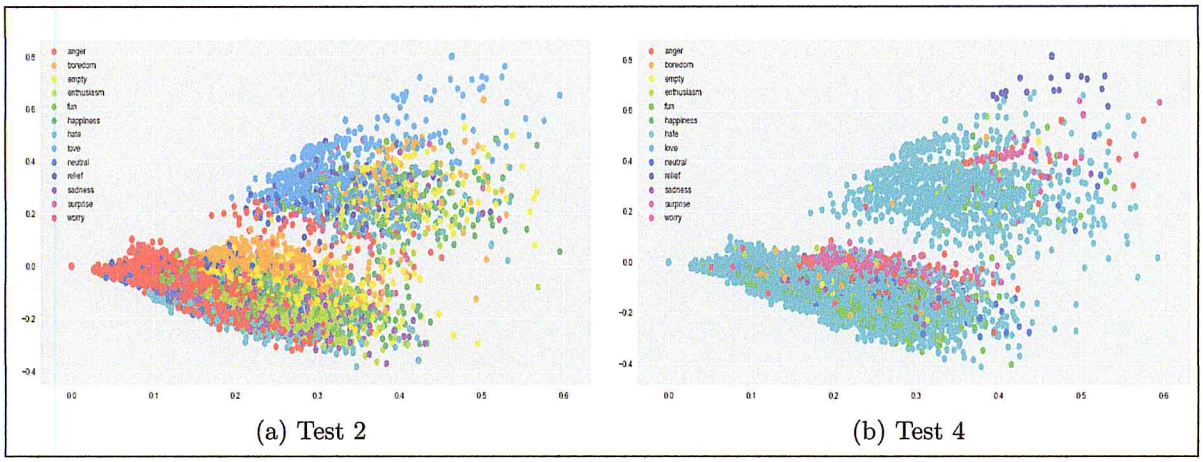
FIGURE 4.19 – Représentation graphique des clusters.

Tableau 4.9 – Résultats du clustering avec LDAxLSI.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	LDA : chunksize = 5000 passes = 50 --- LSI : standard	2254.82 s	8.37 %	17.84 s
Test 2	LDA : chunksize = 5000 passes = 75 iterations = 100 --- LSI : chunksize = 5000 power_iters = 100	4313.01 s	10.35 %	18.97 s
Test 3	LDA : chunksize = 5000 passes = 350 iterations = 100 --- LSI : chunksize = 5000 power_iters = 2	15607.51 s	10.77 %	18.26 s
Test 4	LDA : standard --- LSI : standard	315.49 s	5.8 %	21.69 s

2. K-means

- **Résultats :**



(a) Test 2 (b) Test 4

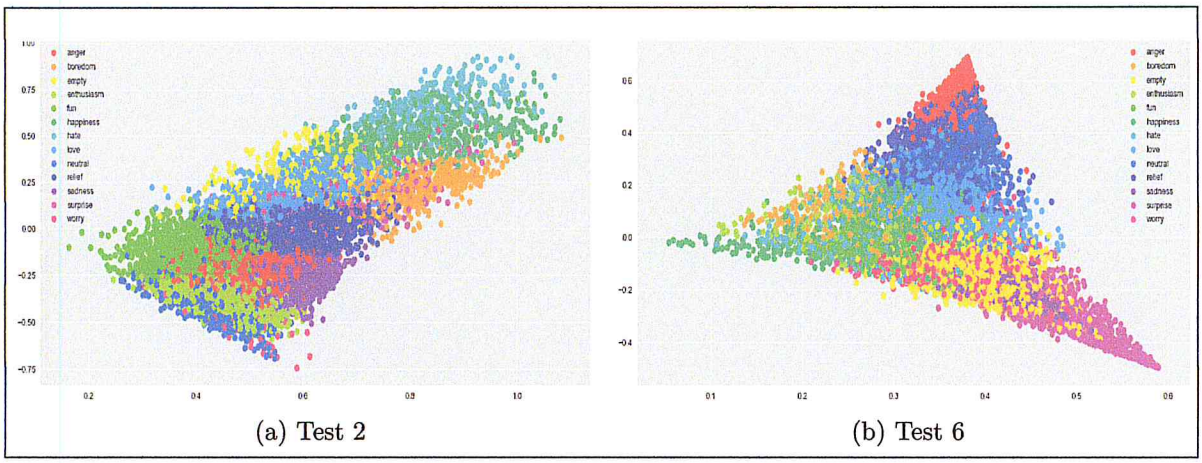
FIGURE 4.20 – Représentation graphique des clusters.

Tableau 4.10 – Résultats du clustering avec K-means.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	TF-IDF : standard ---------- Kmeans : standard	14.60 s	7.53 %	0.03 s
Test 2	TF-IDF : standard ---------- Kmeans : random_state = 0	16.33 s	7.08 %	0.07 s
Test 3	TF-IDF : standard ---------- Kmeans : max_iter = 11	11.27 s	7.75 %	0.03 s
Test 4	TF-IDF : standard ---------- Kmeans : random_state = 125	17.40 s	9.29 %	0.06 s
Test 5	TF-IDF : standard ---------- Kmeans : random_state = 0 max_iter = 11	11.83 s	8.32 %	0.05 s
Test 6	TF-IDF : standard ---------- Kmeans : random_state = 0 max_iter = 155	15.75 s	8.32 %	0.07 s
Test 7	TF-IDF : standard ---------- Kmeans : random_state = 100	20.32 s	8.23 %	0.01 s

3. Réseau de neurones

- **Résultats :**



(a) Test 2 (b) Test 6

FIGURE 4.21 – Représentation graphique des classes.

Tableau 4.11 – Résultats de la classification avec réseau de neurones.

	Paramètres	Temps d'entrainement en secondes	Val_acc en pourcentage %	Temps de classification en secondes	Taux de réussite en pourcentage %
Test 1	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = adam Batch_size = 35 Epochs = 2	43 s	92.20 %	19.75 s	21.70 %
Test 2	Input layer : input_dim = 10000 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = categorical Optimizer = rmsprop Batch_size = 132 Epochs = 3	52 s	33.67 %	37.12 s	23.02 %
Test 3	Input layer : input_dim = 10000 outputs = 1024 ---------- Output layer : activation = sigmoid ---------- Loss = categorical Optimizer = rmsprop Batch_size = 132 Epochs = 5	819 s	29.27 %	219.78 s	17.92 %
Test 4	Input layer : input_dim = 10000 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = rmsprop Batch_size = 45 Epochs = 3	155.06 s	92.19 %	36.38 s	21.61 %
Test 5	Input layer : input_dim = 3000 outputs = 512 ---------- Output layer : activation = sigmoid ---------- Loss = categorical Optimizer = rmsprop Batch_size = 145 Epochs = 3	72 s	33.20 %	43.23 s	20.08 %
Test 6	Input layer : input_dim = 20000 outputs = 128 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 123 Epochs = 5	1056.37 s	29.30 %	62.25 s	17.45 %
Test 7	Input layer : input_dim = 300 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 128 Epochs = 200	201 s	31.23 %	10.91 s	22.86

4.4.2 Comparaison en temps d'exécution et taux de réussite

Tableau 4.12 – Comparaison des approche en temps d'exécution et taux de réussite.

