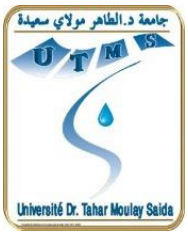
الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Dr. Tahar Moulay Saida جامعة د. الطاهر مولاي
Faculté de la Technologie كلية التكنولوجيا
Département d'Informatique قسم الإعلام الآلي



MEMOIRE DE MASTER

Option : Modélisation Informatique des Connaissances et du Raisonnement

Thème :

Fouille de données massives : Analyse des Sentiments dans un environnement Big Data

Présenté par :
Mr. SOUIDI Mohamed Abdou

Encadré par :
Mr. AMINE Abdelmalek

Juin 2016

A ma mère

A mon père

A mes frères et ma sœur

A toutes les personnes qui m'aiment

Qu'ils trouvent ici l'expression de ma sincère gratitude

Remerciements

Aucune œuvre humaine ne peut se réaliser sans l'aide d'Allah. Je le remercie en premier lieu de m'avoir donné la santé, le courage ainsi qu'une grande volonté pour aboutir à ce travail.

J'adresse tout d'abord, ma profonde reconnaissance et mes vifs remerciements à M. Abdelmalek AMINE pour avoir accepté de diriger ce mémoire. Je le remercie pour son soutien et la patience dont il a fait preuve à mon égard durent ce mémoire.

J'aimerais ensuite remercier tous les professeurs, amis et collègues, agents administratifs que j'ai côtoyé durant mes années à l'UMTS. Ils m'ont aidé et ont tout simplement rendu mes années à l'université agréables.

Finalement je remercie chaleureusement, tous les membres de ma famille pour la confiance qu'ils m'accordent, leur amour, et leurs encouragements et surtout mon très cher père Abdelkader, qui m'a soutenu et encouragé pendant ces années.

ملخص

لقد ساهم التطور والانتشار الواسع للإنترنت في تغيير كيفية اكل الاغذية وتفاعل الناس معها. يعتبر موقع أمازون واحد من أكبر المواقع التي يستطيع المستعملين شراء الأغذية المحتاجين إليها منه بكل سهولة، فكل ما يتطلب هو نقرة فأرة واحدة للشراء منه. معظم المنتوجات المعروضة على الموقع يوجد لها تقييم من 5 درجات ورأي للمستخدمين الاخرين الذين ينتمون لمناطق، ثقافات وعادات غذائية مختلفة.

في هذا العمل، نريد عمل تحليل اوتوماتيكي للمشاعر (الآراء) المعبر عنها من طرف المستخدمين، باستخدام بيئة عمل البيانات الضخمة، ونقترح لذلك دراسة مقارنتيه لثلاثة خوارزميات باستخدام عدة طرق تمثيل للبيانات وطرق معالجة أولية للنصوص مختلفة لكي نستطيع عمل هذا التحليل. ونعرض بعد ذلك النتائج المحصل عليها من الخوارزميات الثلاث ونقدم على عمل مقارنة بين هذه النتائج من اجل توضيح إيجابيات وسلبيات كل طريقة.

الكلمات المفتاحية: الأغذية، امازون، رأي، المستخدمين، البيانات الضحمة، معالجة أولية للنصوص، تمثيل البيانات، تحليل أتوماتيكي للمشاعر.

Abstract

The development of the Internet has changed the way people eat and respond for food. Amazon is one of the biggest websites where users can easily purchase all kinds of foods they need with only a mouse-click. Most of foods have ratings and reviews from other customers who have different emplacements, cultures and eating habits.

In this work, we want to perform an automatic sentiment analysis based on the reviews of customers in a BIG DATA environment and for that, we propose a comparative study between three algorithms with different representation and pretreatment methods, to realize this analysis. We present, thereafter, the results obtained by the three algorithms and we proceed to a comparative study between the obtained results in order to highlight the advantages and the limits of each method.

Keywords: Food, Amazon, reviews, automatic sentiment analysis, big data, representation methods, text pretreatment.

Résumé

Le développement de l'Internet a changé la façon dont les gens mangent et réagissent à la nourriture. Amazon est l'un des plus grands sites web où les utilisateurs peuvent facilement acheter tout type de nourriture dont ils ont besoin, avec seulement un clic de souris. La plupart des aliments ont une note sur 5 et des avis des autres utilisateurs qui ont différents emplacements, cultures, habitudes alimentaires, etc.

Dans ce travail, nous voulons faire une analyse automatique des sentiments exprimés dans les avis de ces utilisateurs, dans un environnement BIG DATA, et pour cela nous proposons une étude comparative entre trois algorithmes avec différentes méthodes de représentation et de prétraitement des données afin de réaliser cette analyse. Nous présentons, par la suite, les résultats obtenus par les trois algorithmes et nous procédons à une étude comparative entre les résultats obtenus afin de souligner les avantages et les limites de chaque méthode.

Mots clés : Nourriture, Amazon, avis des utilisateurs, analyse automatique des sentiments, big data, méthodes de représentations, prétraitement des textes, données massives.

Table des matières

Liste des figures

Liste des figures

Liste des tableaux

1 Introduction

1.1 Présentation du sujet

De nos jours, l'internet est un outil incontournable d'échange d'information tant au niveau personnel que professionnel. Le Web nous offre un monde très riche d'information et a évolué des simples ensembles de pages statiques vers des services de plus en plus complexes. Ces services nous offrent l'achat en ligne des produits, la lecture des livres en ligne, la discussion sur de multiples forums ou la possibilité de s'exprimer sur les réseaux sociaux.

L'internet contient un nombre énorme d'informations, et pour la plupart d'entre nous c'est le premier lieu pour trouver ces informations, acheter des produits, réserver l'avion ou l'hôtel, consulter les avis d'autres utilisateurs sur les produits qui nous intéressent. Amazon est un des sites les plus visités sur le web, ce site nous offre l'achat en ligne de tous les produits dont la nourriture. La plupart des produits alimentaires sur ce site ont une note sur 5 et des commentaires (avis) des clients. Le nombre et la taille de ces derniers se multiplient au cours du temps, ce qui rend l'analyse de ces données, par les méthodes classiques, plus difficile. Pour cela, dans ce mémoire, nous avons utilisé les techniques du BIG DATA afin de résoudre ce problème.

Le domaine de recherche présenté dans ce mémoire est l'Analyse des Sentiments. Le but général est d'effectuer une étude comparative entre trois algorithmes, avec différentes méthodes de représentation et de prétraitement, appliqués sur un corpus d'Amazon.

Dans la littérature, L'analyse des sentiments (parfois appelée opinion mining) est la partie du text mining qui essaye de définir les opinions, sentiments et attitudes présentent dans un texte ou un ensemble de textes. Développée essentiellement depuis les années 2000, elle est particulièrement utilisée en marketing pour analyser par exemple les commentaires des internautes.

L'objectif de ce mémoire est d'extraire les sentiments à partir d'un corpus gigantesque d'Amazon appelé « Amazon fine foods » en utilisant "Apache Spark" qui est un framework (outil) de traitements Big Data open source construit pour effectuer des analyses sophistiquées et conçu pour la rapidité et la facilité d'utilisation, afin d'effectuer une étude comparative entre trois méthodes de classification avec apprentissage supervisé: régression logistique, naïve bayes et "support vector machine" SVM. Nous avons appliqué pour chacune de ces méthodes deux modèles de représentations sac de mot et n-gram,

avec trois techniques de prétraitement des textes: tokenisation, racinisation (stemming) et lemmatisation.

1.2 Organisation du mémoire

Ce mémoire comprend cinq chapitres au total. Au prochain chapitre (Chapitre 2) nous présenterons une étude théorique des données massives (BIG DATA) et ces divers outils surtout le framework « Apache Spark ».

Au Chapitre 3 nous décrirons les techniques de la représentation du corpus documentaire et donc le prétraitement, l'indexation et la vectorisation du corpus. Nous finirons ce chapitre en montrent les techniques de pondérations.

Au Chapitre 4 nous montrerons les différentes approches existantes de l'Analyse des sentiments et leur utilisation. Et nous expliquerons les raisons pour lesquelles l'analyse de l'opinion (sentiments) est si complexe.

Au Chapitre 5 nous décrirons le corpus utilisé, et l'implémentation des classifieurs, méthodes de représentations, techniques de prétraitement. Ensuite, nous présenterons les tests sur chaque classificateur de la notation de l'opinion, nous présenterons les résultats obtenus sur chaque classification sur la même base d'apprentissage. Enfin, nous comparerons ces classificateurs en mettant en évidence les inconvénients et les avantages de chacun.

Enfin, il vient le chapitre de conclusion et perspectives.

Chapitre 2

Les données massives (Big Data)

2.1 Introduction

Les techniques de représentation de l'information se varient au cours des âges, tant au niveau des codes qu'au niveau des supports. Écriture cunéiforme, hiéroglyphes, calligraphies, idéogrammes et alphabets divers et variés ont permis à travers les siècles de représenter l'information. En ce qui concerne les supports de stockage, nous sommes passés de l'argile, la pierre, le bois, le cuir, le métal, ou le parchemin, au silicium aujourd'hui. Mais le but n'a pas changé depuis la nuit des temps. Stocker toujours plus d'information, la reproduire et la diffuser toujours plus vite pour transmettre les idées, l'expérience et le savoir, afin de permettre à chacun, des simples individus aux grands de ce monde, de s'informer pour décider « en connaissance de cause ».

Fin 2012, certains experts annonçaient qu'entre 2010 et 2012, l'humanité a généré autant d'information que tout au long de son histoire passée, et ce volume sera amené à doubler à nouveau tous les deux ans tout au long de la décennie. En réalité, il y a eu entre-temps un changement de paradigme. Les données ne sont plus stockées sous forme scripturale mais digitale. Nos données, quelles qu'elles soient sont stockées sous forme de 0 et de 1 sur des supports magnétiques, ce qui a accéléré leur création, duplication et transmission.

À l'instar de l'évolution industrielle, l'évolution de l'ère digitale repose sur la capacité à toujours mieux exploiter les informations existantes et à exploiter de nouvelles sources d'information. C'est ce que l'on appelle généralement le big data. Nous reviendrons sur des définitions plus précises du big data mais on peut dire, en première approche, que le big data recouvre l'ensemble des technologies, des métiers, des approches conceptuelles permettant d'exploiter l'ensemble des données générées par les hommes de façon consciente ou non et par tous les objets connectés ou non.

2.2 Les données massives (Big Data)

Dès 1944, les universitaires de Weslayan aux Etats-Unis ont anticipé l'explosion des volumes d'information pour l'année 2040. Ils ont ainsi estimé que la bibliothèque du campus contiendrait 200.000.000 livres environ, que ceux-ci devront être gérés par 6000 personnes.

Au fil du temps, les enjeux et progrès techniques liés aux traitements de stockage et d'analyse des données ont laissé entrevoir les possibilités et problématiques futures de quantités de plus en plus massives d'informations cataloguées.

Aujourd'hui 7 milliards d'êtres humains dans le monde, 2,5 milliards d'internautes dont 1,9 milliard sont présents sur les réseaux sociaux, 6,5 milliards de téléphones. 10 milliards d'objets connectés… Chaque jour, Google traite 24 pétaoctets de données. 350 millions de photos sont chargées sur Facebook, 400 millions de tweets sont envoyés ; 100 heures de vidéo sont mises en ligne chaque minute sur YouTube. Ces chiffres donnent le

vertige. Plus impressionnant encore est le rythme auquel ces chiffres croissent. [Monino & Sedkaoui, 2016]

Cette croissance, exponentielle, est liée à :

- l'évolution du nombre d'utilisateurs des solutions IT
- la génération de données par des machines et capteurs
- l'évolution des périmètres couverts et des usages (mobile, ...)
- la finesse de l'information tracée
- la croissance des volumes opérationnels
- l'évolution de l'historique de données disponible.

Bienvenue dans le monde du big data. Pour ne pas être submergé par ce déluge de données, il a fallu développer des nouveaux outils, capables de traiter plus rapidement ces données de natures très variées, pas nécessairement adaptées au mode traditionnel de structure en tables des bases de données, dispersées et parfois extrêmement volumineuses.

Le terme big data serait apparu pour la première fois en 1997 selon les archives de la bibliothèque numérique de l'ACM (Association for Computing Machinery), dans des articles scientifiques sur les défis technologiques à relever pour visualiser les « grands ensembles de données ». Toutefois, certains secteurs comme l'astronomie, la recherche sur le génome et d'autres faisaient du big data sans le savoir depuis bien longtemps. [Gil, 2013].

2.2.1 Définitions

Le concept des big data est vaste, souvent nébuleux et les définitions varient selon les personnes interrogées, leur fonction et leur utilisation des données. On va citer quelques définitions dans la suite :

" Le big data, littéralement les « grosses données », ou mégadonnées parfois appelées données massives, désignent des ensembles de données qui deviennent tellement volumineux qu'ils en deviennent difficiles à travailler avec des outils classiques de gestion de base de donées ou de gestion de l'information." Wikipedia ".

Big data est le terme qui est de plus en plus utilisé pour décrire le processus d'application de puissance de calcul importante "la plus récente dans l'apprentissage machine et de l'intelligence artificielle" a des dataset sérieusement massive et souvent très complexes." Microsoft.

"La définition de big data se réfèrent à des groupes de données qui sont si grandes et complexes que les outils de gestion de base de données régulières ont des difficultés à capturer, stocker, partager et gérer l'information." yourdictionary.com.

2.2.2 Le big data et les 5V

2.2.2.1 La définition technologique

La définition initiale donnée par le cabinet McKinsey and Company en 2011 s'orientait d'abord vers la question technologique, avec la célèbre règle des 3V : un grand Volume de données, une importante Variété de ces mêmes données et une Vitesse de traitement s'apparentant parfois à du temps réel. Ces technologies étaient censées répondre à l'explosion des données dans le paysage numérique (le « data deluge »). Puis, ces qualificatifs ont évolué, avec une vision davantage économique portée par le 4ème V de la définition, celui de Valeur, et une notion qualitative véhiculée par le 5e V, celui de Véracité des données (disposer de données fiables pour le traitement) (2).

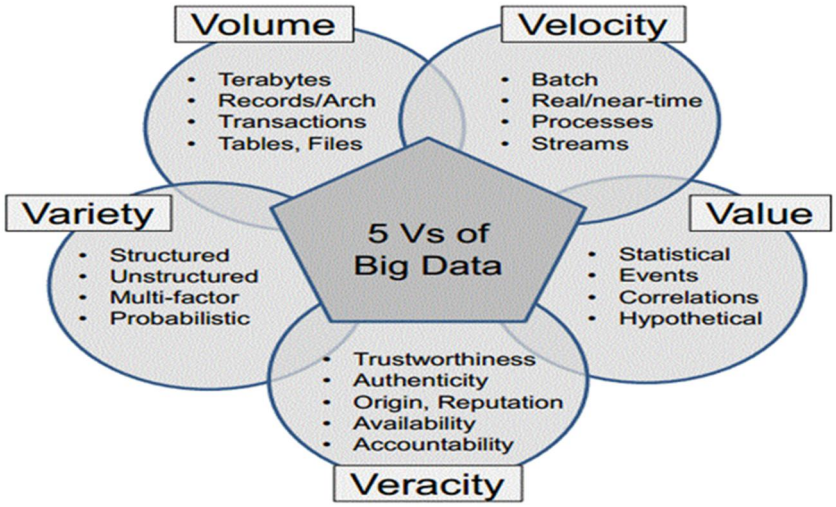


Figure 2.1: Les 5V du big data.

2.2.2.2 Le premier « V » est le volume

C'est évidemment la première caractéristique, portée par le mot « big ». Régulièrement les articles sur la question nous font réviser les échelles numériques : giga, tera, peta, exa, zetta, yotta… La multiplication des capteurs de toutes sortes, l'essor des conversations sur réseaux sociaux, la baisse des couts de stockage, le cloud sont autant de facteurs qui font croître les volumes de façon vertigineuse. Mais, indépendamment des 4 autres « V », la question du volume est purement technologique, et l'évolution constante des outils tant matériels que logiciels montre bien qu'il n'y a pas de réelle limite de ce coté là. En revanche, la limite est dans l'exhaustivité. On peut sans doute stocker et retrouver des quantités phénoménales d'information, mais l'exhaustivité est hors d'atteinte, et n'a d'ailleurs pas grand sens.

2.2.2.3 Le deuxième « V » est la vitesse

Non seulement tout stocker, mais le faire immédiatement, et surtout le rendre accessible tout aussi vite. Et on voit une demande croissante pour ce facteur vitesse. C'est évident dans des domaines comme le renseignement, la lutte contre le terrorisme, la surveillance du territoire dans le cadre de la protection civile, etc. Mais on trouve aussi cette demande de vitesse dans des domaines a priori moins habitués à réagir en temps réel, comme le marketing ou la publicité. Un exemple intéressant est celui de La Redoute qui utilise les données météo en temps réel pour choisi quelles publicités sont sélectionnées sur les tableaux d'affichage dynamiques en fonction du temps réel.

Figure 2.2: Exemple de donnée en temp réel.

Certes, les données ne sont sans doute pas trop « big » au niveau du dispositif, mais le calcul de la météo est bien un cas où la masse de données collectées, analysées et mise à disposition immédiatement répond bien à la définition de « big data ».

2.2.2.4 Le troisième « V » est la variété

Toutes sortes de types de données circulent sur le web : des données structurées, relativement faciles à prendre en compte, mais surtout une immense majorité de données non structurées, en différentes langues, niveaux de langues, dans des textes bien écrits comme des articles de journaux, des dialogues décousus sur les forums, des tweets écrits dans une langue nouvelle pleine de nouveaux mots, des enregistrements audio, des vidéos, etc. Au delà de ces différences de formes, ces données sont totalement hétérogènes : elles se rapportent à des faits, des expériences, des évènements, des opinions, des avis, des

recommandations. Autour d'un même sujet, on va ainsi trouver quantités de données et la vraie difficulté devient, non pas de les trouver, mais de les synthétiser, et le cas échéant les agréger.

La qualité des traitements est très variable selon les types de données.

2.2.2.5 Le 4eme V est la validité (ou la véracité, voire la vérifiabilité)

C'est sans doute le point le plus important. Quelle est la validité des données qui circulent ? Évidemment dans de très nombreuses applications de big data on est assuré de la fiabilité des données. Mais dès qu'on veut utiliser des données ouvertes sur le web, la difficulté survient : fausses informations, publiées de bonne foi, comme indiqué plus haut. Mais aussi de fausses informations publiées de façon plus ou moins malveillante.

Le phénomène prend de l'ampleur et c'est la « confiance numérique » qui est impactée. A tel point que des ripostes sont en préparation, comme le projet de norme de l'Afnor, qui vise à s'assurer de la fiabilité des avis de consommateurs sur Internet. Mais cela reste limité à un cas très spécifique, et le principal moyen retenu est de s'assurer de la traçabilité de l'auteur, qui devra pourvoir être retrouvé pour s'assurer qu'il a bien vécu l'expérience de consommation décrite. On voit mal comment ce procédé pourrait être étendu à une ensemble de données beaucoup plus vaste, incontrôlé, international et hétérogène.

2.2.2.6 Le 5ème V est la valeur

Les données ont une valeur intrinsèque, mais celle-ci doit être découverte. Il existe de nombreuses techniques de quantification et d'analyse qui permettent de tirer parti des données. Cela va de l'identification des préférences ou des goûts d'un utilisateur, aux offres proposées en fonction du lieu, ou encore à l'identification d'une pièce d'équipement sur le point de tomber en panne. Cette percée technologique s'est traduite par une diminution exponentielle du coût de stockage et de traitement des données, et par conséquent, par l'abondance des données disponibles pour effectuer des analyses statistiques. Elle permet également de prendre des décisions beaucoup plus exactes et précises. Toutefois, la recherche de valeur nécessite également de nouveaux processus d'identification impliquant des analystes, des utilisateurs professionnels et des dirigeants intelligents et perspicaces. Le véritable défi du Big Data est un défi humain : il s'agit d'apprendre à poser les bonnes questions, à identifier les modèles, à élaborer des hypothèses pertinentes et à prévoir les comportements.

2.3 Les outils du big data

2.3.1 Les caractéristiques majeures d'une plateforme Big Data

Des évolutions technologiques telles que la puissance des processeurs, les mémoires flash, ou l'augmentation des volumes de stockage permettent d'envisager des gains de performance importants. Pour gérer les Big Data, de nouvelles technologies Open Source et propriétaires, capables d'optimiser ces nouvelles ressources matérielles ont vu le jour. Une plateforme Big Data, dite « Big Data Analytics platform», permet de collecter et de traiter ces nouveaux ensembles de données.

La plateforme Big Data idéale devrait satisfaire aux exigences suivantes :

- elle devrait être scalable afin d'absorber un accroissement important du volume de données (de l'ordre du terabyte ou du petabyte),
- la puissance de calcul devrait être répartie sur plusieurs processeurs indépendamment de toute contrainte géographique,
- elle devrait permettre de fournir une réponse rapide aux requêtes très complexes ainsi que de supporter une grande variété de sources de données,
- elle devrait intégrer des fonctionnalités pour gérer l'apprentissage automatique, fournissant des recommandations, et exécutant ainsi des analyses sur les données entrantes, comme par exemple celles provenant de log, en temps réel
- elle devrait permettre de fournir un environnement pour permettant de gérer les requêtes pré-enregistrées,
- elle devrait être capable de gérer des données provenant d'environnements hétérogènes, tout en offrant des performances élevées pour le chargement et l'analyse,
- la capacité à gérer le fail-over, en garantissant un haut niveau de disponibilité.

2.3.2 Google « BigTable »

En 2004, Google lance à l'interne le projet « BigTable » : une plateforme « haute performance » pour le stockage et le traitement de vastes ensembles de données semi-structurées. L'application, qui repose sur une architecture distribuée (serveurs répartis en « grappes »/« clusters »), est conçue pour pouvoir répondre avec des temps de réponse très courts aux requêtes émanant simultanément de plusieurs milliers d'ordinateurs clients. BigTable est aujourd'hui l'épine dorsale de l'infrastructure Google qui l'utilise pour faire tourner la plupart de ses services en ligne : indexation, crawl, moteur de recherche, GoogleNews, GoogleAlerts, GoogleMaps, Gmail, GoogleBooks (Google a fait son entrée sur le marché de l'informatique décisionnelle alliée aux Big Data ces derniers mois. Le service BigQuery, lancé au printemps dernier aux Etats-Unis, propose aux développeurs une plateforme IaaS pour le chargement et le traitement de "données de masse". Si le moteur avait été précurseur dans le domaine du calcul distribué avec « BigTable », il n'avait pas encore capitalisé sur ces investissements pour se positionner

directement sur la BI (Business Intelligence) et peut être considéré comme un nouvel entrant sur ce secteur.

2.3.3 MapReduce

« BigTable » repose en partie sur l'utilisation de MapReduce : un formalisme pour le développement de langages de programmation et d'applications optimisées pour le traitement de « données de masse » et leur « mise à l'échelle ». Les librairies MapReduce ont été implémentées dans de très nombreux développements orientés « Big Data » par la suite, notamment Apache Hadoop.

2.3.3.1 Qu'est ce que mapReduce

MapReduce est un paradigme qui permet d'effectuer des calculs parallélisables sur un énorme volume de données (dataset stocké sur un filesystem ou dans une base de données) en utilisant un grand nombre de machines (cluster ou grid).

On définit une fonction map qui permettra de générer des tuples (clé/valeur) à partir du dataset. Ainsi qu'une fonction reduce pour déduire le résultat attendu à partir de ces tuples.

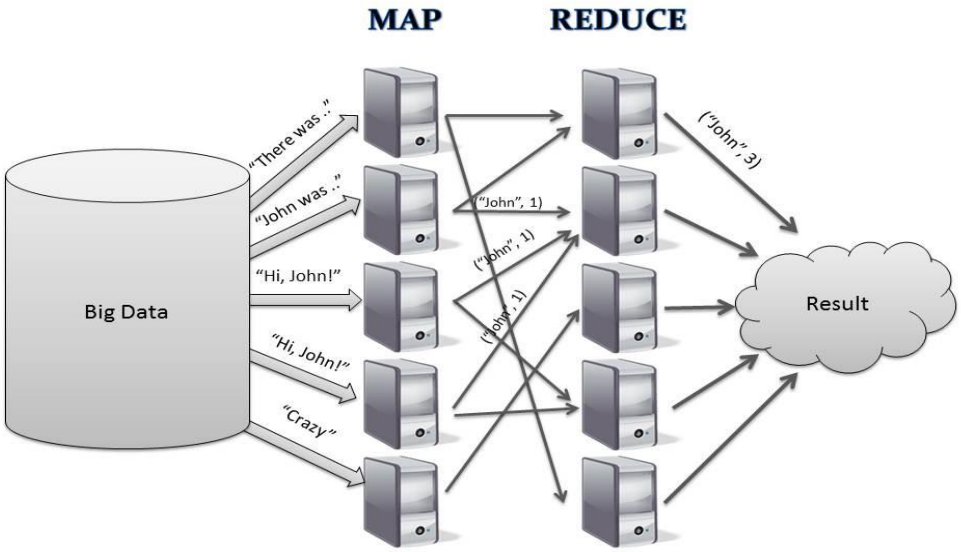


Figure 2.3: La technique MAP-REDUCE.

Donc MapReduce est une solution permettant de traiter des problématiques spécifiques nécessitant des calculs sur un gros volume de données. Le problème va être découpé en sous-problèmes dont les calculs vont être distribués sur plusieurs machines (ce qui permet une distribution du temps de traitement et des données à traiter). Les résultats seront ensuite agrégés.

2.3.4 Apache Hadoop

Créée en 2004 par Douglass Cutting pour Yahoo, Hadoop a été inspiré par la publication de MapReduce, et BigTable de Google. Apache Hadoop est un framework open-source complet écrit en java pour le Big Data, représente l'une des meilleures approches pour traiter d'importants ensembles de données variés. Le framework Hadoop fournit un modèle de programmation simple pour le traitement distribué d'ensembles de données volumineux. Il inclut le système de fichiers distribué Apache Hadoop (HDFS), une structure de planification des tâches appelée Apache Hadoop YARN et une structure de traitement parallèle appelée Apache Hadoop MapReduce. Plusieurs composants supportent l'ingestion de données (service Apache Flume), les requêtes et analyses (logiciels Apache Pig, Apache Hive et Apache HBase), la coordination des workflows (Apache Oozie), ainsi que la gestion et la surveillance du cluster de serveurs sous-jacent (logiciel Apache Ambari). Ensemble, les composants logiciels Apache créent un puissant framework pour traiter et analyser en lots des données dispersées pour des analyses historiques. [Jolia-Ferrer, 2014].

2.3.4.1 Principaux composants Hadoop

Framework de traitements parallélisés Map-Reduce

Hadoop Map-Reduce est un puissant framework Java de traitement de données massives. A noter que dans le cas de l'utilisation conjointe avec HDFS et HBase et suivant la configuration du cluster Hadoop, une partie des traitements sont effectués au niveau des noeuds de stockage.

HDFS : Hadoop Distributed File System

HDFS est un système de fichiers distribué sur des noeuds d'un cluster Hadoop. HDFS est adapté au stockage et la réplication de fichiers de grande taille (>256MB).

A noter qu'il existe plusieurs formats de stockage des données dans HDFS dont certains en colonne comme ORC, Parquet, ...

2.3.5 Bases de données NoSQL(Not Only SQL)

Ces bases de données non relationnelles, qui se présentent sous la forme de quatre types de magasins différents (clé-valeur, colonne, graphe ou document) fournissent un stockage hautes performances et haute disponibilité à l'échelle du Web. Elles permettent de traiter des flux de données massifs, ainsi que des types de schémas et de données flexibles, avec des temps de réponse rapides. Les bases de données NoSQL utilisent une architecture distribuée tolérante aux pannes, qui garantit la fiabilité et l'évolutivité du système. Apache HBase, Apache Cassandra*, MongoDB* et Apache CouchDB* sont des exemples de bases de données NoSQL.[Beranger, 2015].

2.3.5.1 Bases de données analytiques Clé/Valeur

Dans ce modèle, chaque objet/enregistrement est identifié par une clé unique. La structure de l'objet est libre.

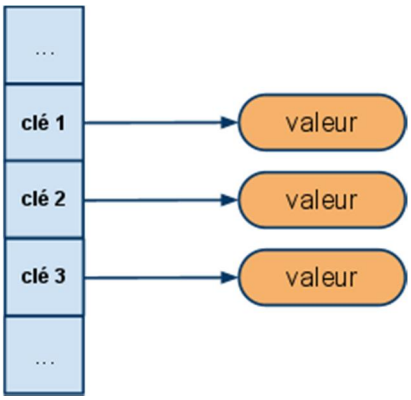


Figure 2.4 Base de données analytique Clé-Valeur

Dans ce modèle on ne dispose généralement que des quatre opérations Create, Read, Update, Delete (CRUD) en utilisant la clé de l'enregistrement à manipuler.

Du fait des limites fonctionnelles d'accès aux données de ces types de base, nous ne leur voyons pas d'application décisionnelle.

2.3.5.2 Bases de données analytiques Colonnes

Ces bases de données basées sur des grilles stockent les données dans des colonnes (pas dans des lignes), réduisant le nombre d'éléments à lire pendant le traitement des requêtes et accélérant l'exécution de nombreuses requêtes simultanées. Il s'agit d'environnements en lecture seule qui améliorent le prix, les performances et l'évolutivité des SGBDR conventionnels. Elles conviennent aux EDW et autres applications impliquant beaucoup de requêtes, et sont optimisées pour le stockage et la récupération d'analyses avancées. Les plates-formes analytiques SAP* Sybase* IQ, ParAccel* et HP* Vertica* s'appuient sur des bases de données colonnes.

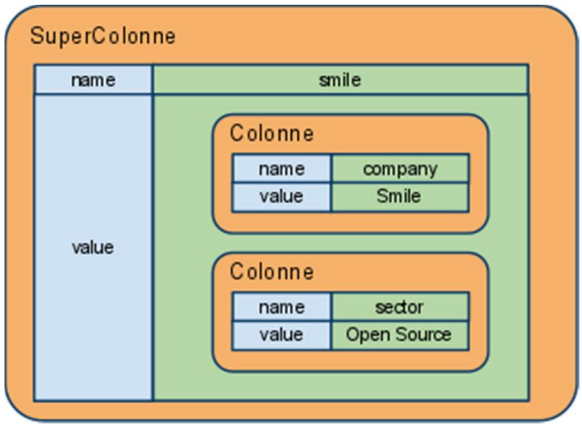


Figure 2.5 Base de données analytique colonne

La colonne représente l'entité de base de la structure de données. Chaque colonne d'un objet est défini par un couple clé / valeur. Une colonne contenant d'autres colonnes est nommée super-colonne.

Ces types de bases sont adaptés au stockage opérationnel de masse (ODS) et de source d'analyses massives dans un système décisionnel.