	Temps d'entrainement en secondes	Temps de clustering/classification en seconde	Taux de réussite en pourcentage %
LDA x LSI	15607.51 s	18.26 s	10.77 %
Kmeans	17.40 s	0.06 s	9.29 %
Réseau de neurones	52 s	37.12 s	23.02 %

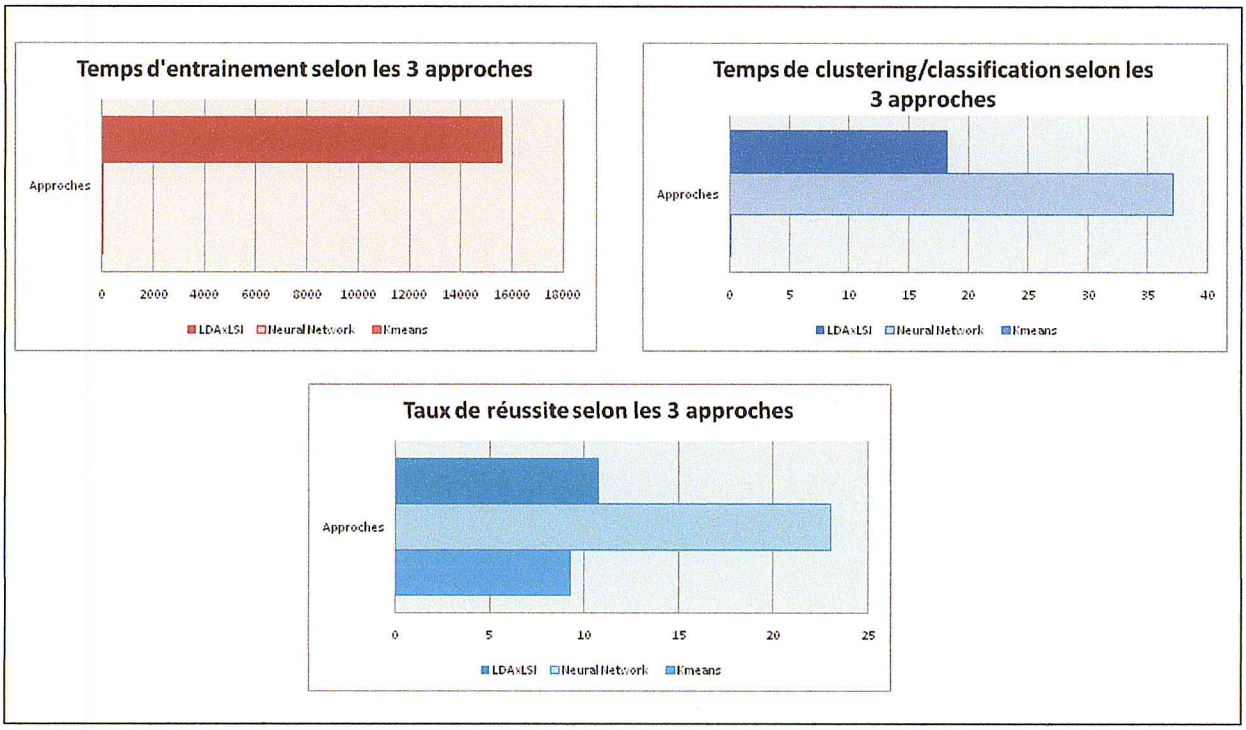


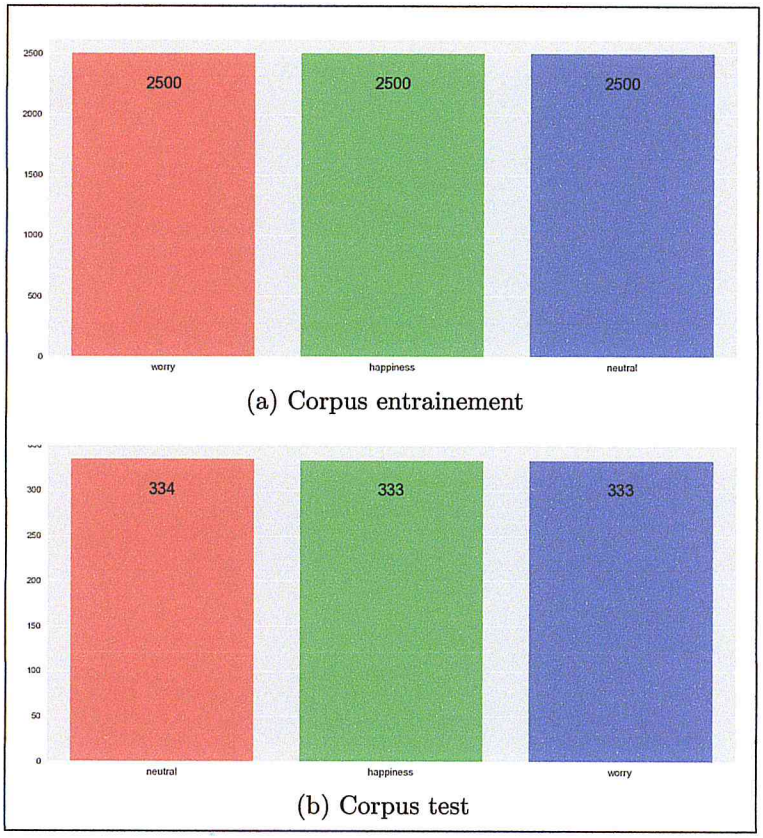
FIGURE 4.22 – Comparaison des approches en temps d'entrainement, segmentation/classification et taux de réussite

4.5 Dataset Emotions_v2

Vus le nombre de classes important (13 classes) et les taux de réussite obtenus dans la première version (Dataset Emotions_v1) étaient faible, on a décidé de réduire le nombre de classes à 3 et refaire le processus.

- Un total de 8 500 tweets.
- Un total de 3 classes ; happiness, neutral et worry. Donc, pour chaque algorithme la variable K prendra la valeur 3.

Séparation de l'ensemble de données en : 7 500 tweets pour jeu de données d'entrainement et 1 000 tweets pour jeu de données de test.



(a) Corpus entrainement

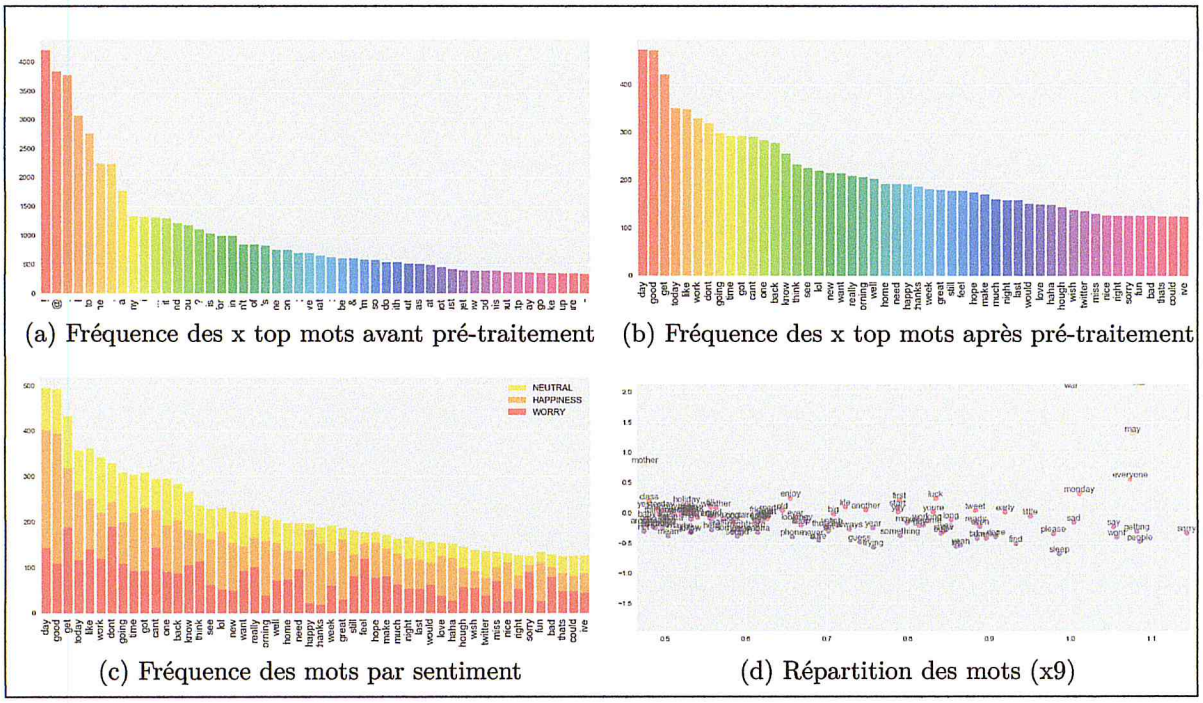
(b) Corpus test

FIGURE 4.23 – Séparation du dataset en jeu de données d'entrainement et de test.