2.3.5.3 Bases de données et outils analytiques Graphes

Les bases de données graphes représentent un type de base de données NoSQL qui gagne en importance. Elles sont particulièrement utiles pour les données hautement connectées dans lesquelles les relations sont plus nombreuses ou plus importantes que les entités individuelles. Les graphes sont des structures flexibles qui facilitent la connexion et la modélisation des données. Ils sont plus rapides à interroger, plus intuitifs à modéliser et visualiser. La croissance des Big Data est par nature en grande partie imputable aux graphes. Les bases de données graphes fonctionnent seules ou avec d'autres outils (tels que : visualisation, analyse de graphes, apprentissage automatique). Par exemple, avec l'apprentissage automatique (machine learning), elles servent à explorer et prévoir des relations pour résoudre une série de problèmes. [Bruchez, 2015].

2.3.5.4 Bases de données analytiques Documentaires

Les bases de données documentaires sont constituées de collections de documents. Les collections sont généralement assimilées à des tables d'un modèle relationnel.

Bien que les documents soient structurés, ces bases sont sans schéma de données prédéfini. Il n'est donc pas nécessaire de définir au préalable l'ensemble des champs utilisés dans un document. Les documents peuvent donc avoir une structure hétérogène au sein de la base.

Un document est composé de champs et de valeurs associées, ces dernières pouvant être requêtées. Les valeurs peuvent être, soit d'un type simple (entier, chaîne de caractère, date, ...), soit composées de plusieurs couples clé/valeur (imbrications nested sets). Les structures de données sont donc très souples.

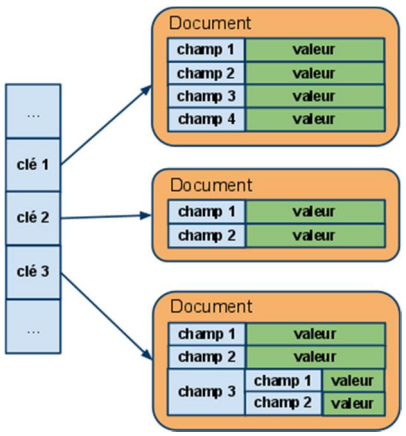


Figure 2.6: Base de données analytique documentaire.

La souplesse du modèle de données, les performances et les capacités de requêtage orientent l'usage des bases documentaires vers du stockage opérationnel de masse (ODS) dans un système décisionnel.

2.3.6 Apache Spark

2.3.6.1 Qu'est-ce que Spark

Apache Spark est un framework de traitements Big Data open source construit pour effectuer des analyses sophistiquées et conçu pour la rapidité et la facilité d'utilisation. Celui-ci a originellement été développé par AMPLab, de l'Université UC Berkeley, en 2009 et passé open source sous forme de projet Apache en 2010.

Spark présente plusieurs avantages par rapport aux autres technologies big data et MapReduce comme Hadoop et Storm. D'abord, Spark propose un framework complet et unifié pour répondre aux besoins de traitements Big Data pour divers jeux de données, divers par leur nature (texte, graphe, etc.) aussi bien que par le type de source (batch ou flux temps-réel). Ensuite, Spark permet à des applications sur clusters Hadoop d'être exécutées jusqu'à 100 fois plus vite en mémoire, 10 fois plus vite sur disque. Il vous permet d'écrire rapidement des applications en Java, Scala ou Python et inclut un jeu de plus de 80 opérateurs haut-niveau. De plus, il est possible de l'utiliser de façon interactive pour requêter les données depuis un shell.

Enfin, en plus des opérations de Map et Reduce, Spark supporte les requêtes SQL et le streaming de données et propose des fonctionnalités de machine learning et de traitements orientés graphe. Les développeurs peuvent utiliser ces possibilités en stand-alone ou en les combinant en une chaîne de traitement complexe.

2.3.6.2 Les fonctionnalités de Spark

Spark apporte des améliorations à MapReduce grâce à des étapes de shuffle moins coûteuses. Avec le stockage en mémoire et un traitement proche du temps-réel, la performance peut être plusieurs fois plus rapide que d'autres technologies big data. Spark supporte également les évaluations paresseuses ("lazy evaluation") des requêtes, ce qui aide à l'optimisation des étapes de traitement. Il propose une API de haut-niveau pour une meilleure productivité et un modèle d'architecture cohérent pour les solutions big data.

Spark maintient les résultats intermédiaires en mémoire plutôt que sur disque, ce qui est très utile en particulier lorsqu'il est nécessaire de travailler à plusieurs reprises sur le même jeu de données. Le moteur d'exécution est conçu pour travailler aussi bien en mémoire que sur disque. Les opérateurs réalisent des opérations externes lorsque la donnée ne tient pas en mémoire, ce qui permet de traiter des jeux de données plus volumineux que la mémoire agrégée d'un cluster. Spark essaye de stocker le plus possible en mémoire avant de basculer sur disque. Il est capable de travailler avec une partie des données en mémoire, une autre sur disque.

Il est nécessaire d'examiner ses données et ses cas d'utilisation pour évaluer ses besoins en mémoire car, en fonction du travail fait en mémoire, Spark peut présenter d'importants avantages de performance. Les autres fonctionnalités proposées par Spark comprennent :

- Des fonctions autres que Map et Reduce
- L'optimisation de graphes d'opérateurs arbitraires
- L'évaluation paresseuse des requêtes, ce qui aide à optimiser le workflow global de traitement
- Des APIs concises et cohérentes en Scala, Java et Python
- Un shell interactif pour Scala et Python (non disponible encore en Java)

Spark est écrit en Scala et s'exécute sur la machine virtuelle Java (JVM). Les langages supportés actuellement pour le développement d'applications sont :

- Scala
- Java
- Python
- Clojure
- R

2.3.6.3 L'écosystème de Spark

À côté des API principales de Spark, l'écosystème contient des librairies additionnelles qui permettent de travailler dans le domaine des analyses big data et du machine learning. Parmi ces librairies, on trouve :

- Spark Streaming : Spark Streaming peut être utilisé pour traitement temps-réel des données en flux. Il s'appuie sur un mode de traitement en "micro batch" et utilise pour les données temps-réel DStream, c'est-à-dire une série de RDD (Resilient Distributed Dataset).
- Spark SQL : Spark SQL permet d'exposer les jeux de données Spark via API JDBC et d'exécuter des requêtes de type SQL en utilisant les outils BI et de visualisation traditionnels. Spark SQL permet d'extraire, transformer et charger des données sous différents formats (JSON, Parquet, base de données) et les exposer pour des requêtes ad-hoc.
- Spark MLlib : MLlib est une librarie de machine learning qui contient tous les algorithmes et utilitaires d'apprentissage classiques, comme la classification, la régression, le clustering, le filtrage collaboratif, la réduction de dimensions, en plus des primitives d'optimisation sous-jacentes.
- Spark GraphX : GraphX est la nouvelle API (en version alpha) pour les traitements de graphes et de parallélisation de graphes. GraphX étend les RDD de Spark en introduisant le Resilient Distributed Dataset Graph, un multi-graphe orienté avec des propriétés attachées aux nœuds et aux arrêtes. Pour le support de ces traitements, GraphX expose un jeu d'opérateurs de base (par exemple subgraph, joinVertices,

aggregateMessages), ainsi qu'une variante optimisée de l'API Pregel. De plus, GraphX inclut une collection toujours plus importante d'algorithmes et de builders pour simplifier les tâches d'analyse de graphes.

Le diagramme suivant montre les relations entre ces différentes librairies de l'écosystème.

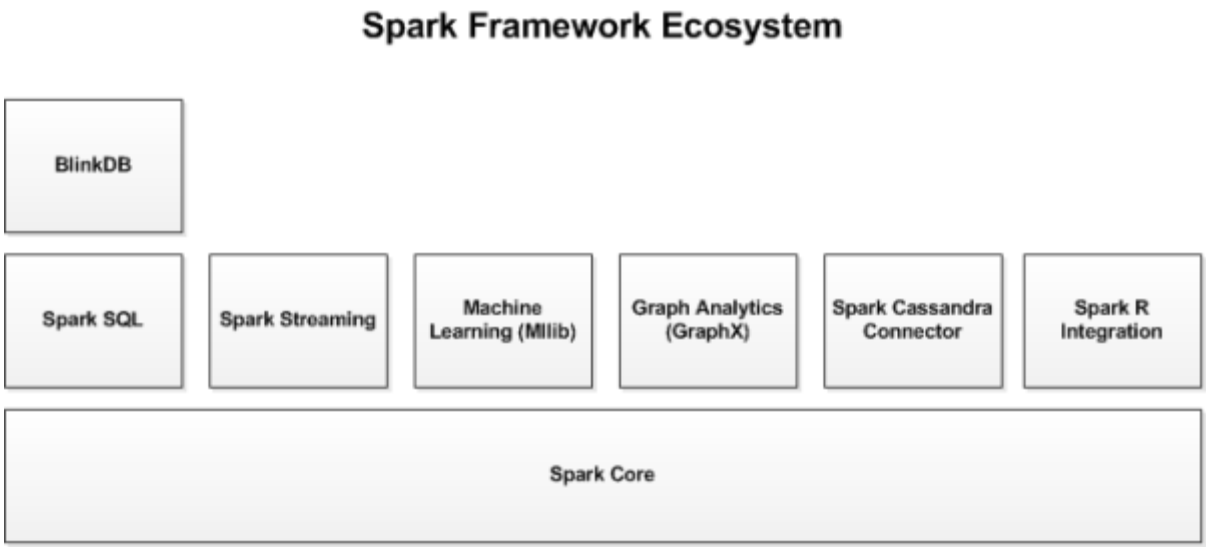


Figure 2.7 Spark Framework Libraries

2.3.6.4 L'architecture de Spark

L'architecture de Spark comprend les trois composants principaux suivants :

- Le stockage des données
- L'API
- Le Framework de gestion des ressources

Regardons chacun de ces composants plus en détails

Le stockage des données :

Spark utilise le système de fichiers HDFS pour le stockage des données. Il peut fonctionner avec n'importe quelle source de données compatible avec Hadoop, dont HDFS, HBase, Cassandra, etc.

L'API :

L'API permet aux développeurs de créer des applications Spark en utilisant une API standard. L'API existe en Scala, Java et Python. Les liens ci-dessous pointent vers les sites présentant les API Spark pour chacun de ces langages :

- Scala API
- Java
- Python

Gestion des ressources

Spark peut être déployé comme un serveur autonome ou sur un framework de traitements distribués comme Mesos ou YARN. La figure 2 illustre les composants du modèle d'architecture de Spark.

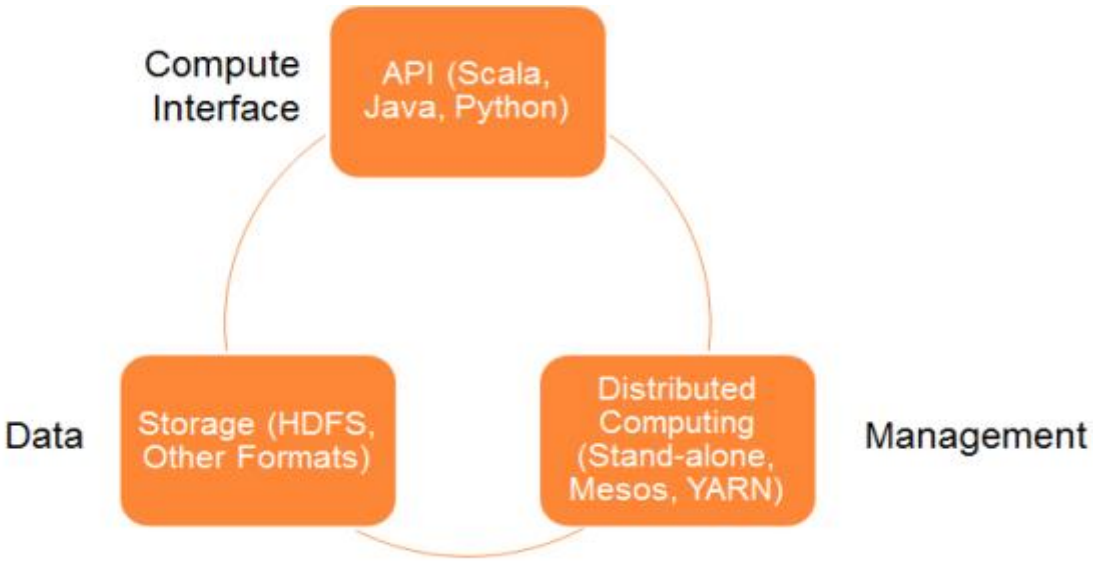


Figure 2.8: L'architecture du spark.

Chapitre 3

Représentation des textes

3 Représentation des textes

3.1 Introduction

L'explosion de la quantité d'informations textuelles provoquée par l'évolution à grande échelle des outils de communication essentiellement Internet qui est sorti de l'aspect réservé à un milieu restreint à un aspect de vulgarisation au grand public, a rapidement, fait sentir le besoin de recherche de mécanismes et outils de traitement automatique des quantités d'informations diffusées sur le Web. Ainsi, avec les bases de données multimédia, les dépêches d'agences de presse, les publications scientifiques, les bibliothèques électroniques, etc. Qui sont consultés habituellement sur le réseau, on dispose de plus en plus de grandes masses de documents non ou faiblement structurées, en particulier les documents textuels qui sont considérés comme étant des documents non structurés, surnommés « documents plats » par quelques auteurs, c'est-à-dire comme une séquence ou un ensemble de mots sans informations complémentaires sur le document. Différents formats HTML, SGML, XML, DOC, PDF, … peuvent être des moyens pour stocker et représenter ces documents. Le manque de structure au sein de ces collections volumineuses rend difficile l'accès à l'information qu'elles contiennent, d'où la nécessité aujourd'hui, de chercher comment structurer automatiquement ces corpus pour les rendre utilisables d'une façon rapide et optimale pour y faciliter leurs traitements automatiques et notamment la classification. Pour pouvoir y appliquer les différentes techniques et algorithmes d'apprentissage, une transformation de ces documents non ou peu structurés est indispensable. La transformation ou le codage de ces documents est une préparation à « l'informatisation » de ces derniers, chaque type de documents comme les images, les vidéos et notamment les textes dispose de ses propres techniques de codage. Plusieurs approches de représentation des documents textuels ont été proposées dans ce contexte, la plupart étant des méthodes vectorielles. Les principales méthodes de représentation de textes n'utilisent pas d'information grammaticale ni d'analyse syntaxique des mots : seule la présence ou l'absence de certains mots est porteuse d'informations. Un état de l'art des différentes approches de représentation et codage de textes est développé dans ce chapitre.

3.2 Prétraitements de données textuels

Les données textuelles sont une forme particulière de données complexes. Elles ne sont pas délimitées, structurées et étiquetées sémantiquement de façon explicite. En conséquence ces données nécessitent un traitement préalable. De manière générale, l'objectif de ce prétraitement est de minimiser l'espace de recherche. Le prétraitement est généralement effectué en cinq étapes séquentielles :

1. La segmentation
2. Suppression des mots fréquents

3. Suppression des mots rares
4. Le traitement morphologique
5. Le traitement syntaxique

3.2.1 La segmentation

La première opération que doit effectuer un système de classification est la reconnaissance des termes utilisés. La segmentation consiste à découper la séquence des caractères afin de regrouper les caractères formant un même mot.

Habituellement, cette étape permet d'isoler les ponctuations (reconnaissance des fins de phrase ou de paragraphe), ensuite découper les séquences de caractères en fonction de la présence ou l'absence de caractères de séparation (de type « espace », « tabulation » ou « retour à la ligne »), puis regrouper les chiffres pour former des nombres (reconnaissance éventuelle des dates), de reconnaitre les mots composés.

3.2.2 Suppression des mots fréquents ou élimination des "Mots Outils"

Les mots qui apparaissent le plus souvent dans un corpus sont généralement les mots grammaticaux, mots vides (empty words) ou mots outils (stop words) : les articles, les prépositions, les mots de liaisons, les déterminants, les adverbes, les adjectifs indéfinis, les conjonctions, les pronoms et les verbes auxiliaires etc.., qui constituent une grande part des mots d'un texte, mais malheureusement sont faiblement informatifs, sur le sens d'un texte puisqu'ils sont présents sur l'ensemble des textes. A titre d'exemple on peut citer en dans la langue Française, le cas des articles « le », « la », « les » ou de certains mots de liaison « ainsi », « toutefois » etc... Ou en Anglais : Les prépositions (about, after, though.), les déterminants (the, no, one.), les conjonctions (though, and, or.), les adverbes (above, almost, yet.), les pronoms (who, another, few.) et certains verbes (are, can, have, may,will.).

Ces termes très fréquents peuvent être écartés du corpus pour en réduire la dimension. Cette possibilité de réduire la taille des entrées de l'index en éliminant les mots vides s'explique par le fait que ces termes sont présents dans la quasi-totalité des documents et ont donc un pouvoir discriminant faible en comparaison avec d'autres termes.

Donc leur élimination lors d'un prétraitement du document permet par la suite de gagner beaucoup de temps lors de la modélisation et l'analyse du document. Ces mots doivent être supprimés de la représentation des textes pour deux raisons :

- D'un point de vue linguistique, ces mots ne comportent que très peu d'informations. La présence ou l'absence de ces mots n'aident pas à deviner le sens d'un texte. Pour cette raison, ils sont communément appelés « mots vides ».
- D'un point de vue statistique, ces mots se retrouvent sur l'ensemble des textes sans aucune discrimination et ne sont d'aucune aide pour la classification

L'élimination systématique du corpus des mots vides peut se faire par l'intermédiaire d'une liste prédéfinie de mots pour chacune des langues étudiées. Cependant, l'établissement d'une telle liste peut poser des problèmes. D'une part, il n'est pas facile de déterminer le nombre de mots exacts qu'il faut inclure dans cette liste. D'autre part, cette liste est intimement liée à la langue utilisée et n'est donc pas transposable directement à une autre langue.

Enfin, un dernier point concernant les opérateurs de négation (ex : pas, ne, non) qui peuvent être supprimés sans gravité. Dans un contexte de classification de textes, une notion affectée par un opérateur de négation reste inchangée contrairement à une négation dans un contexte de recherche d'information qui peut être déterminante pour les résultats attendus. Dans le cadre d'une recherche documentaire, le but à atteindre pour l'utilisateur de rechercher l'information en lien avec la requête. En revanche, dans le cade d'une classification de textes en plusieurs catégories, les opérateurs de négation ne vont guère influencer les résultats puisque l'on cherche à distinguer les thèmes les uns des autres. Par exemple les deux phrases suivantes : il est malade et il n'est pas malade traitent toutes les deux le même sujet de santé, et le terme malade, avec ou sans négation, est un terme décrivant cette notion de santé. En évidence, elles ont un sens opposé mais sont toutes liées au sujet de santé.

3.2.3 Suppression des mots rares

En général, les auteurs cherchent également à supprimer les mots rares, qui n'apparaissent qu'une ou deux fois sur un corpus, afin de réduire de façon appréciable la dimension des vecteurs utilisés pour représenter les textes, puisque, ces mots rares sont très nombreux.

D'un point de vue linguistique, la suppression de ces mots n'est pas nécessairement justifiée : certains mots peuvent être très rares, mais très informatifs. Néanmoins, ces mots ne peuvent pas être utilisés par des méthodes à bases d'apprentissage du fait de leur très faible fréquence ; il n'est pas possible de construire de statistiques fiables à partir d'une ou deux occurrences ; Une des méthodes communément retenues pour supprimer ces mots consiste à ne considérer que les mots dont la fréquence totale est supérieure à un seuil fixé préalablement.

Notons enfin, que les mots ne contenant qu'une seule lettre sont généralement écartés pour les mêmes raisons précédentes, comme par exemple le mot « D » dans la « Vitamine D » ou le mot « C » dans le « langage C ».

3.2.4 Le traitement morphologique

Consiste à effectuer un traitement au niveau de chacun des mots en fonction de leurs variations morphologiques : flexion, dérivation, composition afin de rassembler les mots de sens identiques. Donc, le but est de regrouper par exemple les termes «manger» et «mangent» ou les termes « cheval » et « chevaux » car ils ont la même signification.

L'intérêt de cette opération est la réduction des dimensionnalités de l'espace de codage des textes afin d'améliorer davantage la performance du système de classification en matière d'espace mémoire et vitesse de traitement.

3.2.5 La lemmatisation et la racinisation

3.2.5.1 Les Flexions et le Lemme

Les flexions sont les différentes formes fléchies d'un même mot. Les formes fléchies correspondent aux formes "conjuguées" ou "accordées" d'un mot de base non conjugué et non accordé : le Lemme.

Par exemple, le mot "jouer", verbe à l'infinitif ni accordé, ni conjugué est un lemme. Il possède différentes flexions qui correspondent à ses formes conjuguées à diverses personnes et temps : "il jouera", "nous jouons", "ils ont joué", …

3.2.5.2 Lemmatisation

La lemmatisation désigne l'analyse lexicale du contenu d'un texte regroupant les mots d'une même famille. Chacun des mots d'un contenu se trouve ainsi réduit en une entité appelée lemme. La lemmatisation regroupe les différentes formes que peut revêtir un mot soit : le nom, le pluriel, le verbe à l'infinitif, etc.

Plusieurs ressources et logiciels existent pour réaliser cette tâche. Parmi eux, nous citerons :

- Le Dictionnaire électronique des formes fléchies du Français (DELAF), développé par l'Institut d'Electronique et d'Informatique Gaspard-Monge (IGM)
- TreeTagger, développé par l'Université de Stuttgart et dont les ressources linguistiques sont disponibles pour de multiples langues dont le français.
- Le Lexique Électronique des Formes Fléchies du Français (LEFFF) (même si on nous parle "d'ivresse des mots", il reste consommable sans modération), développé par l'Inria et le Laboratoire Bordelais de Recherche en Informatique (LABRI) depuis 2003.

Parmi ces choix, TreeTagger se démarque pour deux raisons :

- Sa capacité d'adaptation à de multiples langues par l'intermédiaire de ces fichiers de configuration. On dénombre d'après la page officielle de l'outil, la gestion de 13 langues différentes : l'allemand, l'anglais, le français, l'italien, le néerlandais, l'espagnol, le bulgare, le russe, le grec, le portugais, le chinois, le Swahili et le vieux français.
- La présence d'un algorithme sous la forme d'un arbre de décision lui permettant de décider en contexte de la fonction grammaticale à attribuer au mot et d'en déduire le lemme le plus probable. Sur ce point, les autres alternatives (le LEFFF et le DELAF) nous fournissent uniquement pour chaque forme fléchie, la liste des lemmes possibles en fonction de la fonction grammaticale du mot (on parle alors de lexiques). Ces réflexions nous amènent à aborder les deux types de

lemmatisation existantes : la “lemmatisation en contexte” (comme celle réalisée par TreeTagger) et la “lemmatisation hors contexte” (que l’on peut réaliser avec les lexiques présentés).

3.2.5.3 Racinisation / Stemmatisation

La stemmatisation ou racinisation est le nom donné au procédé qui vise à transformer les flexions en leur radical ou stemme.

Quelque soit l’outil retenu, la façon de procéder est toujours la même : le stemmatiseur recherche selon la forme du mot fléchi et la langue définie, le radical le plus probable pour ce mot. Contrairement à la lemmatisation qui repose donc sur une base de connaissance des formes fléchies de la langue auxquelles on associe les lemmes possibles (appelée lexique), la stemmatisation fonctionne uniquement avec une base de connaissance des règles syntaxiques et grammaticales de la langue.

Cela provoque deux différences notables avec la lemmatisation :

- La stemmatisation est moins sensible aux fautes d’orthographes que la lemmatisation. La lemmatisation échoue à la moindre faute d’orthographe (la forme fléchie servant à la recherche dans la base de connaissance devenant inconnue ou erronée) alors que la stemmatisation peut réussir si la faute ne perturbe pas la détection du radical et si celui-ci n’est pas modifié.
- La stemmatisation n’a pas besoin du contexte pour fonctionner. Elle ne requiert que le mot à raciniser et la langue dans laquelle ce mot est écrit.

Plusieurs stemmers ont été développés pour déterminer les racines lexicales, l’algorithme le plus couramment utilisé pour la langue anglaise est celle de PORTER.

3.2.6 Le traitement syntaxique

Traite les combinaisons et l'ordre des mots dans la phrase. Le traitement syntaxique identifie et regroupe un ensemble de mots dont la sémantique dépend de leur association. Par exemple, les mots « La syntaxe casque bleu » ne signifient habituellement pas qu’on a affaire à un casque qui est bleu, mais plutôt à une organisation militaire dépendante de l’ONU. L’analyseur syntaxique a pour but d’identifier ce type de cas. La phase d’analyse syntaxique consiste aussi à éliminer des ambiguïtés comme par exemple les problèmes d’homographie.

3.3 La représentation des documents

Il existe dans la littérature de nombreux modèles de représentation de documents textuels. Nous pouvons citer, par exemple, l’utilisation de vecteurs dont les composantes représentent des termes [Sal 1973, Sal 1989], des matrices de distribution de termes ou de relations entre termes. Dans la section suivante, nous allons définir les différents modèles, utilisés dans la littérature, pour la représentation d’un document texte.

3.3.1 Le modèle standard MS « sacs de mots »

Dans le cadre du modèle vectoriel standard (MS), les textes sont considérés comme des « sacs de mots » [Sal 1971a, Sal 1971b, Sal&McG 1983]. L'idée principale est de transformer les différents documents d'une base documentaire en vecteurs où chacun des éléments d'un vecteur de texte représente des unités textuelles ou tout simplement des mots appelés aussi « termes d'indexation ».

Dans le modèle vectoriel standard, les composantes d'un vecteur représentant un texte sont fonction de l'occurrence des mots dans le texte.

Soit C une collection de documents textuels, Di un document de C, soit t le nombre de termes d'indexation et $T = \{T1 \dots, Tj, \dots, Tt\}$ l'ensemble de ces derniers. Dans le modèle vectoriel standard, le document Di est représenté par un vecteur Vi. La collection de textes peut être ainsi représentée par une matrice dont les colonnes représentent les termes d'indexation et les lignes représentent les documents de cette collection.

$$Vi = (W1, \dots, Wj \dots Wt),$$

où Wj est le poids d'un terme Tj dans le document Di et j = 1..t.

Le poids donné à un terme d'indexation dans un document est calculé en fonction de la fréquence d'occurrence du terme TF (Term Frequency) dans le document et du nombre de documents contenant le terme IDF (Inverse Document Frequency). Les calculs des poids et les pondérations accordées à un document D ont fait l'objet de nombreuses études [Sin 1997, Lee 1995, Buc&al 1992, Sal&Buc 1988].

3.3.2 Représentation des textes par des collocations

Cette approche proposée ici, consiste à regrouper certains mots (collocations) afin d'obtenir des descripteurs ou expressions plus porteurs de sens au lieu d'utiliser des mots isolés composant le texte.

Identifier des collocations consiste à trouver des mots qui "vont ensemble" et qu'il est naturel de trouver proches dans le langage. Pour former ces groupes de mots, on n'a pas besoin de syntagmes nominaux, juste des paires de mots qui peuvent être séparés par des mots vides.

Le but n'est pas ici de chercher à analyser les textes d'un point de vue syntaxique, mais les représenter selon un ensemble d'usages de la langue, qui ont une influence sur le système classification. (Par exemple : repas-bien-garni, parler-en-connaissance-de-cause, tout-à-fait-normal).

Rémi Lavalley, Patrice Bellot et Marc El-Bèze dans (Lavalley & all, 2009), expliquent pourquoi le fait de considérer une suite de mots comme une seule unité informative permet d'améliorer les performances d'un système de classification dans leur article « Interactions entre le calcul de collocations et la catégorisation automatique de textes ».

Tout d'abord pour augmenter la significativité du terme : par exemple, si nous arrivons à repérer l'expression « effet particulièrement désagréable » dans un texte, on pourra préjuger que la critique est négative, alors qu'un système classique aurait pris les mots séparément et aurait pu décider autrement, par exemple :

– effet : fait pencher vers une critique positive (comme dans l'expression "cette odeur fait bon effet") ;

– désagréable vers une critique négative.

Ainsi, en traitant l'expression dans son intégralité nous pensons accroître son pouvoir discriminant.

La seconde raison qui pousse à penser que l'on peut améliorer les résultats, selon (Lavalley & all, 2009), vient du fait qu'on peut envisager de créer des collocations propres à une catégorie pendant la phase d'apprentissage.

Pour extraire les collocations présentes dans le corpus d'apprentissage, ils se sont appuyés sur la méthode du Rapport de Vraisemblance.

Néanmoins, parmi les grands problèmes dans cette approche est de savoir lesquelles garder : toutes n'ayant pas la même pouvoir discriminant sur la classification finale, certaines peuvent en effet se recouper.

3.3.3 Représentation des textes avec la méthode N-grammes de mots

Il existe beaucoup de mots ayant la même forme, mais des sens différents. Par exemple, "prix" n'a pas le même sens dans "prix Goncourt", "grand prix" ou "prix marchandise". Ces mots augmentent l'ambiguïté du sens des textes (classification erronée). En utilisant les N-grammes de mots (N mots consécutif), un sens parmi d'autres est favorisé. Par exemple, "actes biologie" (issu des données d'ITESOFT) est un bi-gramme de mots particulièrement pertinent.

Nous donnons ci-dessous des exemples (issus des données d'ITESOFT) de N-grammes de mots :

– N=1 (uni-gramme) : "biologie", "médicale", "frais", "accessoires" et "maladie".

– N=2 (bi-grammes) : "biologie médicale", "frais maladies" et "malade no ".

– N=3 (trigrammes) : "malade no facture", "prescription actes renseignements".

Paradis et Nie (2005) appliquent une telle représentation afin d'effectuer une classification de documents. La méthode consiste à classer les documents bruités en se fondant sur le filtrage du contenu avec les N-grammes de mots et les entités nommées sur des documents de type "appels d'offres". Les travaux de Tan et al. (2002) utilisent des bi-grammes ou des uni-grammes de mots comme descripteurs pour la représentation des données pour une tâche de classification. Les résultats révèlent que l'utilisation des bi-grammes sur ces données améliore les résultats de manière significative.

Notons que l'exemple donné précédemment montre clairement que l'utilisation de N-grammes de mots favorise un sens parmi d'autres. Cependant, en se fondant sur le groupement de mots, nous pourrions dans certains cas dégrader la classification en introduisant une quantité supplémentaire de bruit. Par exemple dans le cas d'utilisation d'un trigramme de mots, les trois mots le composant vont être éloignés du sens de chacun des mots.

3.3.4 Représentation des textes avec la méthode des n-grammes de caractères

Le N-gramme de caractères est une séquence de N caractères issus d'une chaîne de caractères. La notion de N-grammes de caractères a été utilisée de manière fréquente dans l'identification de la langue ou dans l'analyse de corpus oraux (Jalam et Teytaud (2001)). Dans les recherches récentes, elle est utilisée pour l'acquisition et l'extraction des connaissances dans les corpus. De nombreux travaux (Cavnar et Trenkle (1994), Náther (2005)) utilisent les N-grammes de caractères comme méthode de représentation de documents d'un corpus pour la classification.

L'ensemble des N-grammes de caractères (en général, N varie de 2 à 5) est le résultat du déplacement d'une fenêtre de N cases sur le texte. Ce déplacement s'effectue par étapes, une étape correspondant à un caractère. Ensuite les fréquences des N-grammes de caractères sont calculées. Par exemple, nous avons les N-grammes ci-dessous (N=3) avec la phrase suivante : "la nourrice nourrit le nourrisson"

Trigrammes = [la_=1, a_n=1, _no=3, nou=3, our=3, urr=3, rri=3, ric=1, ice=1, _ce=1, e_n=2, rit=1, it_=1, t_l=1, _le=1, le_=1, ris=1, iss=1, sso=1, son=1].