4.5.1 Exposition des résultats

➢ **Pré-traitement**

1. **Jeu de données d'entrainement**

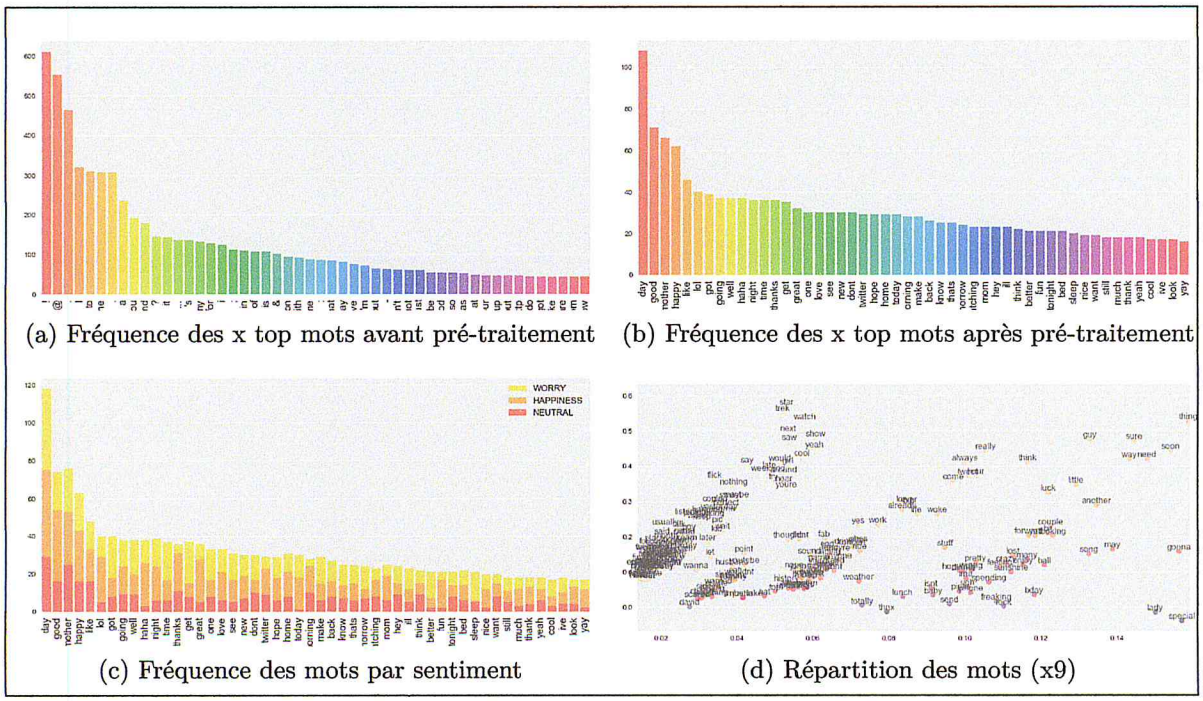


(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.24 – Emotions_v2 Twitter. -corpus d'entrainement-

2. **Jeu de données de test**



(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

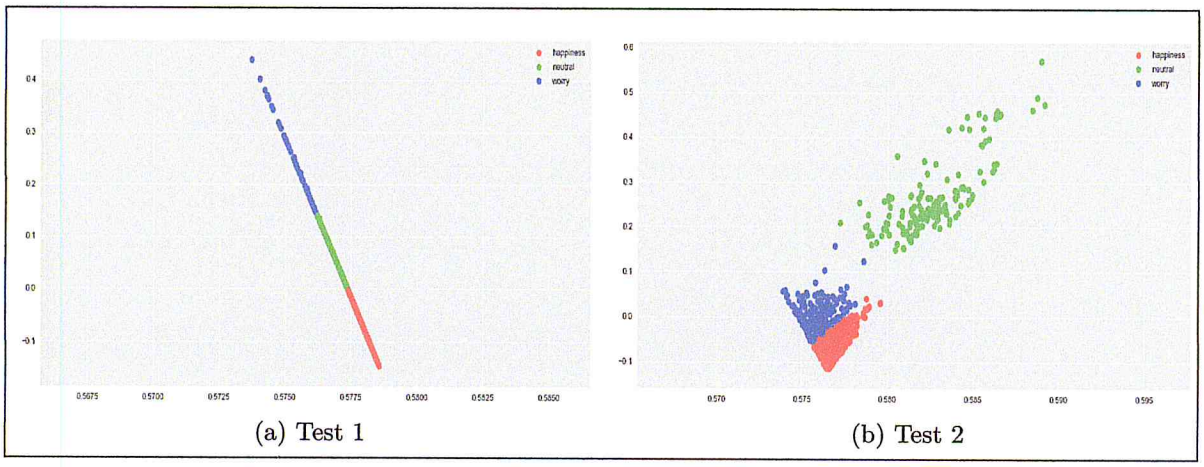
(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.25 – Emotions_v2 Twitter. -corpus de test-

➢ Clustering/Classification

1. Hybridation LDA_LSI

- **Résultats :**



(a) Test 1 (b) Test 2

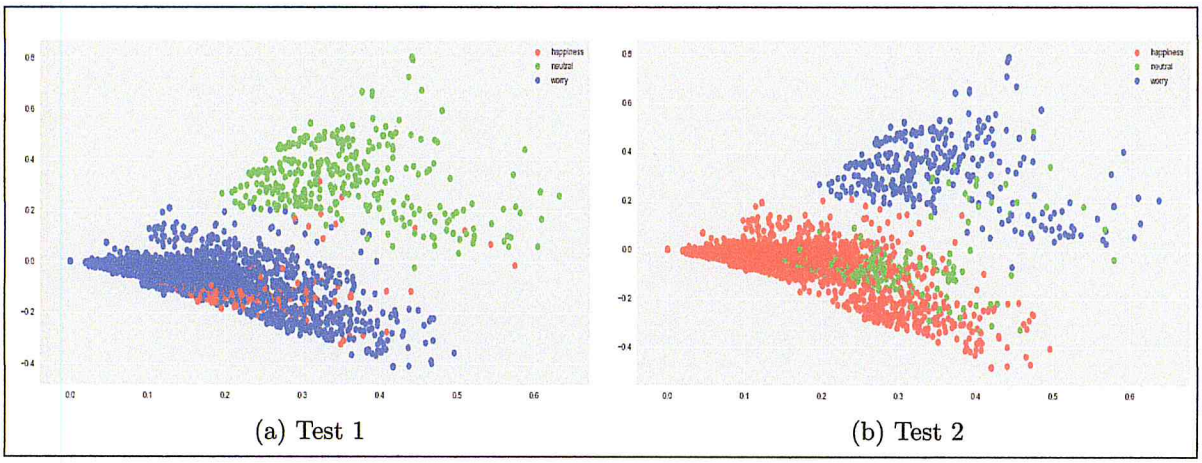
FIGURE 4.26 – Représentation graphique des clusters.

Tableau 4.13 – Résultats du clustering avec LDAxLSI.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	LDA : standard --- LSI : standard	43.53 s	37.93 %	2.44 s
Test 2	LDA : chunksize = 5000 passes = 10 --- LSI : standard	229.89 s	44.64 %	2.03 s
Test 3	LDA : chunksize = 5000 passes = 50 --- LSI : standard	792.1 s	42.35 %	1.84 s
Test 4	LDA : chunksize = 1500 passes = 15 --- LSI : standard	207.09 s	44.62 %	2.07 s
Test 5	LDA : chunksize = 1500 passes = 5 random_state = 10 --- LSI : standard	94.66 s	42.61 %	1.91 s
Test 6	LDA : chunksize = 1500 passes = 7 --- LSI : standard	123.78 s	41.53 %	1.83 s
Test 7	LDA : chunksize = 100 passes = 7 --- LSI : chunksize = 100	79.32	39.19 %	1.17 s

2. **K-means**

- **Résultats :**



(a) Test 1 (b) Test 2

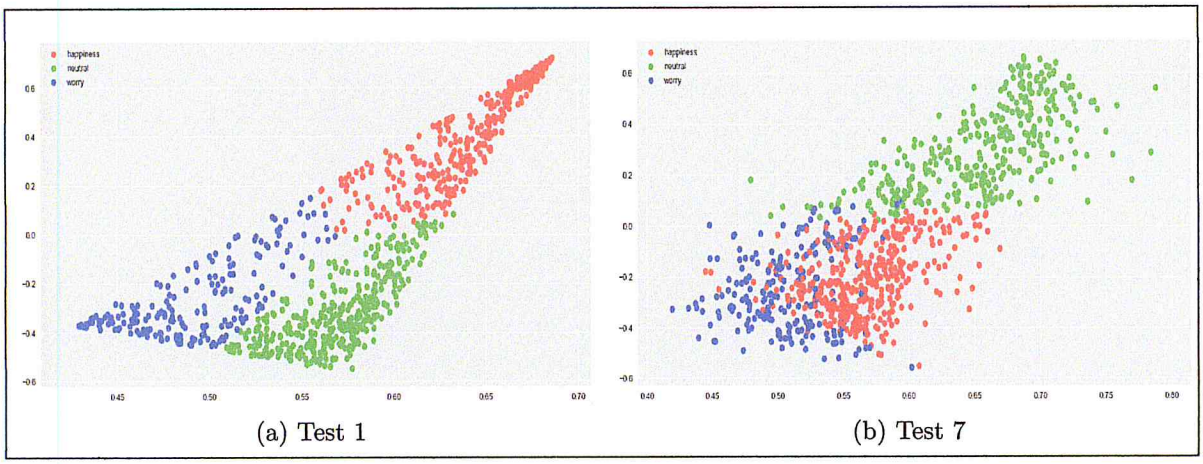
FIGURE 4.27 – Représentation graphique des clusters.