Dans cet exemple, l'espace est représenté par le caractère "_", pour faciliter la lecture.

3.3.5 Représentation des textes basée sur les concepts

Les approches précédentes n'extraient pas la sémantique d'un document mais simplement une comparaison morphologique. Si on peut supposer que chaque terme a un sens, il est plus difficile de prouver que deux documents étant composés des mêmes termes aient forcément le même sens. Les auteurs proposent donc, une nouvelle approche de représentation textuelle « plus sémantique » basée non pas sur les termes présents sur le texte à traiter mais sur les concepts correspondants. Ainsi, au lieu de définir un espace vectoriel dont chaque composante représente un terme (mot, stem, lemme, ou n-gram), on projette l'ensemble de termes du texte sur un ensemble fini de concepts.

Un concept peut représenter un objet matériel, une notion, une idée (Uschold & King, 1995). Trois éléments constituent la notion de concept {terme(s), notion, objet(s)}, un terme ou plusieurs, une notion et un ensemble d'objets. La notion, également appelée intention du concept, contient la sémantique du concept, exprimée en termes de propriétés et d'attributs.

L'ensemble d'objets, également appelé extension du concept, regroupe les objets manipulés à travers le concept ; ces objets sont appelés instances du concept. Par exemple, le terme « stylo », a pour intention « instrument nécessaire pour écrire ou dessiner », et a plusieurs réalisations : « marqueur, stylo à bille, stylo à encre, etc.… ».

Un concept est ainsi doté d'une sémantique référentielle (celle imposée par son extension) et d'une sémantique différentielle (celle imposée par son intension). Un concept ayant une extension vide est appelé concept générique, ces concepts génériques correspondent généralement à des notions abstraites (par exemple, la « vérité »).

L'avantage d'une telle méthode est en particulier de réduire les problèmes d'ambigüité et de synonymie dans le vocabulaire et de la construction syntaxique (restituer l'ordre des termes). Cette nouvelle approche de représentation permet donc, une factorisation des termes par regroupement de leur champ sémantique. (Jaillet & all, 2003). En effet plusieurs synonymes seront représentés par un seul concept, d'où une réduction de dimensionnalité considérable dans l'espace de codage.

Cependant, les deux inconvénients majeurs de ce type de codage restent, que les noms propres du texte ne sont pas pris en compte (Absents du thésaurus puisque ces derniers sont sémantiquement vides par définition), et le coût excessif pour la conception, la réalisation et la maintenance d'une telle solution appuyée sur les ontologies.

3.3.6 Représentation fréquentielle

Cette représentation consiste à présenter le texte sous forme de vecteur dont les éléments renseignent non seulement sur la présence ou l'absence d'un terme comme dans un vecteur binaire mais aussi informe sur le nombre de présences du terme dans le texte. Ainsi, un document est transformé en un vecteur dont les composantes vont correspondre au nombre d'occurrences des termes dans le document. Pour chaque document, un poids est attribué à chacun des termes qu'il contient. Une matrice « documents /termes » représente l'ensemble des documents (un vecteur est associé à chaque document, les composantes des vecteurs sont les poids des termes) (Salton & McGill, 1983).

Cette méthode conçoit le calcul du poids proprement dit des termes. Aucune analyse linguistique n'est utilisée, ni aucune notion de distances entre les mots. Ainsi les deux inconvénients majeurs de cette représentation, sont la non prise en charge des interactions des termes entre eux traduit par une indépendance de ces termes d'une part et d'autre part la déstructuration syntaxique du document causée par le fait que le modèle ne permet pas de conserver l'ordre des mots.

3.3.7 Représentation fréquentielle normalisée

Du point de vue statistique, la représentation fréquentielle confronte un problème majeur du fait qu'un texte long sera représenté par un vecteur dont la norme sera supérieure à celle de la représentation d'un document plus court. Il est donc recommandé de normaliser la représentation fréquentielle par rapport à la taille du document. Ainsi

le poids du terme sera le nombre d'occurrences de ce dernier dans le texte sur le nombre d'occurrences de tous les termes du texte.

3.3.8 Vecteur TF-IDF

La représentation fréquentielle se fonde sur le nombre d'occurrences du descripteur dans le corpus. Cependant, en procédant ainsi nous donnons une importance trop grande aux descripteurs qui apparaissent très souvent dans toutes les classes et qui sont peu représentatifs d'une classe en particulier. Nous trouvons dans la littérature (Salton et Buckley (1988)) une autre mesure de poids connue sous le nom TF.IDF (Term Frequency Inverse Document Frequency). Elle permet de mesurer l'importance d'un mot en fonction de sa fréquence dans le document (TF = Term Frequency) pondérée par la fréquence d'apparition du terme dans tout le corpus (IDF = Inverse Document Frequency). Cette mesure permet de donner un poids plus important aux mots discriminants d'un document. Ainsi, un terme apparaissant dans tous les documents du corpus aura un poids faible. Le poids d'un descripteur t_k dans un document D_j est calculé ainsi :

$$TF.idf(t_k, D_j) = TF(t_k, D_j) IDF(t_k) \qquad (1)$$

Où T F (t_k, D_j) est le nombre d'occurrences de $t_k$dans D_j, $IDF(t_r) = \frac{\log|s|}{DF(t_r)}$ avec |S| le nombre de documents dans le corpus et DF(t_R) est le nombre de documents contenant t_R. D'autres variantes du TF.IDF qui n'ont pas été utilisées dans nos travaux sont données par (Nobata et al. (2003)) :

$$TF.idf(t_k, D_j) = \frac{TF(t_k, D_j) - 1}{TF(t_k, D_j)} IDF(t_k) \qquad (2)$$

$$TF.idf(t_k, D_j) = \frac{TF(t_k, D_j)}{TF(t_k, D_j) + 1} IDF(t_k) \qquad (3)$$

À la différence de la formule (1) qui utilise la fréquence des termes à l'état brut (mesure utilisée dans notre étude), les deux formules (2) et (3) sont utilisées pour normaliser la fré- quence. Ainsi, les travaux de Robertson et Walker (1994) rapportent que la méthode (3) est plus efficace dans la recherche documentaire.

Une des particularités du TF.IDF tient au fait que cette mesure donne un poids faible aux mots présents dans l'ensemble des documents car ils ne sont pas discriminants (cas des "stop words"). Un tel traitement peut donc se révéler particulièrement pertinent pour le processus fondé sur les N-grammes de caractères qui n'utilisent pas de prétraitements (comme la suppression des "stop words").

3.4 Conclusion

Pour pouvoir appliquer les différents algorithmes d'apprentissage sur les documents de type textuels, un ensemble de techniques ont été développé pour montrer comment l'information textuelle est habituellement prise en compte pour la représentation « informatique » de ces documents. Les différentes approches de représentation informatique de textes sont exposées dans ce chapitre.

Ainsi avant la codification des documents, un ensemble d'opérations préliminaires doivent être faites pour épurer le texte de tous les mots inutiles et conserver seulement ceux qui sont porteurs d'informations et utiles pour le processus de classification. Mais malgré tous les prétraitements appliqués sur le document, l'espace des descripteurs, qui peuvent être des n-grammes, des stems, des phrases, des concepts ou tout simplement des mots, reste très grand et très creux, d'où la nécessité d'une diminution préalable de cet espace.

Finalement on peut qualifier notre texte, par fichier « informatique » apte à être employé dans les différentes méthodes d'apprentissage automatique.

Chapitre 4

L'Analyse des sentiments

4.1 Introduction

Il est de tradition dans toute cette communauté de l'opinion mining, voire du TAL en général, de distinguer deux grands types de textes : ceux qui relatent des faits et ceux qui présentent des opinions. Dans le premier cas, il s'agit de descriptions objectives. Dans l'autre, il s'agit d'expressions subjectives, qui peuvent porter par ailleurs sur des entités identiques descriptibles objectivement. Ce qui ne serait pas le cas d'expressions sur des états d'âme, qu'on ne peut rapporter à une entité objectivable. Cette distinction sommaire mérite d'être discutée et les développements de méthodes de classification sur ce thème sont considérables car de nombreuses occurrences troublent la frontière, comme nous le verrons.

De manière courante, l'étude de l'opinion en fouille de textes, ce qu'on appelle l'analyse de sentiment, se réduit à distinguer les opinions positives, les opinions négatives, les tonalités, et éventuellement à classer dans une catégorie neutre les opinions indécidables (parce qu'elles sont mixtes par exemple, ou qu'il n'y a pas d'opinion exprimée dans le document, ou parfois encore parce que la méthode ne peut trancher clairement dans l'ensemble du document).

Enfin, l'unité de traitement peut être le document, l'expression qualifiante au sein d'un segment, un élément (part-of-speech) tel qu'un adjectif, ou un ensemble de n-grams, dont il s'agit précisément de tester l'étendue pour formater la recherche.

4.2 Opinion Mining, Analyse des Sentiments

L'Opinion Mining' est le domaine qui s'occupe de traitement d'opinion, du sentiment, et de la subjectivité dans le texte et nous avons précisé que c'est un sous domaine de la catégorisation de texte.

Les principales tâches de l'Opinion Mining sont l'analyse de l'opinion et l'analyse de la subjectivité. Cette dernière est utilisée pour reconnaitre le langage décrit l'opinion afin de distinguer les langues objectives.

Le terme Opinion Mining apparaît dans un article de Dave [Dave et al. (2003)] qui a été publié dans l'acte de conférence WWW 2003. Selon Dave, l'Opinion Mining devrait "traiter un ensemble de résultats de recherche pour un cas donné, générer une liste des attributs (qualité, caractéristiques, etc.) et agréger des avis sur chacun d'entre eux (mauvais, modéré, de bonne qualité). Toutefois, l'Opinion Mining a récemment été interprétée de manière plus générale pour inclure de nombreux types d'analyse d'évaluation de texte [Liu (2006)].

Le terme "Analyse des Sentiments" est utilisé pour décrire l'analyse automatique de texte évaluatif et pour la recherche de valeur prédictive des jugements. Elle a été introduite dans les travaux de Das et Chen [Das & Chen (2001)] et Tong [Tong (2001)] en 2001 afin d'analyser des sentiments dans le cadre de l'économie de marché. Ensuite d'autres travaux sur l'analyse des sentiments ont été proposés par Turney.

Depuis 2002, un nombre important d'articles citant l'Analyse des Sentiments ont vus le jour, ces travaux se concentrent sur la classification des commentaires et à leur polarité (positif ou négatif). Aujourd'hui, l'Opinion

Mining et l'Analyse des Sentiments font partie du même domaine de recherche.

4.3 La complexité de notation d'opinion

Pour démontrer la complexité de la notation de l'opinion nous allons nous baser sur un exemple d'une critique cinématographique retrouvée sur le site IMDB.com.

L'exemple est le suivant :

It's A Wonderful Life. I've only met 2 people in real life and 1 person on the

IMDB who hates this one. My favorite film ever !

Comme nous pouvons le constater, la critique est composée de trois phrases qui ont une polarité opposée. Même si nous arrivons à déduire facilement que la première phrase étant le titre de film, It's a Wonderful Life, nous aurons deux phrases subjectives mais difficile à noter correctement. La dernière phrase est plutôt facile à noter : Mon film préféré de tout les temps. Mais le problème se pose pour la notation de la phrase : J'ai connu seulement ... qui ont détesté ce film. Car une étude statistique nous montre que la polarité est négative pour cette phrase pourtant la polarité est réellement positive et avec une très grande intensité.

Les résultats d'une étude de Pang et al. [Pang et al. (2002)], sur les critiques cinématographiques montrent que l'utilisation de mots clés corrects peut être moins triviale que l'on pourrait penser initialement. Le but de Pang et al. était de mieux comprendre la difficulté de classification de polarité des sentiments. Deux personnes ont été invitées à choisir des mots clés qu'ils considèrent comme de bons indicateurs des opinions positives et négatives. Les deux listes de mots clés réalisent environ 60% de précision.

En revanche, les listes de mots de la même taille, mais choisis en fonction de traitement statistiques sur un document d'apprentissage réalisent près de 70% de précision.

En effet, l'application de techniques d'apprentissage automatique (ML) basée sur les modèles d'uni-gramme peut atteindre plus de 80% de précision [Pang et al. (2002)], qui est beaucoup mieux que la performance basée sur les mots-clés des experts.

Les sentiments peuvent souvent être exprimés d'une manière très subtile, ce qui rend difficile l'identification par les unités du document quand nous les considérons séparément. Si nous considérons une phrase qui indique une très forte opinion, il est difficile d'associer cette opinion avec les mots-clés ou les expressions dans cette phrase. En général, les sentiments et la subjectivité sont très sensibles au contexte et dépendent de domaine. La dépendance de domaine est en partie une conséquence des changements de vocabulaire, par exemple la même expression peut indiquer différents sentiments dans différents domaines.

De plus sur l'internet chacun utilise son propre vocabulaire, ce qui rend la tâche plus difficile - même s'il s'agit du même domaine. En plus il est très difficile d'affecter

correctement le poids pour des phrases de la critique. Très souvent nous avons une description très positive d'un film, avec les meilleurs acteurs, meilleurs metteurs en scène, mais la dernière phrase peut être Malgré tout ça je suis sortie du cinéma avant la fin.

Même si nous arrivons à attribuer l'intensité de cette opinion nous nous retrouverons avec une seule opinion négative contre plusieurs positives.

Ces exemples montrent qu'il est encore impossible d'arriver à un cas idéal de notation des sentiments dans un texte écrit par les divers utilisateurs. Car ça ne respecte aucune règle et il est impossible de prévoir tout les cas possibles, en plus très souvent la même phrase peut être considérée comme positive pour une personne et négative pour une autre.

4.4 Différentes approches pour l'analyse des sentiments

4.4.1 Traitement de la négation

Une autre caractéristique importante pour déterminer la polarité de l'opinion est la négation. Le seul lemme décrivant la négation peut changer complètement la polarité de la phrase. Das et Chen [Das & Chen (2001)] proposent de rajouter une indication de négation "NON " à des mots qui se trouvent près de la négation, de sorte que dans la phrase "Je ne comprends pas", le lemme "comprendre" est converti en un nouveau lemme "comprendre-NON". Cependant, certaines apparences de la négation n'inversent pas la polarité de la phrase. Na et al. [Na et al. (2004)] améliorent de 3% la précision résultante de la modé- lisation de la négation. Ils analysent le texte en effectuant une décomposition spécifique d'une phrase en recherchant des occurrences de négation. Si ces dernières ne sont pas étiquetées comme des mots de négation prédéfinis, ils classent la phrase entière comme étant une phrase avec négation, et non le mot séparé. Une autre difficulté avec la négation est qu'elle peut être décrite de manière très subtile, ainsi le sarcasme et l'ironie sont très difficiles à détecter.

4.4.2 Approches lexicales

Les approches lexicales utilisent des dictionnaires de mots subjectifs, considérés comme des références universelles à partir de l'anglais.

Ces dictionnaires peuvent être généraux comme le General Inquirer, Sentiwordnet, Opinion Finder, NTU Sentiment Dictionary (NTUSD), etc. Ils peuvent également être construits *ad hoc* en fonction des corpus. Dans ces dictionnaires, une polarité est associée *a priori* à chacun des mots. Quel que soit le contexte dans lequel il sera inséré, le mot devrait ainsi avoir toujours la même polarité. On donne ensuite au document un score d'opinion en fonction de la présence de mots issus de ces dictionnaires dans le texte.