Tableau 4.14 – Résultats du clustering avec K-means.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	TF-IDF : standard --- Kmeans : standard	4.11 s	36.04 %	0.01 s
Test 2	TF-IDF : standard --- Kmeans : random_state = 0	1.78 s	30.7 %	0.04 s
Test 3	Cosine --- Kmeans : random_state = 125	55.59 s	33.55 %	0.11 s
Test 4	Cosine --- Kmeans : standard	50.60 s	34.36 %	0.10 s
Test 5	TF-IDF x Cosine --- Kmeans : standard	60.50 s	34.45 %	0.11 s
Test 6	TF-IDF x Cosine --- Kmeans : random_state = 125	52.03 s	32.52 %	0.11 s
Test 7	TF-IDF x Cosine --- Kmeans : max_iter = 11	45.87 s	33.35 %	0.11 s

3. Réseau de neurones

- **Résultats :**



(a) Test 1 (b) Test 7

FIGURE 4.28 – Représentation graphique des classes.

Tableau 4.15 – Résultats de la classification avec réseau de neurones.

	Paramètres	Temps d'entrainement en secondes	Val_acc en pourcentage %	Temps de classification en secondes	Taux de réussite en pourcentage %
Test 1	Input layer : input_dim = 3000 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 35 Epochs = 10	11 s	52.53 %	1.71 s	50.60 %
Test 2	Input layer : input_dim = 144 outputs = 144 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 45 Epochs = 5	6 s	47.73 %	2.01 s	50.76 %
Test 3	Input layer : input_dim = 4000 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 35 Epochs = 5	6 s	47.73 %	1.35 s	51.68 %
Test 4	Input layer : input_dim = 3000 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = adam Batch_size = 35 Epochs = 2	20.02 s	73.51 %	7.67 s	50.41 %
Test 5	Input layer : input_dim = 4000 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = adam Batch_size = 32 Epochs = 2	15.02 s	71.96 %	2.46 s	51.61 %
Test 6	Input layer : input_dim = 4000 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = adam Batch_size = 32 Epochs = 3	22.03 s	67.33 %	2.61 s	52.82 %
Test 7	Input layer : input_dim = 3500 outputs = 130 ---------- Output layer : activation = sigmoid ---------- Loss = binary Optimizer = adam Batch_size = 32 Epochs = 2	14.01 s	72.18 %	3.11 s	53.03 %

4.5.2 Comparaison en temps d'exécution et taux de réussite

Tableau 4.16 – Comparaison des approche en temps d'exécution et taux de réussite.

	Temps d'entrainement en secondes	Temps de clustering/classification en seconde	Taux de réussite en pourcentage %
LDA x LSI	229.89 s	2.03 s	44.64 %
Kmeans	4.11 s	0.01 s	36.04 %
Réseau de neurones	14.01 s	3.11 s	53.03 %

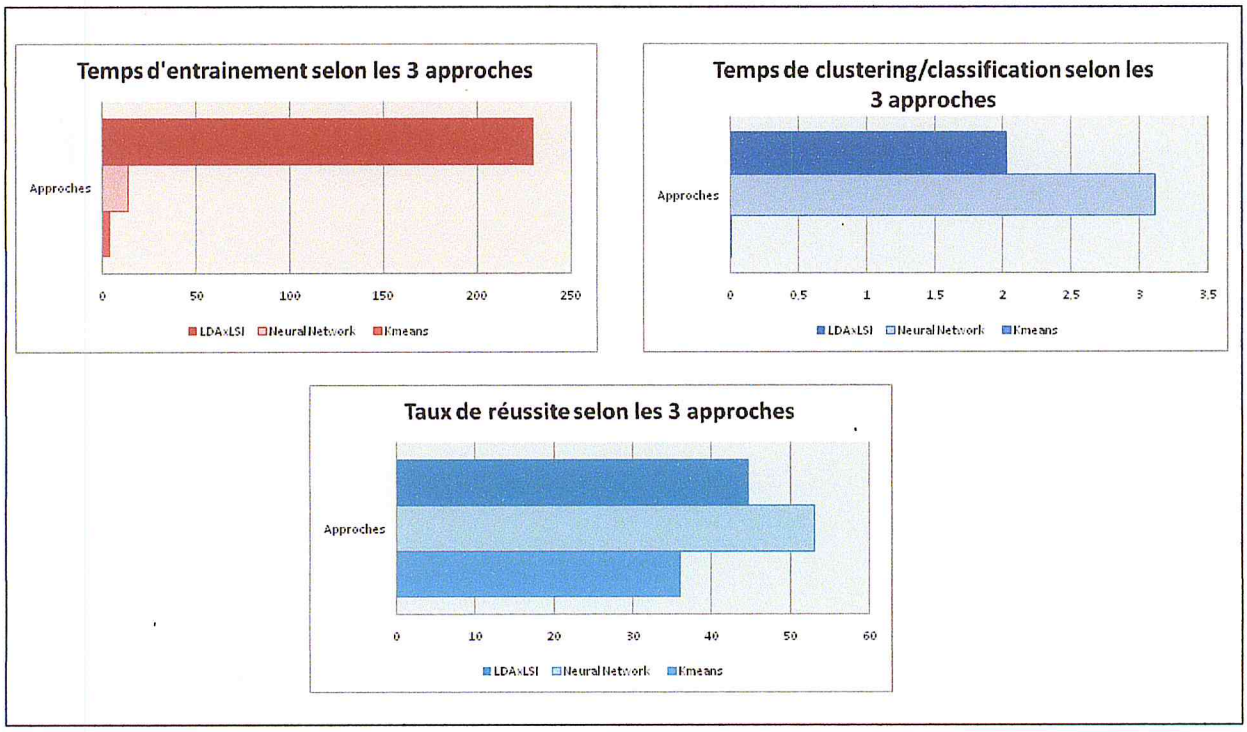


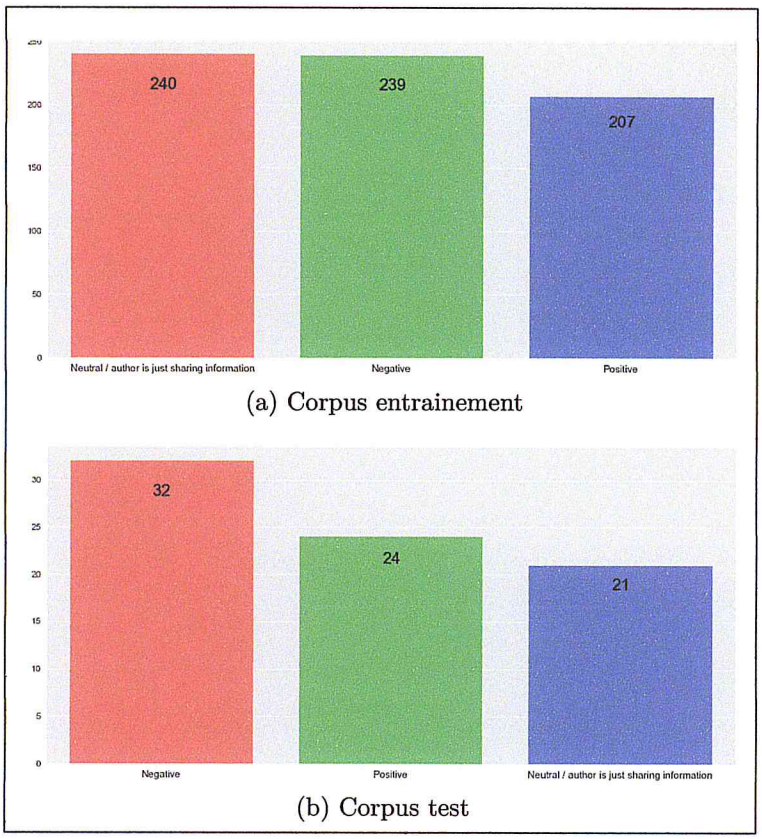
FIGURE 4.29 – Comparaison des approches en temps d'entrainement, segmentation/classification et taux de réussite

4.6 Dataset Weather

L'ensemble de données "*Weather sentiment*" contient des tweets en relation avec la météo.