Ces dictionnaires sont utilisés pour classifier des textes dont on sait qu'ils parlent de l'entité nommée qui nous intéresse, par exemple pour les critiques de cinéma, le titre d'un film. La méthode consiste à détecter le terme (adjectif ou l'expression qualifiante) qui est en cooccurrence avec l'entité nommée, souvent au sein de la même phrase. Le dictionnaire fournit ainsi la tonalité affectée à l'entité concernée par ces termes qualifiants. De

nombreux problèmes, notamment syntaxiques, rendent problématique cette relation entre une entité et ses qualificatifs, c'est pourquoi d'autres traitements doivent être effectués pour obtenir un résultat satisfaisant. Aussi standardisée que puisse paraître cette approche par les dictionnaires, c'est elle qui est le plus souvent utilisée dans les services qui sont mis sur le marché et, en première approximation, elle donne des résultats intéressants, selon le niveau d'exigence que l'on se fixe.

WordNet Affect et SentiWordNet ne contiennent essentiellement que des mots simples. Or les passages évaluatifs peuvent parfois être exprimés au moyen d'expressions comportant plusieurs mots, voire sous des formes plus détournées encore. L'approche par les dictionnaires a donc des limites évidentes, mais elle a surtout l'avantage de permettre des calculs rapides sur de grands corpus.

4.4.2.1 Limites de l'analyse de sentiment à partir des lexiques

- Les dictionnaires affectent une tonalité positive ou négative à un mot, sans tenir compte du contexte, c'est-à-dire du texte environnant, comme des paramètres de la communication située ;
- Les dictionnaires ont tendance à éliminer les mots à valence ambiguë a priori ;
- Le traitement des expressions ambiguës reste à faire et demande de faire appel à d'autres principes et à d'autres techniques ;
- Lorsque la négation n'est pas prise en compte (ce qui peut paraître étonnant mais qui existe encore, par exemple dans les méthodes basées sur les « sacs de mots », qui calculent des fréquences d'occurrence dans un texte), le score de polarité peut être complètement faussé ;
- Ces dictionnaires ne permettent pas de traiter des figures de rhétorique qui peuvent changer entièrement la valence des expressions (le sarcasme, l'ironie : « encore une belle réussite de notre super président ! »).

4.4.3 Utilisation des méthodes d'apprentissage automatique

Les techniques d'apprentissage sont souvent mobilisées pour la catégorisation de textes ou l'extraction d'information. On ne s'intéressera qu'aux techniques d'apprentissage dédiées à l'analyse de sentiment.

Quelle est la procédure ? La machine est entraînée à détecter des expressions subjectives en la faisant travailler sur un premier corpus test. Elle est aussi entraînée à détecter des modèles ou des motifs dans le corpus. Elle doit être capable de détecter ensuite ces modèles dans le corpus lui-même, voire d'en détecter de nouveaux, proches de ceux qu'elle connaît déjà.

4.4.3.1 Machines à Vecteurs Support – SVM

4.4.3.1.1 Présentation de l'approche L'algorithme

SVM (Support Vector Machine) est une méthode d'apprentissage supervisée relativement récente introduite pour résoudre un problème de reconnaissance de formes à deux classes (Vapnik, 1995). Le principe de SVM a été proposé par Vapnik à partir de la théorie du risque empirique.

La méthode SVM est un classificateur linéaire utilisant des mesures de distance. En ce qui concerne son application à la problématique de catégorisation de documents, l'algorithme repose sur une interprétation géométrique simple est l'idée générale est de représenter l'espace des exemples (ici des documents) dans un espace vectoriel où chaque document étant un point dans cet espace et de trouver la meilleure séparation possible de cet espace en deux classes. L'espace de séparation est une surface de décision appelée marge, défini par les points « vecteur support ». Ces points se trouvent au minimum de marge. La marge se présente alors comme la plus courte distance entre un vecteur de support et "son" hyperplan. La marge se définit comme la plus petite distance entre les exemples de chaque classe et la surface séparatrice S :

$$marge\ (S)\ =\ \sum_{c_j\,\epsilon\,C} min_{x_i\,\epsilon\,c_j}\ (d(x_i, S))$$

Ainsi la décision s'appuie sur les SVM pour couper l'espace en deux : d'un côté, ce qui est dans la catégorie, de l'autre côté, ce qui n'y est pas.

L'approche par SVM permet donc de définir, par apprentissage, un hyperplan dans un espace vectoriel qui sépare au mieux les données de l'ensemble d'apprentissage en deux classes, minimisant le risque d'erreur et maximisant la marge entre deux classes. La qualité de l'hyperplan est déterminée par son écart avec les hyperplans parallèles les plus proches des points de chaque classe. Le meilleur hyperplan est celui qui a la marge la plus importante. SVM a été étendu pour les points ne pouvant être séparées de manière linéaire (par exemple notre cas des vecteurs de documents), en transformant l'espace initial des vecteurs de données à un espace de dimension supérieure dans lequel les points deviennent séparables linéairement. Nous trouvons dans (Joachims, 1998) une application efficace des SVM. La figure 3.1 montre une telle séparation dans le cas d'une séparation linéaire par un hyperplan.

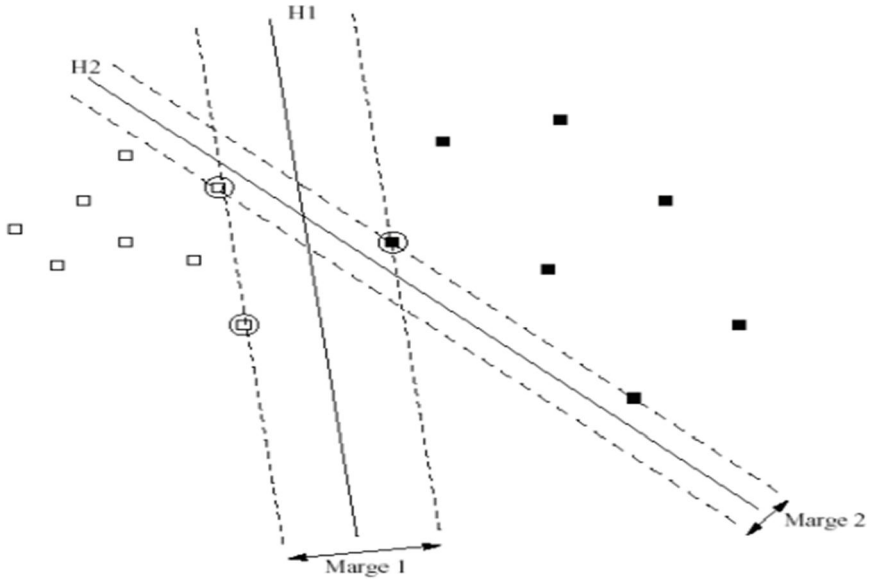


Figure **Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.**.1: Exemples d'hyperplans séparateurs en dimension deux. Les vecteurs de support sont encerclés.

Dans l'exemple de la figure 3.1, les exemples des deux classes peuvent être séparés par un hyperplan, le problème est dit linéairement séparable. Les deux hyperplans H_1 et H_2 sont tous les deux des séparateurs acceptables, mais l'hyperplan H_1 a une plus grande marge et sera donc préféré. Pour calculer l'hyperplan optimal et donc la marge, seuls les exemples les plus proches de la zone-frontière sont mis à contribution. L'apprentissage consiste à déterminer ces exemples appelés vecteurs de support. Tous les autres peuvent être écartés et n'interviennent plus dans les calculs.

Le problème se traduit mathématiquement en un problème d'optimisation quadratique : trouver l'hyperplan (w, b) (b est la distance à l'origine de l'hyperplan) qui minimise la norme de w sous les contraintes :

$$\forall d_i , c_i \ (w. d_i - \ b) \ \leq 1$$

Avec d_i le i^{éme} document, de classe c_i (+1 ou –1). Si les exemples ne sont pas linéairement séparables, on peut les plonger conceptuellement dans un espace de dimension plus grande (la dimension peut même être infinie) par une fonction de transformation appelée noyau (kernel). Dans cet espace, les exemples seront plus facilement séparables. Une propriété de l'algorithme est qu'il ne requiert pas les coordonnées de chaque exemple mais seulement les produits scalaires de chaque couple d'exemples, qui restent calculable une fois les exemples plongés dans un nouvel espace même de dimension infini.

Si cela ne suffit pas pour rendre les exemples séparables, il est possible d'ajouter encore un terme correctif qui autorise un nombre limité d'exemples à être mal classés. Pendant l'apprentissage, on cherchera à rendre ce terme le plus petit possible. Un paramètre de l'algorithme permet de donner plus ou moins d'importance à ce terme correctif. Dans sa formulation initiale, SVM ne peut gérer que des problèmes bi-classes (des extensions commencent à apparaître pour faire du SVM multi-classe). La méthode la plus commune

pour résoudre un problème multi-classe reste de le transformer préalablement en plusieurs sous problèmes bi-classe.

Cet algorithme est particulièrement bien adapté à la catégorisation de textes car il est capable de gérer des vecteurs de grande dimension. Dans la pratique, les catégories sont quasiment toujours linéairement séparables, il n'est donc pas nécessaire d'employer les méthodes avec des noyaux sophistiqués qui alourdissent inutilement les calculs. SVM a été introduit dans le domaine pour la première fois par Joachims qui a notamment travaillé à rendre SVM compatible avec les données textuelles qui sont caractérisées par de grandes dimensions avec des matrices (documents * termes) très creuses.

4.4.3.2 Naïve Bayes

L'algorithme Naïve Bayes (NB), est une autre méthode bien utilisée en apprentissage, elle est également employée dans la classification automatique de textes. NB est très connu pour son efficacité, sa facilité d'implémentation et ses résultats considérables.

4.4.3.2.1 Description de l'approche

L'algorithme Naïve Bayes (NB) est une méthode basée sur le modèle probabiliste, il vise à estimer la probabilité conditionnelle d'une catégorie sachant un document et affecte au document la (ou les) catégorie(s) la (les) plus probable(s). Ainsi ce classifieur va donc tenter d'estimer la probabilité qu'un document d fasse partie de la classe c.

Un modèle probabiliste de classification est un modèle qui permet le calcul pour chaque classe $c \in C$ et chaque document $d \in D$ la probabilité $P(c / d)$ que le document soit assigné à cette classe.

Le nom Naïve Bayes découle du fait que l'algorithme utilise le théorème de Bayes sans toutefois prendre en compte les dépendances existantes entre les variables (Dans notre cas les termes c'est des mots ou n-grammes, etc..) ; de ce fait, ses suppositions sont dites naïves. Donc la partie naïve de ce modèle est l'hypothèse d'indépendance des termes, c'est à dire que la probabilité conditionnelle d'un terme sachant une catégorie est supposée indépendante de cette probabilité pour les autres termes.

Dans un contexte probabiliste bayésien général, le choix de la classe d'une nouvelle instance **A** est réalisé par la règle du maximum a posteriori (MAP). L'estimation des probabilités est réalisée en général par maximisation de la vraisemblance conditionnelle de l'ensemble d'apprentissage par la formule $P(B) * P(A/B)$ qui fait apparaître deux termes : la vraisemblance de l'observation **A** conditionnellement à **B** et la probabilité a priori de **B**. Dans un cadre supervisé, ces deux distributions sont estimées à partir des exemples d'apprentissage. Si l'on considère un problème à deux classes c_1 et c_2, L'apprentissage consiste à déterminer la distribution de probabilité connaissant les données d'apprentissage : une probabilité fixée a priori, et, une fois que les données d'apprentissage ont été observées, cette probabilité a priori est transformée en probabilité a posteriori grâce au théorème de Bayes.

Pour la probabilité a priori c'est la probabilité conditionnelle $P(d_i / c_j)$. Cela représente la probabilité d'avoir $d_i$comme entrée quand on sait que l'on est dans la classe

c_j. La classification d'un exemple s'obtient alors par estimation de $P(c_j / d_i)$; la probabilité connaissant l'exemple d_i que celui-ci fasse partie de la classe c_j. Le choix optimal (pour minimiser le taux d'erreur) est de mettre l'exemple dans la classe qui à la plus forte probabilité a posteriori. Comme cette probabilité n'est pas connue, il faut l'estimer à partir des données contenues dans le corpus d'apprentissage.

Fondés sur l'idée qu'on peut estimer la probabilité qu'un document appartient à une classe en connaissant la probabilité qu'une classe correspond à ce document. La formule de bayes permet « d'inverser » la probabilité conditionnelle :

$$P(c_j|d_i) = \frac{P(c_j) \times P(d_i / c_j)}{P(d_i)}$$

Comme le but est de discriminer les différentes classes (il suffit d'ordonner les $P(c_j|d_i)$ pour toutes les classes ; il est inutile d'obtenir la valeur exacte), on peut alors supprimer le terme $P(d_i)$ qui est le même pour toutes les classes. $P(c_j)$ est la probabilité a priori qui est le plus couramment estimée par le pourcentage d'exemples (Dans ce cas pourcentage de documents) appartenant à la classe c_j dans le corpus d'apprentissage.

$$P(c_j) = \frac{N(c_j)}{N}$$

Quant à la classification, l'estimation $P(d_i / c_j)$ est calculée à partir des deux formules suivant le modèle utilisé en remplaçant les probabilités par leur estimateur et la classe c_j ayant la probabilité la plus élevée est choisie.

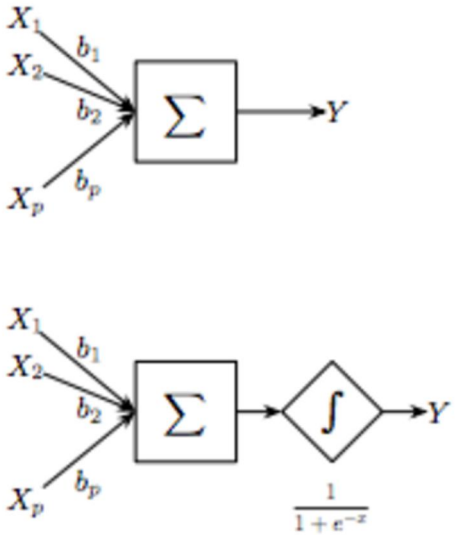
4.4.4 La régression logistique

La régression logistique permet de traiter le cas où la variable réponse est de type binaire (oui/non, malade/pas malade, etc.), et non pas continu comme dans le modèle de régression linéaire. Tout en relaxant certaines des hypothèses du modèle de régression multiple, on maintient quand même l'idée d'une relation linéaire entre la réponse et les prédicteurs.

Dans le cas d'une variable binaire, sa "moyenne" correspond à la proportion d'individus possédant la caractéristique étudiée ou répondant positivement à l'événement, d'où l'idée de modéliser la probabilité de succès, comprise entre 0 et 1, en fonction d'un certain nombre de prédicteurs. Dans les enquêtes épidémiologiques cas-témoin (avec ou sans matching) où l'incidence de la maladie n'est pas connue, la régression logistique fournit une estimation de l'odds-ratio ajusté sur les co-facteurs d'intérêt (âge, sexe, etc.). D'autre part, lorsque la prévalence de la maladie est faible, l'OR fournit une bonne approximation du risque relatif.

4.4.4.1 Parallèle avec la régression linéaire

Comme dans la régression linéaire, on cherche la meilleure combinaison linéaire des données d'entrée pour modéliser la réponse, à ceci près que c'est une transformation de cette combinaison (on parle d'une fonction de lien) qui est utilisée en sortie.