- Un total de 763 tweets.
- Un total de 3 classes ; positive, negative et neutral. Donc, pour chaque algorithme la variable K prendra la valeur 3.

Séparation de l'ensemble de données en : 686 tweets pour jeu de données d'entrainement et 77 tweets pour jeu de données de test.



(a) Corpus entrainement

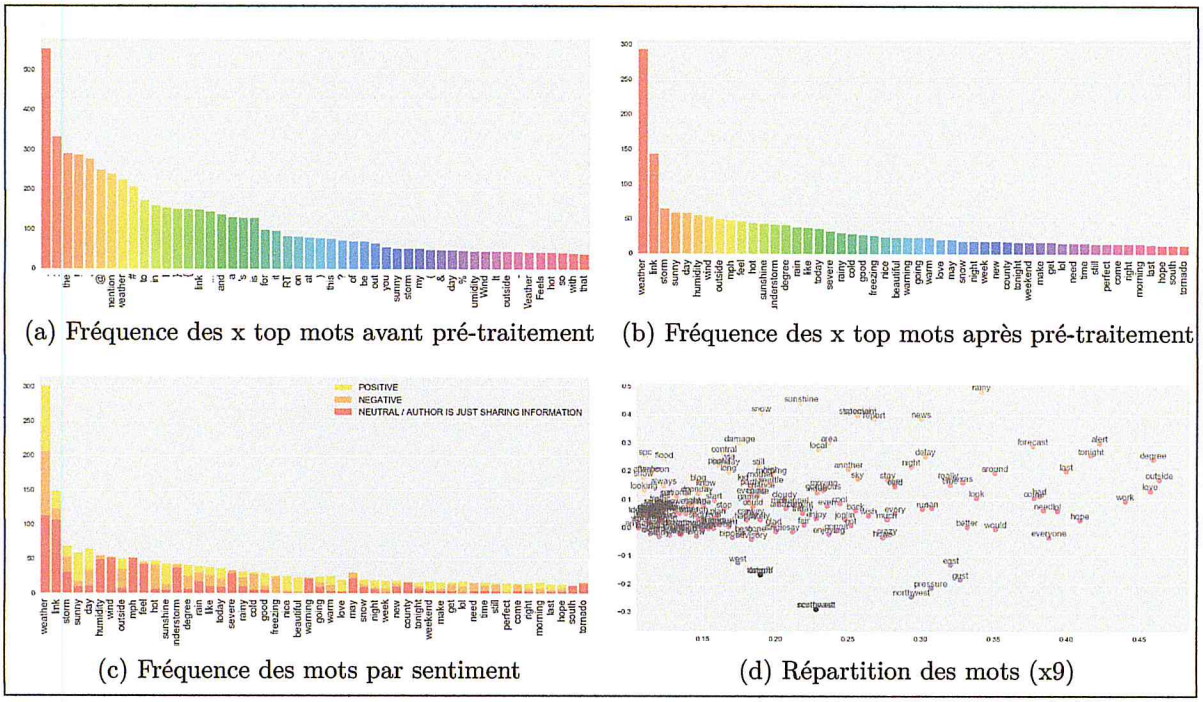
(b) Corpus test

FIGURE 4.30 – Séparation du dataset en jeu de données d'entrainement et de test.

4.6.1 Exposition des résultats

➢ **Pré-traitement**

1. **Jeu de données d'entrainement**

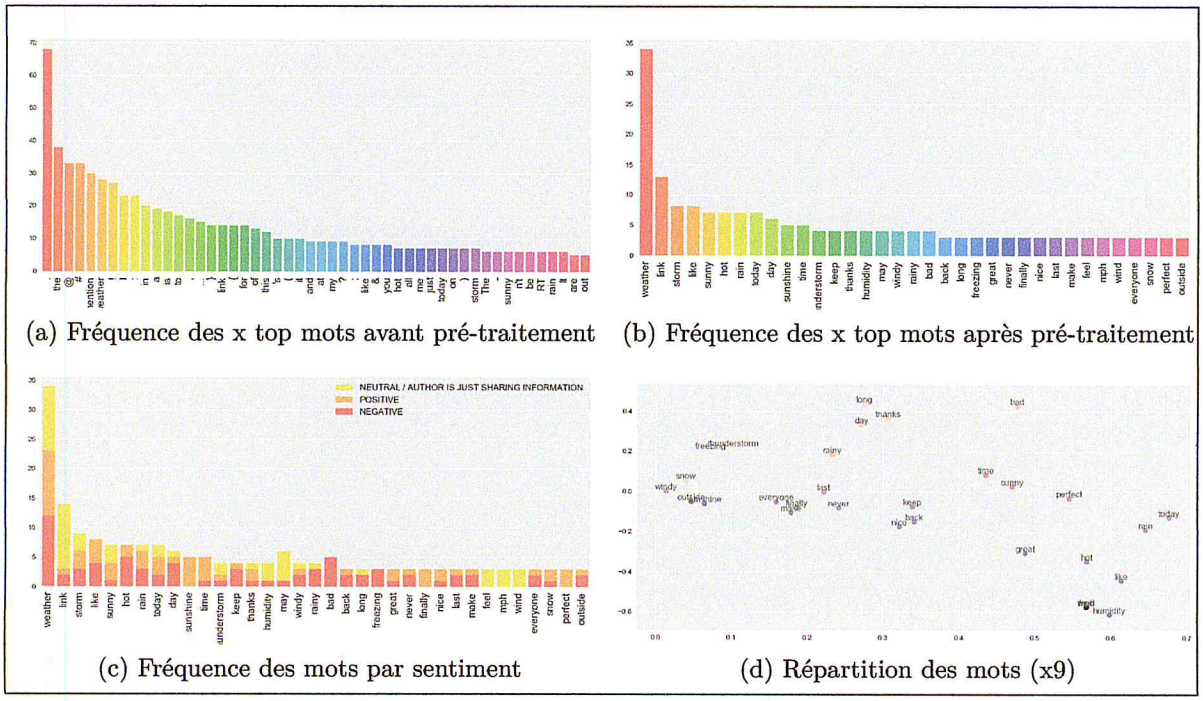


(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.31 – Weather sentiment. -corpus d'entrainement-

2. **Jeu de données de test**



(a) Fréquence des x top mots avant pré-traitement (b) Fréquence des x top mots après pré-traitement

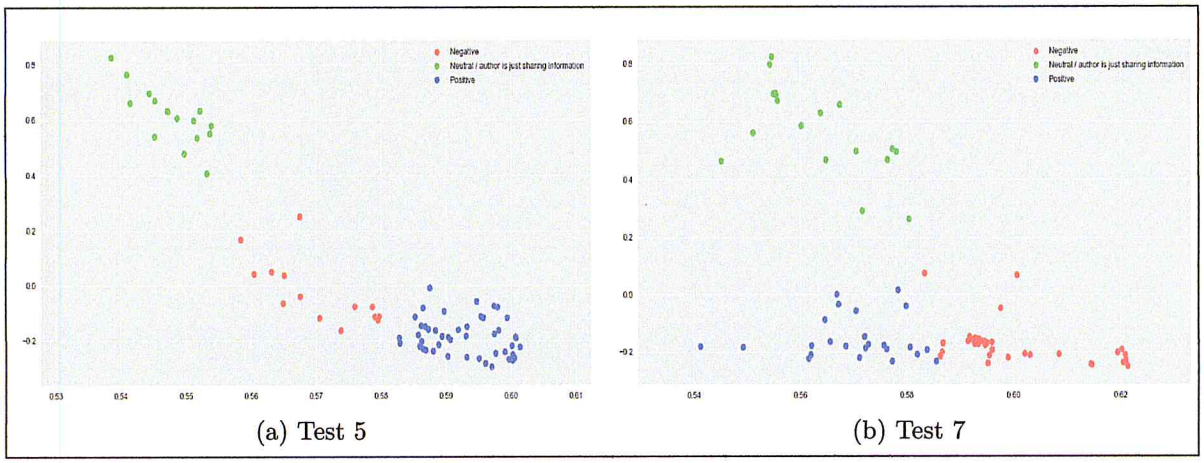
(c) Fréquence des mots par sentiment (d) Répartition des mots (x9)

FIGURE 4.32 – Weather sentiment. -corpus de test-

➢ Clustering/Classification

1. Hybridation LDA_LSI

- **Résultats :**



(a) Test 5 (b) Test 7

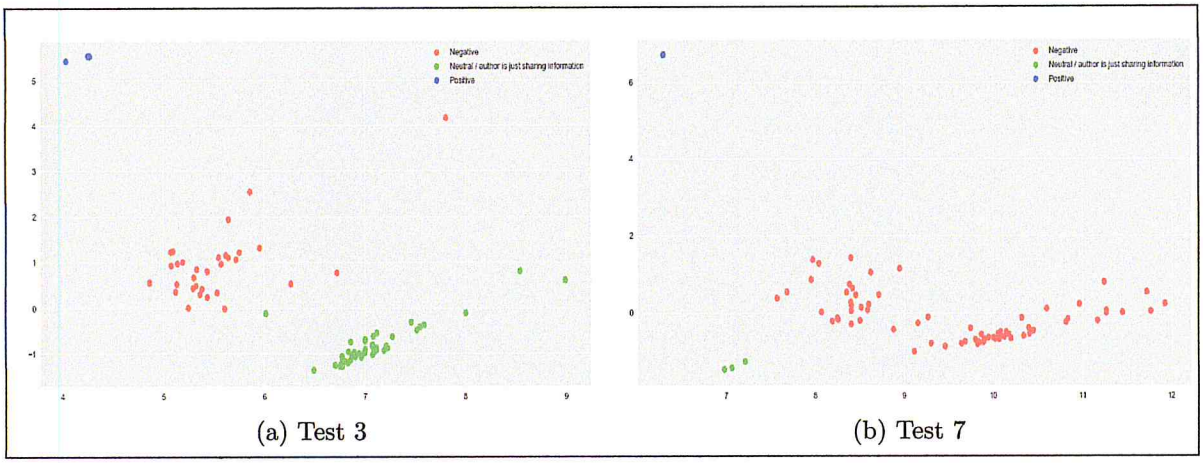
FIGURE 4.33 – Représentation graphique des clusters.

Tableau 4.17 – Résultats du clustering avec LDAxLSI.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	LDA : standard --- LSI : standard	7.7 s	58.62 %	0.25 s
Test 2	LDA : chunksize = 100 passes = 5 --- LSI : standard	6.27 s	58.02 %	0.18 s
Test 3	LDA : chunksize = 5 passes = 2 --- LSI : standard	6.51 s	60.90 %	0.15 s
Test 4	LDA : chunksize = 5 passes = 7 --- LSI : standard	10.84 s	62.98 %	0.14 s
Test 5	LDA : chunksize = 5 passes = 10 --- LSI : standard	14.38 s	52.24 %	0.13 s
Test 6	LDA : chunksize = 5 passes = 7 --- LSI : chunksize = 5	15.09 s	58.42 %	0.14 s
Test 7	LDA : chunksize = 5 passes = 7 --- LSI : chunksize = 5 power_iters = 7	19.02 s	63.68 %	0.16 s

2. **K-means**

- **Résultats :**



(a) Test 3 (b) Test 7

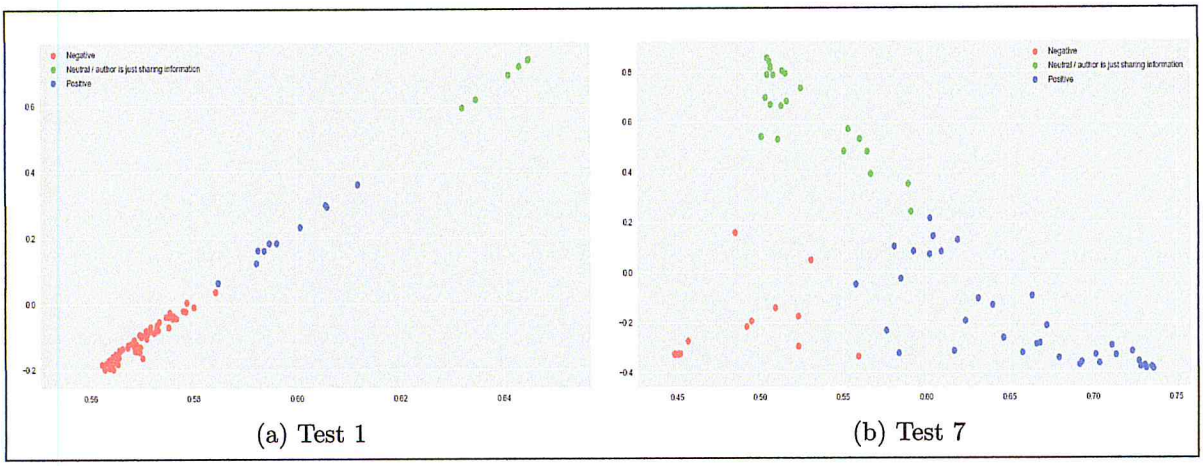
FIGURE 4.34 – Représentation graphique des clusters.

Tableau 4.18 – Résultats du clustering avec K-means.

	Paramètres	Temps d'entrainement en secondes	Taux de réussite en pourcentage %	Temps de clustering en secondes
Test 1	TF-IDF : standard --- Kmeans : standard	0.53 s	29.38 %	0.01 s
Test 2	TF-IDF : standard --- Kmeans : random_state = 125	0.76 s	39.36 %	0.01 s
Test 3	Cosine --- Kmeans : standard	0.43 s	21.63 %	0.01 s
Test 4	Cosine --- Kmeans : random_state = 100	0.43 s	22.89 %	0.01 s
Test 5	TF-IDF x Cosine --- Kmeans : standard	0.55 s	38.40 %	0.01 s
Test 6	TF-IDF x Cosine --- Kmeans : random_state = 100	0.54 s	45.02 %	0.19 s
Test 7	TF-IDF x Cosine --- Kmeans : random_state = 100 max_iter = 5	0.45 s	45.02 %	0.01 s

3. Réseau de neurones

- **Résultats :**



(a) Test 1 (b) Test 7

FIGURE 4.35 – Représentation graphique des classes.

Tableau 4.19 – Résultats de la classification avec réseau de neurones.

	Paramètres	Temps d'entrainement en secondes	Val_acc en pourcentage %	Temps de classification en secondes	Taux de réussite en pourcentage %
Test 1	Input layer : input_dim = 100 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 5 Epochs = 5	2 s	66.67 %	0.21 s	64.28 %
Test 2	Input layer : input_dim = 100 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 5 Epochs = 20	10 s	69.57 %	0.26 s	78.49 %
Test 3	Input layer : input_dim = 100 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 5 Epochs = 100	50 sec	68.12 %	0.30 s	71.53 %
Test 4	Input layer : input_dim = 100 outputs = 8 ---------- Output layer : activation = softmax ---------- Loss = binary Optimizer = adam Batch_size = 5 Epochs = 20	10 s	82.61 %	0.35 s	78.49 %
Test 5	Input layer : input_dim = 100 outputs = 128 ---------- Output layer : activation = sigmoid ---------- Loss = categorical Optimizer = rmsprop Batch_size = 5 Epochs = 30	15 s	73.91 %	0.40 s	76.23 %
Test 6	Input layer : input_dim = 2000 outputs = 256 ---------- Output layer : activation = softmax ---------- Loss = binary Optimizer = adam Batch_size = 2 Epochs = 5	42.16 s	84.54 %	0.80 s	79.31 %
Test 7	Input layer : input_dim = 2500 outputs = 256 ---------- Output layer : activation = softmax ---------- Loss = categorical Optimizer = adam Batch_size = 2 Epochs = 3	33.54 s	76.81 %	1.12 s	79.40 %

4.6.2 Comparaison en temps d'exécution et taux de réussite

Tableau 4.20 – Comparaison des approche en temps d'exécution et taux de réussite.