4.4.4.2 Le modèle de régression logistique

Si l'on note π la probabilité d'observer l'événement y = 1 (vs. 0), alors le log odds (transformation logit) peut s'exprimer comme une fonction linéaire des paramètres du modèle à p prédicteurs :

$$g(x) = \log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 x_1 + .. + \beta_p x_p$$

et la probabilité prédite s'écrit alors

$$P(y = 1 \mid x_1, x_2, \ldots, x_p) = \hat{y}_i = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + .. + \hat{\beta}_p x_p)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + .. + \hat{\beta}_p x_p)}$$

Dans ce type de modèle, on fait l'hypothèse que y_i suit une distribution binomiale et que les observations sont indépendantes. Notons également l'absence de terme d'erreur. L'estimation d'un tel type de modèle se fait par la méthode du maximum de vraisemblance. D'autres fonctions de lien existent (probit, log-log).

4.5 Conclusion

Dans ce chapitre nous avons présenté l'utilisation des techniques de domaines de catégorisation du texte et d'apprentissage automatique pour les besoins d'analyse des sentiments. Comme nous pouvons le constater ce domaine de recherche est utilisable dans plusieurs problématiques comme la détection du spam, l'analyse des dépêches de

presse, l'analyse des dépêches politiques, l'analyse des dépêches médicales, la génération automatique de réponses ou la production d'un résumé à partir d'un texte. Nous nous sommes focalisés sur l'utilisation de l'analyse des sentiments pour la notation des critiques sur les produits alimentaires d'Amazon.

Chapitre 5

Implémentation et résultats

5.1 Introduction

Afin de réaliser l'analyse des sentiments, il existe différentes approches parmi lesquelles : approche lexicale, approche par apprentissage automatique. Dans ce travail nous avons choisie la méthode d'apprentisage automatique qui s'adapte mieux aux données massives (Big data).

Nous avons appliqué cette méthode pour extraire les sentiments (positifs, négatifs) à partir du corpus "Amazon fine foods".

Dans la suite de ce chapitre nous allons présenter plus en détail les techniques de représentations utilisées, les classifieurs et les résultats.

5.2 L'ensemble de données (dataset)

L'ensemble de données "Amazon fine foods reviews" est constitué des commentaires sur les produits alimentaires "fine" de Amazon. Les données couvrent une période de plus de 10 ans (Oct 1999 - Oct 2012), comportant environ 570.000 avis. Les avis comprennent le produit et des informations sur l'utilisateur, des notes , et un commentaire textuel.

5.2.1 Format de l'ensemble de données

L'ensemble de données "Amazon fine foods reviews" est constitué de huit attributs **productId, userId, profileName, helpfulness, score, time, summary, text.**

Un exemple d'une instance du corpus :

product/productId: B001E4KFG0
review/userId: A3SGXH7AUHU8GW
review/profileName: delmartian
review/helpfulness: 1/1
review/score: 5.0
review/time: 1303862400
review/summary: Good Quality Dog Food
review/text: I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.

L'attribut product/productId désigne l'ID du produit, review/userID désigne l'ID de l'utilisateur, review/profileName désigne le nom de l'utilisateur, review/helpfulness désigne le nombre des utilisateurs qui ont trouvé l'avis utile, review/score désigne l'evaluation donné au produit par un utilisateur, review/time désigne le temps de l'avis (unix time), review/summary désigne le résumé de l'avis et review/text désigne le commentaire de l'utilisateur (texte).

Des statistiques sur le corpus :	
Nombre de avis	568,454
Nombre des utilisateurs	256,059
Nombre des produits	74,258
Utilisateurs avec > de 50 avis	260
Le no. moyen des mots par avis	56
Timespan	Oct 1999 - Oct 2012

Dans ce qui suit, nous verrons les étapes de la construction des modèles afin d'effectuer l'analyse des sentiments sur le corpus "Amazon fine foods" avec le Framework Apache SPARK [voir 2.3.6] en utilisant l'API de python de SPARK « pyspark ».

5.3 Importation et sélection des données

Tant que le corpus est sous format sqlite, il faut tout d'abord le connecter à Spark en utilisant la fonction **sqlite3()**. Ensuite, il faut passer une requête sql à la table connectée à Spark usant **read_sql_query()** afin de sélectionner les colonnes qui nous intéressent pour réaliser l'analyse « Score, Summary ». (figure 5.1)

[2]:
```
import pandas as pd

con = sqlite3.connect('database.sqlite')

messages = pd.read_sql_query("""
SELECT Score, Summary
FROM Reviews
""", con)
messages.head(20)
```

[2]:

	Score	Summary
0	5	Good Quality Dog Food
1	1	Not as Advertised
2	4	"Delight" says it all
3	2	Cough Medicine
4	5	Great taffy
5	4	Nice Taffy
6	5	Great! Just as good as the expensive brands!
7	5	Wonderful, tasty taffy

Figure 5.1: Connexion de la base de données à Spark.

Après, nous devons créer un Dataframe() des résultats de la "sql query" pour avoir la possibilité d'appliquer les fonctions issues de librairie "Mllib" sur ces données.

```
In [3]: food = sqlContext.createDataFrame(messages)
```

Figure 5.2: Création du dataframe.

5.4 Analyse du corpus

Nous avons fait une analyse statistique du corpus pour calculer la fréquence des Scores.

Score	Fréquence
1	52268
2	29769
3	42640
4	80655
5	363122

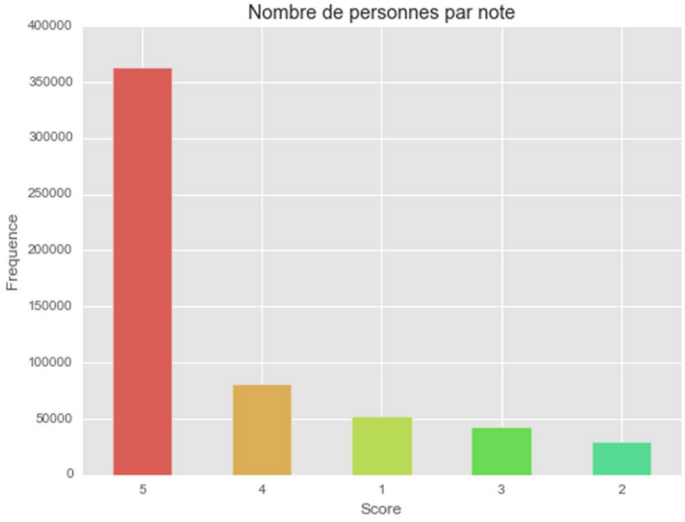


Figure 5.3: Fréquence des scores.

Nous avons affecté aux scores supérieurs ou égals à 3 la valeur 1 (positif) et pour les autres la valeur 0 (négatif).

```
def labelForResults(s):
    if s < 3:
        return 0.0
    else:
        return 1.0
```

```
label = UserDefinedFunction(labelForResults, DoubleType())
labeledData = food.select(label(food.Score).alias('label'), food.Summary).cache()
```

Figure 5.4: Affectation des labels.

label	Summary
1	Good Quality Dog Food
0	Not as Advertised
1	"Delight" says it all
0	Cough Medicine
1	Great taffy
1	Nice Taffy
1	Great! Just as good as the expensive brands!
1	Wonderful, tasty taffy
1	Yay Barley
1	Healthy Dog Food
1	The Best Hot Sauce in the World
1	My cats LOVE this "diet" food better than thei...
0	My Cats Are Not Fans of the New Food

Figure 5.5: Affichage des labels.

Nous avons calculé les pourcentages de chaque classe. Notant que le pourcentage de la classe « positif (1) » est largement supérieur à celui de la classe « négatif (0) ». (Figure 5.6)

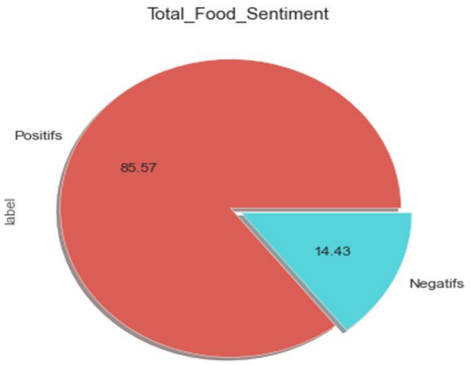


Figure 5.6: Les pourcentages des classes.

Les figures 5.7 et 5.8 représentent les mots « négatif » et « positif » respectivement dont la taille dépend de la fréquence d'apparition dans le corpus en utilisant la technique WordCloud.

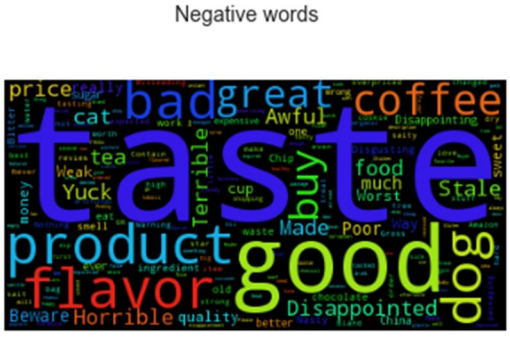


Figure 5.7: Les mots négatifs.

Figure 5.8: Les mots positifs.

Nous devons partitionner, en utilisant la fonction randomSplit(), aléatoirement le corpus en deux sous-ensembles disjoints: le premier représente 60% du corpus et servira à l'apprentissage des modèles de prédiction, le second sera utilisé pour tester ces modèles.

5.5 Prétraitement du texte

Dans cette étape, nous avons utilisé trois différentes méthodes de prétraitement des données textuels :

1. Tokenisation en utilisant la fonction Tokenizer() de la librairie MLlib.

```
tokenizer = Tokenizer(inputCol="Summary", outputCol="words")
```

Figure 5.9: Le « Tokenizer ».

label	Summary	words
1	"Delight" says it all	["delight", says, it, all]
1	Nice Taffy	[nice, taffy]
1	Great! Just as good as the expensive brands!	[great!, , just, as, good, as, the, expensive,...
1	Yay Barley	[yay, barley]
1	The Best Hot Sauce in the World	[the, best, hot, sauce, in, the, world]

Figure 5.10: Résultats de tokenisation.

2. La racinisation en utilisant la fonction PorterStemmer() de la librairie NLTK.

```
def racinisation(text):
    lowercased = [e.lower() for e in text.split() if len(e) >= 3]
    no_punctuation = []
    for word in lowercased:
        punct_removed = ''.join([letter for letter in word if not letter in PUNCTUATION])
        no_punctuation.append(punct_removed)
    stemmed = [STEMMER.stem(w) for w in no_punctuation]
    return [w for w in stemmed if w]
```

Figure 5.11: Racinisation.

label	words
1	[good, qualiti, dog, food]
1	[great, taffi]
1	[wonder, tasti, taffi]
1	[healthi, dog, food]
1	[fresh, and, greasi]
1	[strawberri, twizzler, yummi]
1	[lot, twizzler, just, what, you, expect]
1	[delici, product]
1	[twizzler]
1	[pleas, sell, these, mexico]

Figure 5.12: Résultats de la racinisation.

3. La lemmatisation en utilisant la fonction WordNetLemmatizer() de la librairie NLTK :

```
lemmatizer = WordNetLemmatizer()
def lemmatize(text):
    lowercased = [e.lower() for e in text.split() if len(e) >= 3]
    no_punctuation = []
    for word in lowercased:
        punct_removed = ''.join([letter for letter in word if not letter in PUNCTUATION])
        no_punctuation.append(punct_removed)
    lemmed = [lemmatizer.lemmatize(w) for w in no_punctuation]
    return [w for w in lemmed if w]

lemming = UserDefinedFunction(lemmatize, ArrayType(StringType()))
trainlme = trainset.select(trainset.label, lemming(trainset.Summary).alias('words'))
testlme = testset.select(testset.label, lemming(testset.Summary).alias('words'))
```

Figure 5.13: Lemmatisation.

1	[great, bargain, for, the, price]
1	[the, best, hot, sauce, the, world]
1	[best, the, instant, oatmeal]
1	[good, instant]
1	[great, irish, oatmeal, for, those, hurry]
1	[love, gluten, free, oatmeal]
1	[good, way, start, the, day]
1	[wife, favorite, breakfast]
1	[why, wouldnt, you, buy, oatmeal, from, mcanns...

Figure 5.14: Résultats de la lemmatisation.

5.6 Méthode de représentation

Nous avons utilisé deux modèles de représentation Sac de mots et Ngram pour chaque méthode il existe deux méthode de pondération TF et tf.IDF à l'aide de la librairie "Mllib" de Spark.

```
tokenizer = Tokenizer(inputCol="Summary", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
```

Figure 5.15: Sac de mots avec TF.

label	Summary	words	features
1	"Delight" says it all	["delight", says, it, all]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
1	Nice Taffy	[nice, taffy]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
1	Great! Just as good as the expensive brands!	[great!, , just, as, good, as, the, expensive,...	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
1	Yay Barley	[yay, barley]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...

Figure 5.16: Résultats de la représentation sac de mots avec TF.

```
from pyspark.ml.feature import NGram
ngram = NGram(inputCol="words", outputCol="ngrams")
hashingTFN = HashingTF(inputCol="ngrams", outputCol="features")
```

Figure 5.17: Ngram avec TF.

5.7 Apprentissage

L'étape suivante est consacrée à la construction des modèles et à l'attribution des étiquettes (0,1) aux individus de l'échantillon test.

Nous avons construit 36 modèles en utilisant trois algorithmes de classification: régression logistique, naïve bayes et svm. Pour chaque algorithme nous avons utilisé deux méthodes de représentations : Sac de mot et Ngram. Pour chacune de ces méthodes nous avons appliqué trois techniques de prétraitements : tokenisation, racinisation et lemmatisation, avec les pondérations TF et Tf.IDF.

Afin de faire l'apprentissage d'un modèle, nous avons utilisé des pipelines de Spark (Figure 5.18).

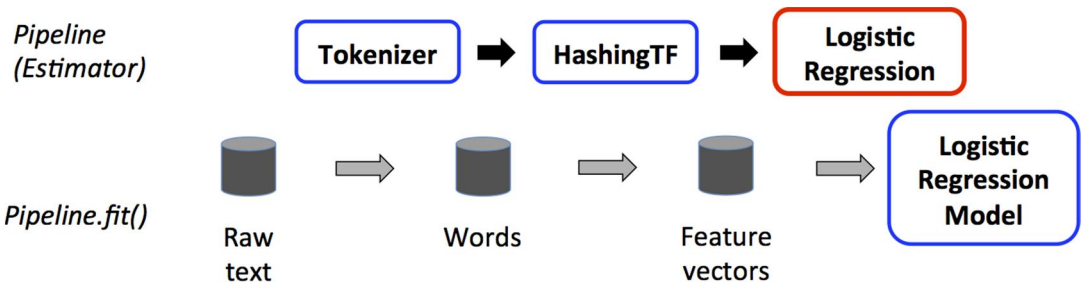


Figure 5.18: Spark pipeline.

Régression logistique : Nous utilisons la fonction **LogisticRegression()** pour réaliser la régression logistique sur l'échantillon d'apprentissage; la fonction **CrossValidator()** permet d'entrainer le modèle avec différents paramètres et de prendre les meilleurs. La fonction Transform() permet ensuite de produire les prédictions pour l'échantillon test.

```
tokenizer = Tokenizer(inputCol="Summary", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])
```

Figure 5.19: Pipeline Régression logistique.

```
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from  pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(labelCol='label',
                                              predictionCol='prediction', metricName='f1')
paramGrid = ParamGridBuilder().addGrid(lr.maxIter, [5, 10, 20]).addGrid(lr.regParam, [0.2, 0.1, 0.01, 0.001]).build()
numFolds=2

crossval = CrossValidator(
    estimator=pipeline,
    estimatorParamMaps=paramGrid,
    evaluator=evaluator,
    numFolds=numFolds)

model = crossval.fit(trainset)
predictionsDf = model.transform(testset)
```

Figure 5.20: Crossvalidator.

Pour le classifieur Naïve bayes, nous utilisons la fonction **NaiveBayes()**, avec le model "multinomial" qui est typiquement utilisé pour la classification des documents.

```
from pyspark.ml.classification import NaiveBayes

hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
nb = NaiveBayes(smoothing=1.0, modelType="multinomial")
pipeline3 = Pipeline(stages=[tokenizer, hashingTF, nb])

model3 = pipeline3.fit(trainset)
nbp = model3.transform(testset)
```

Figure 5.21: Pipeline Naïve bayes.

Concernant les SVM, nous utilisons la fonction **SVMWithSGD(),** avec la fonction de perte **"hinge loss"**.

```
from pyspark.mllib.classification import SVMWithSGD, SVMModel
from pyspark.mllib.regression import LabeledPoint


hashingTFa = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
#svm = SVMWithSGD()
pipeline5 = Pipeline(stages=[tokenizer, hashingTFa])

model5 = pipeline5.fit(trainset)
dts = model5.transform(trainset)

datasss = dts.rdd.map(lambda s: (LabeledPoint(s[0], s[3])))
modelsv = SVMWithSGD.train(datasss, iterations=50)
hashingTFa = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
pipeline5 = Pipeline(stages=[tokenizer, hashingTFa])

model6 = pipeline5.fit(testset)
dtss = model6.transform(testset)

labelsAndPredss = dtss.map(lambda p: (p[0], modelsv.predict(p[3])))
```

Figure 5.22: Pipeline SVM.

Un exemple de la prédiction fournit par un modèle :

	label	Summary	words	TFfeatures	features	rawPrediction	probability	prediction
0	1	"Delight" says it all	["delight", says, it, all]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-1.91549005427, 1.91549005427]	[0.128365328458, 0.871634671542]	1
1	1	Nice Taffy	[nice, taffy]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-2.32668503246, 2.32668503246]	[0.0889368997914, 0.911063100209]	1
2	1	Great! Just as good as the expensive brands!	[great!, , just, as, good, as, the, expensive,...	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(4.27859555166, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-3.27173272848, 3.27173272848]	[0.0365537567445, 0.963446243256]	1
3	1	Yay Barley	[yay, barley]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-3.02706254215, 3.02706254215]	[0.0462181436655, 0.953781856335]	1
4	1	The Best Hot Sauce in the World	[the, best, hot, sauce, in, the, world]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-3.15594510255, 3.15594510255]	[0.0408576626543, 0.959142337346]	1
1		Oatmeal For Oatmeal Lovers	[oatmeal, for, oatmeal, lovers]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-3.74467893345, 3.74467893345]	[0.0230971284134, 0.976902871587]	1	
1		Good Hot Breakfast	[good, hot, breakfast]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-3.42067341745, 3.42067341745]	[0.031655579263, 0.968344420737]	1	
1		good	[good]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[-2.90991684819, 2.90991684819]	[0.0516655094357, 0.948334490564]	1	
1		Mushy	[mushy]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.273394885248, -0.273394885248]	[0.567926153753, 0.432073846247]	0	

Figure 5.23: Prédictions d'un modèle.

5.7.1 Evaluation

Pour évaluer les modèles, nous avons utilisé les mésures suivantes: le rappel, la précision, la f-mesure, l'entropie et la matrice de confusion.

$$Rappel_{\ i} = \frac{Nb\ de\ documents\ correctement\ attribués\ à\ la\ classe\ i}{nb\ de\ documents\ appartenant\ à\ la\ classe\ i}$$

$$Précision_{\ i} = \frac{Nb\ de\ documents\ correctement\ attribués\ à\ la\ classe\ i}{nb\ de\ documents\ attribués\ à\ la\ classe\ i}$$

$$F - mesure = 2\ .\frac{(précision\ .rappel)}{(précision + \ rappel)}$$

$$Entropie = \ -LOG(Précison)$$

	Décision Positifs	Décision Negatifs
Étiquette Positifs	**Vrai Positifs, TP**	**Faux Négatifs, FN**
Étiquette Négatifs	**Faux positifs, FP**	**Vrai Négatifs, TN**

Figure 5.24: Matrice de confusion.

Exemple d'évaluation d'un modèle :

```
numSuccesses = cvPrediction.where("""(prediction = 0 AND label = 0) OR
                                     (prediction = 1 AND label = 1)""").count()
numInspections = cvPrediction.count()
print "There were", numInspections, "inspections and there were", numSuccesses, "successful predictions"
print evaluator1.getMetricName(), ':', evaluator1.evaluate(cvPrediction)
evaluator1.setMetricName('recall')
print evaluator1.getMetricName(), ':', evaluator1.evaluate(cvPrediction)
evaluator1.setMetricName('precision')
print evaluator1.getMetricName(), ':', evaluator1.evaluate(cvPrediction)



failSuccess = cvPrediction.where("prediction = 0 AND label = 0").count()
failFailure = cvPrediction.where("prediction = 0 AND label <> 0").count()
passSuccess = cvPrediction.where("prediction = 1 AND label <> 0").count()
passFailure = cvPrediction.where("prediction = 1 AND label = 0").count()
labels = ['True positive', 'False positive', 'True negative', 'False negative']
sizes = [failSuccess, failFailure, passSuccess, passFailure]
colors = ['turquoise', 'seagreen', 'mediumslateblue', 'palegreen', 'coral']
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors)
plt.axis('equal')
```

```
There were 227093 inspections and there were 204478 successful predictions
f1 : 0.884480317937
recall : 0.900415248378
precision : 0.900415248378
```

Figure 5.25: Evaluation d'un modèle.

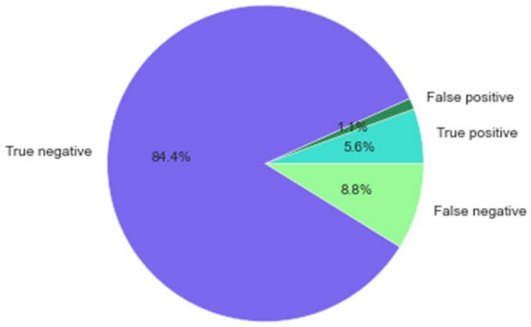


Figure 5.26: Matrice de confusion par pourcentage.

```
Nacc = (float(negp) / float(negj))
Pacc = (float(posp) / float(posj))
Npre = (float(negp) / float(negc))
Ppre = (float(posp) / float(posc))

print "NegativeP :  ",negc
print "PositiveP :  ",posc

print "NegativeEG :  ",negp
print "PositiveEG :  ",posp

#print "Negative :  ",negj
#print "Positive :  ",posj

print "Negative accuracy :  ",Nacc
print "Positive accuracy :  ",Pacc
print "Negative precision :  ",Npre
print "Positive precision :  ",Ppre
```

```
NegativeP :   15309
PositiveP :   211784
NegativeEG :   12725
PositiveEG :   191753
Negative accuracy :   0.388478446697
Positive accuracy :   0.986703509882
Negative precision :   0.831210399112
Positive precision :   0.905417784157
```

Figure 5.27: Rappel et précision pour chaque classe.

5.8 Résultats et discussions

Avec toutes les expérimentations que nous avons réalisé en utilisant les trois classifieurs (Regression Logistique « LR », Naive Bayes « NB », SVM), nous avons obtenu les résultats suivants :

1. Résultats obtenus en utilisant la technique de représentation sac de mots (NP: le nombre des documents bien classés pour la classe ”0”, PP: le nombre des documents bien classés pour la classe ”1”, N.rappel : le rappel pour la classe ”0”, P.rappel : le rappel pour la classe ”1”) :

	Sac de mot + (TF)			Sac de mot + (tf.IDF)		
	LR	NB	SVM	LR	NB	SVM
NP	19115	1059	4594	12725	**24978**	4594
PP	188254	192131	192460	191753	171979	192460
N.Rappel	0,58355	0,32336	0,14024	0,38847	**0,76254**	0,14024
P.Rappel	0,96869	0,98864	0,99034	0,98670	0,88495	0,99034
N.Precision	0,75859	0,82762	0,70993	0,83121	0,52767	0,70993
P.Precision	0,93243	0,89657	0,87235	0,90541	**0,95673**	0,87235
N.F1	**0,65966**	0,46503	0,23421	0,52948	0,62373	0,23421
P.F1	**0,95021**	0,94036	0,92761	0,94431	0,91944	0,92761
Rappel	**0,91314**	0,89268	0,86772	0,90041	0,86729	0,86772
Precision	**0,91314**	0,89268	0,86772	0,90041	0,86729	0,86772
F1	**0,90831**	0,87179	0,82759	0,88448	0,87678	0,82759
Entropie	**0,039**	0,049	0,062	0,046	0,062	0,062

Tableau 5.1: Résultats des 3 classifieurs avec sac de mots.

2. Résultats obtenus en utilisant la technique de représentation sac de mots avec racinisation :

	Sac de mot + Racinisation (TF)			Sac de mot + Racinisation (tf.IDF)		
	LR	NB	SVM	LR	NB	SVM
NP	17205	8484	4877	17205	23344	4877
PP	188505	**192989**	192559	188505	175098	192559
N.Rappel	0,52524	0,25900	0,14888	0,52524	0,71266	0,14888
P.Rappel	0,96999	**0,99306**	0,99085	0,96999	0,90100	0,99085
N.Precision	0,74684	**0,86289**	0,73283	0,74684	0,54819	0,73283
P.Precision	0,92379	0,88828	0,87352	0,92379	0,94898	0,87352
N.F1	0,61674	0,39841	0,24748	0,61674	0,61970	0,24748

P.F1	0,94633	0,93775	0,92849	0,94633	0,92437	0,92849
Rappel	0,90584	0,88718	0,8694	0,90584	0,87383	0,8694
Precision	0,90584	0,88718	0,8694	0,90584	0,87383	0,8694
F1	0,89878	0,85996	0,83027	0,89878	0,88042	0,83027
Entropie	**0,043**	0,052	0,061	0,043	0,059	0,061

Tableau 5.2 Résultats des 3 classifieurs avec sac de mot et racinisation

3. Résultats obtenus en utilisant la technique de représentation sac de mots avec lemmatisation :

	Sac de mot + Lemmatisation (TF)			Sac de mot + Lemmatisation (tf.IDF)		
	LR	NB	SVM	LR	NB	SVM
NP	17880	9066	4952	17880	23631	4952
PP	188454	192852	192492	188454	174725	192492
N.Rappel	0,54585	0,27677	0,15117	0,54585	0,72142	0,15117
P.Rappel	0,96972	0,99235	0,99050	0,96972	0,89908	0,99050
N.Precision	0,75243	0,85925	0,72855	0,75243	0,54646	0,72855
P.Precision	0,92683	0,89059	0,87378	0,92683	0,95036	0,87378
N.F1	0,63270	0,41868	0,25039	0,63270	0,62187	0,25039
P.F1	0,94779	0,93872	0,92849	0,94779	0,92401	0,92849
Rappel	0,90858	0,88914	0,86944	0,90858	0,87345	0,86944
Precision	0,90858	0,88914	0,86944	0,90858	0,87345	0,86944
F1	0,90234	0,86371	0,83068	0,90234	0,88043	0,83068
Entropie	0,042	0,051	0,061	0,042	0,059	0,061

Tableau 5.3 Résultats des 3 classifieurs avec sac de mot et lemmatisation

4. Résultats obtenus en utilisant la technique de représentation ngram :

	NGRAM + (TF)			NGRAM + (tf.IDF)		
	LR	NB	SVM	LR	NB	SVM
NP	17609	14736	13443	17609	**32756**	13443
PP	187774	187536	168452	187774	**194337**	168452
N.Rappel	0,53758	0,44987	0,41039	0,53758	0,68219	0,41039
P.Rappel	0,96622	0,965	0,8668	0,96622	0,86767	0,8668
N.Precision	0,72848	0,68421	0,34181	0,72848	0,46495	0,34181
P.Precision	0,92535	0,91233	0,89714	0,92535	0,94185	0,89714
N.F1	0,61864	0,54283	0,37297	0,61864	0,55300	0,37297

P.F1	0,94534	0,93793	0,88171	0,94534	0,90324	0,88171
Rappel	0,9044	0,8907	0,80097	0,9044	0,84092	0,80097
Precision	0,9044	0,8907	0,80097	0,9044	0,84092	0,80097
F1	0,89822	0,88094	0,80833	0,89822	0,85272	0,80833
Entropie	0,044	0,050	0,096	0,044	0,075	0,096

Tableau 5.4 Résultats des 3 classifieurs avec ngram.

5. Résultats obtenus en utilisant la technique de représentation ngram avec racinisation :

	NGRAM + Racinisation (TF)			NGRAM + Racinisation (tf.IDF)		
	LR	NB	SVM	LR	NB	SVM
NP	17978	22707	18071	17978	13789	18071
PP	187956	169598	169239	187956	188799	169239
N.Rappel	0,54884	**0,69321**	0,55168	0,54884	0,42096	0,55168
P.Rappel	0,96716	0,8727	0,87085	0,96716	**0,9715**	0,87085
N.Precision	0,73804	0,47858	0,41861	**0,73804**	0,71345	0,41861
P.Precision	0,9271	**0,94406**	0,92015	0,9271	0,9087	0,92015
N.F1	**0,62953**	0,56624	0,47602	0,62953	0,52950	0,47602
P.F1	**0,94671**	0,90698	0,89482	0,94671	0,93905	0,89482
Rappel	**0,90682**	0,84681	0,82481	0,90682	0,89209	0,82481
Precision	**0,90682**	0,84681	0,82481	0,90682	0,89209	0,82481
F1	**0,90096**	0,85783	0,83441	0,90096	0,87998	0,83441
Entropie	0,042	0,072	0,084	0,042	0,050	0,084

Tableau 5.5 Résultats des 3 classifieurs avec ngram et racinisation

6. Résultats obtenus en utilisant la technique de représentation ngram avec tokenisation et lemmatisation :

	NGRAM + Lemmatisation (TF)			NGRAM + Lemmatisation (tf.IDF)		
	LR	NB	SVM	LR	NB	SVM
NP	17806	13890	15998	17806	22722	15998
PP	188016	188535	164363	188016	169210	164363
N.Rappel	0,54359	0,42404	0,48839	0,54359	0,69367	0,48839
P.Rappel	0,96747	0,97014	0,84576	0,96747	0,8707	0,84576
N.Precision	0,73801	0,70536	0,34799	0,73801	0,47486	0,34799
P.Precision	0,92634	0,90903	0,90747	0,92634	0,94402	0,90747

N.F1	0,62605	0,52966	0,40641	0,62605	0,56378	0,40641
P.F1	0,94646	0,93859	0,87553	0,94646	0,90588	0,87553
Rappel	0,90633	0,89137	0,79421	0,90633	0,84516	0,79421
Precision	0,90633	0,89137	0,79421	0,90633	0,84516	0,79421
F1	0,90024	0,87961	0,80786	0,90024	0,85653	0,80786
Entropie	0,043	0,050	0,100	0,043	0,073	0,100

Tableau 5.6 Résultats des 3 classifieurs avec ngram et lemmatisation

Comme il est montré dans la figure 5.28, le rappel du modèle « régression logistique » est généralement constant (90,6 %) et légèrement surélevé (91,3%) en représentation par sac de mot avec tokenisation et TF. Pour la classe « Positive » le rappel est presque le même (96%) pour toutes les représentations, avec une petite augmentation (de 2%) dans le cas de la représentation sac de mot avec tokenisation et tf.IDF contrairement à celui de la classe « négative » où il est petit par rapport aux autres représentations dont les rappels sont presque stables à 52-54%. Tandis que pour le cas de la représentation sac de mot avec tokenisation et TF le rappel pour la classe négative est élevé (58%). Donc pour l'algorithme « régression logistique » le meilleur rappel est celui de la représentation sac de mot avec TF, notant que les autres représentations donnent des résultats inférieurs.