	Temps d'entrainement en secondes	Temps de clustering/classification en seconde	Taux de réussite en pourcentage %
LDA x LSI	19.02 s	0.16 s	63.68 %
Kmeans	0.54 s	0.01 s	45.02 %
Réseau de neurones	33.54 s	1.12 s	79.40 %

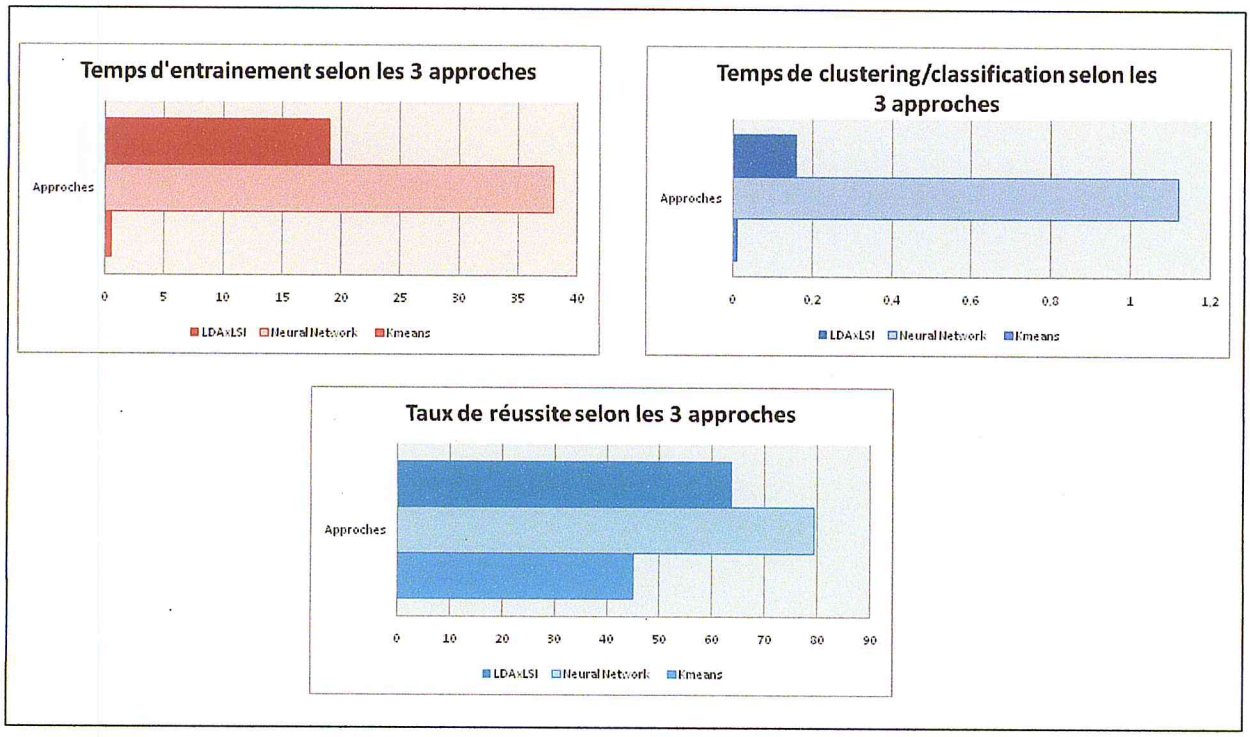


FIGURE 4.36 – Comparaison des approches en temps d'entrainement, segmentation/classification et taux de réussite

4.7 Analyse

D'après les tableaux et figures exposés dans les parties précédentes on déduit les faits suivants :

- l'approche supervisé (réseau de neurones) a obtenu un taux de réussite meilleur que les deux approches non-supervisés (Hybridation LDA_LSI et K-means) et cela sur les différents dataset.
- Un nombre non équilibré de tweets dans chaque classe rend la tâche de segmentation très difficile. Le réseau neuronal a surmonté cette difficulté lors de la classification car on a un processus d'apprentissage supervisé derrière.
- Le choix des paramètres joue un rôle très important lors de l'entrainement d'un modèle pour qu'il puisse donner de bon résultats lors de la classification de nouvelles données.
- Le paramétrage change en fonction du type de données traiter et leur densités.
- La phase de pré-traitement a un impact très important sur la qualité du résultat. Un pré-traitement judicieux facilitera la tâche de segmentation/classification aux algorithmes d'apprentissage automatique (non-supervisés et supervisés) et vis-vers-ça.
- Pour l'apprentissage supervisé, il faut s'assurer que les données sont correctement étiquetées pour pouvoir fournir en sortie des résultats cohérents.
- Le temps nécessaire pour entrainer un modèle varie selon l'ampleur des données à traiter.
- Pour le temps d'exécution à savoir le temps de clustering/classification était quant à lui rapide, et cela à cause de l'utilisation des modèles déjà entrainés.

La Figure 4.37 ci-dessous est une représentation des 13 topics de la dataset *Emotions_v1* avec leur top termes.

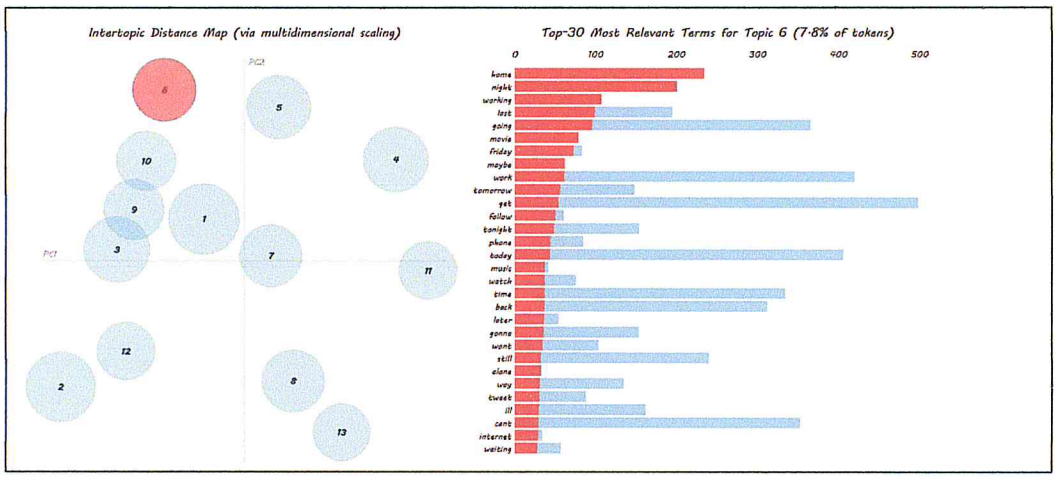


FIGURE 4.37 – Fréquence des 30 top termes dans le 6ème topic.

Conclusion

Afin de valider les approches proposées pour la fouille de tweets, plusieurs tests expérimentaux ont été effectué sur différents datasets.

Pour l'approche non-supervisée, l'algorithme hybridant LDA_LSI a surclassé K-means sur tous les tests sauf celui sur le dataset *Coachella* où il a été très influencé par le nombre non équilibré de la classe "positive".
En ce qui concerne l'approche supervisée, plus exactement, avec le réseau neuronal à une seule couche cachée, nous avons prouvé qu'en effet nous pouvons atteindre un taux de réussite très signifiant voir à 78.81%.

CONCLUSION ET FUTURE PERSPECTIVES

A travers cette étude, on a pu réaliser un système de segmentation/classification des tweets selon les thèmes et/ou les centres d'intérêts.
Pour cette réalisation, on est passé par plusieurs phases, la phase de pré-traitement, suivie par la conversion du texte vers des vecteurs/matrices numériques, en fonctions de plusieurs mesures de similarités et au final l'utilisation des algorithmes d'apprentissage automatique.
Avec les modèles proposés on a pu comparer les performances, et atteindre des résultats acceptables et cela en les appliquant sur différents dataset.

Le meilleur taux de réussite a été atteint par le modèle neuronal et cela sur les différents corpus de test, ce qui nous pousse à dire que l'approche supervisée a surpassé les approches non-supervisées. En revanche ces méthodes supervisées ne nous apprennent pas de nouvelles connaissances sur les données traitées, contrairement aux approches non-supervisées qui nous permettent de découvrir de nouvelles relations entre les données (comme vu avec la méthode du *topic modeling*).

Comme future perspectives, nous prévoyons le passage vers des approches de clustering/classification en ligne. Ceci serai une application très intéressante pour détecter des menaces, ou des catastrophes en temps réel. Ce projet pourrait se réaliser avec une hybridation avec des méta-heuristiques pour fixer le bon choix des paramètres représentatives.

Une autre perspective forte intéressante, et de proposer une version parallèle aux modèles proposés, afin d'accélérer la phase d'apprentissage, qui nous a pris ici un temps considérable, surtout pour les large corpus de tweets.

BIBLIOGRAPHIE

[Anumol and P., 2016] Anumol, B. and P., R. V. (2016). Efficient density based clustering of tweets and sentimental analysis based on segmentation. *International Journal of Computer Techniques*, 3 :53–57.

[Azencott, 2018] Azencott, C.-A. (2018). Un cadre de validation croisée. https ://openclassrooms.com/courses/evaluez-et-ameliorez-les-performances-d-un-modele-de-machine-learning/mettez-en-place-un-cadre-de-validation-croisee.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Blei et al., 2003] Blei, D. M., Ng, Y., A., Jordan, and I, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3 :993–1022.

[Boom et al., 2015] Boom, D., C., V. C., S., and Dhoedt, B. S.-D. E. C. I. T. F. (2015). *Making Sense Of Microposts*. Ceur.

[CHEN et al., 2015] CHEN, S. H., SANTOSO, A., and LEE, Y. S. (2015). *and WANG.* J. -C. Latent dirichlet allocation based blog analysis for criminal intention detection system. Security Technology (ICCST), 2015 International Carnahan Conference on, 2015. IEEE.

[Chrislb, 2005] Chrislb (2005). Structure d'un neurone artificiel. `https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_francais.png?uselang=fr`.

[Crowd.Flower, 2018] Crowd.Flower, F. (2018). Open datasets. `https://www.figure-eight.com/data-for-everyone/`.

[DeMello. et al., 2005] DeMello., R. F., J., S. L., and T., Y. L. (2005). Automatic text classification using an artificial neural network. *In : Ng M. K., Doncescu A., Yang L. T., Leng T. (eds) High Performance Computational Science and Engineering. IFIP — The International Federation for Information Processing, vol*, 172.

[Ester and P., 1996] Ester, M. and P., H. (1996). *Kriegel.* J.

[Fodeh et al., 2009] Fodeh, S. J., Punch, W. F., and Tan, P.-N. (2009). Combining statistics and semantics via ensemble model for document clustering. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, pages 1446–1450, New York, NY, USA. ACM.

[Forgy, 1965] Forgy, E. W. (1965). *Cluster analysis of multivariate data : efficiency versus interpretability of classifications.* Biometrics. 21 :768–769.

[Friedemann, 2015] Friedemann, V. (2015). *Clustering A Customer Base Using Twitter Data.*

[Gao et al., 2014] Gao, D., Li, W., Cai, X., Zhang, R., and Ouyang, Y. (2014). *Sequential summarization : A full view of twitter trending topics.*

[GERBER, 2014] GERBER, M. S. (2014). Predicting crime using Twitter and kernel density estimation. *Decision Support Systems*, 61 :115–125.

[Giller, 2012] Giller, G. L. (2012). *The Statistical Properties of Random Bitstreams and the Sampling Distribution of Cosine Similarity.*

[Glosser.ca, 2013] Glosser.ca (2013). Artificial neural network with layer coloring. `https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg`.

[Gurney, 2003] Gurney, K. (2003). An introduction to neural networks. ROUTLEDGE.

[Han et al., 2015] Han, Y., Lee, H., and Kim, Y. (2015). A real-time knowledge extracting system from social big data using distributed architecture. In *Proceedings of the 2015 Conference on research in adaptive and convergent systems - RACS*, page 74–79.

[H.Benhabiles, 2017] H.Benhabiles (2017). Cours master 1 -systèmes multimédia et la rdf-.

[HRcommons, 2009] HRcommons (2009). Réseau de neurones formels de type perceptron multicouche. `https://commons.wikimedia.org/wiki/File:Perceptron_4layers.png?uselang=fr`.

[Ifrim et al., 2014] Ifrim, G., Shi, B., and Brigadir, I. (2014). Event detection in twitter using aggressive filtering and hierarchical tweet clustering. *Second Workshop on Social News on the Web (Snow), Seoul, Korea*, 8.

[Jelodar et al., 2017] Jelodar, H., Wang, Y., Yuan, C., and Feng, X. (2017). Latent dirichlet allocation (LDA) and topic modeling : models, applications, a survey. *CoRR*, abs/1711.04305.

[Jones and K., 1972] Jones, S. and K. (1972). *A Statistical Interpretation of Term Specificity and Its Application in Retrieval.* Journal of Documentation. 28 :11–21.

[Kathy et al., 2011] Kathy, L., Palsetia, D., Narayanan, R., and Md (2011). Mostofa ali patwary, ankit agrawal, and alok n. *Choudhary. Twitter trending topic classification. In Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, December*, 11(23) :251–258.

[Kaufman and Rousseeuw, 1987] Kaufman, L. and Rousseeuw, P. J. (1987). *Clustering by means of Medoids in Statistical Data Analysis Based on the Norm and Related Methods.* edited by Y.

[K.Smith, 2017] K.Smith (17 December 2017). Twitter statistics. `https://www.brandwatch.com/blog/44-twitter-stats/`.

[Kumar et al., 2014] Kumar, S., Morstatter, F., and Liu, H. (2014). *Twitter Data Analytics.* Springer.

[MIKOLOV et al., 2013] MIKOLOV, T., K., C., G., C., and J., D. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301*, 3781.

[MILLER, 1995] MILLER, G. A. (1995). *Wordnet : a lexical database for english.* Commun.

[Mitchell,] Mitchell, T. M. Machine learning. *WCB–McGraw–Hill. ISBN 0-07-042807-7.. In particular see.*

[MmeFareh, 2017] MmeFareh (2017). Cours master 1 - matser 2.

[Nair and Hinton,] Nair, V. and Hinton, G. E.

[NAVIGLI, 2009] NAVIGLI, R. (2009). Word sense disambiguation : A survey. *ACM Comput. Surv.*, 41(2) :10.

[Ozdikis et al., 2012] Ozdikis, O., Senkul, P., and Oguztuzun, H. (2012). (2012). *Semantic Expansion of Tweet Contents for Enhanced Event Detection in Twitter. In*, 2012 :20–24.

[Popovici et al., 2014] Popovici, R., Weiler, A., and Grossniklaus, M. (2014). *On-line Clustering for Real-Time Topic Detection in Social Media Streaming Data.*

[Purwitasari et al., 2015] Purwitasari, D., Fatichah, C., Arieshanti, I., and Hayatin, N. (2015). *k-Medoids Algorithm on Indonesian Twitter Feeds for Clustering Trending Issue as Important Terms in News Summarization.* Information and Communication Technology And Systems (ICTS).

[REN et al., 2016] REN, Y., WANG, R., and JI, D. (2016). A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences*, 369 :188–198.

[SHARMA et al., 2015] SHARMA, V., KULSHRESHTHA, R., SINGH, P., AGRAWAL, N., and KUMAR, A. (2015). *Analyzing Newspaper Crime Reports for Identification of Safe Transit Paths.* HLTNAACL.

[Siwei et al., 2015] Siwei, L., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *In Proc. Conference of the Association for the Advancement of Artificial Intelligence (AAAI.*

[Soni and Mathai, 2015] Soni, R. and Mathai, K. J. (2015). Improved twitter sentiment prediction through cluster-then-predict model. *Arxiv Preprint Arxiv :1509*, page 02437.

[Székely and Rizzo, 2005] Székely, G. J. and Rizzo, M. L. (2005). Hierarchical clustering via Joint Between-Within Distances : Extending Ward's Minimum Variance Method. *Journal of Classification*, 22 :151–183.

[TAN et al., 2014] TAN, S., LI, Y., SUN, H., GUAN, Z., YAN, X., BU, J., CHEN, C., and HE, X. (2014). Interpreting the public sentiment variations on twitter. *IEEE transactions on knowledge and data engineering*, 26 :1158–1170.

[Twitter, 2018] Twitter (2018). Twitter statistics. `https://about.twitter.com/fr/company.html`.

[Wang. et al., 2015] Wang., S., W., W., Y., Z., and X., F. (2015). An ontology evolution method based on folksonomy. In *Journal of Applied Research and Technology.*

[WANG et al., 2012] WANG, X., GERBER, M. S., and BROWN, D. E. (2012). Automatic crime prediction using events extracted from twitter posts. *SBP*, 12 :231–238.

[WANG et al., 2016] WANG, Y., LUO, J., NIEMI, R., LI, Y., and HU, T. (2016). *Catching Fire via Likes : Inferring Topic Preferences of Trump Followers on Twitter.* ICWSM.

[Wu and Palmer, 1994] Wu, Z. e. M. and Palmer (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics, ACL '94*, page 133–138.

[Yoon, 2014] Yoon, K. (2014). (2014). *Convolutional neural networks for sentence classification. arXiv preprint arXiv :1408*, 5882.

[Yuhua Li and Crockett, 2006] Yuhua Li, David McLean, Z. A. B. J. D. O. and Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics.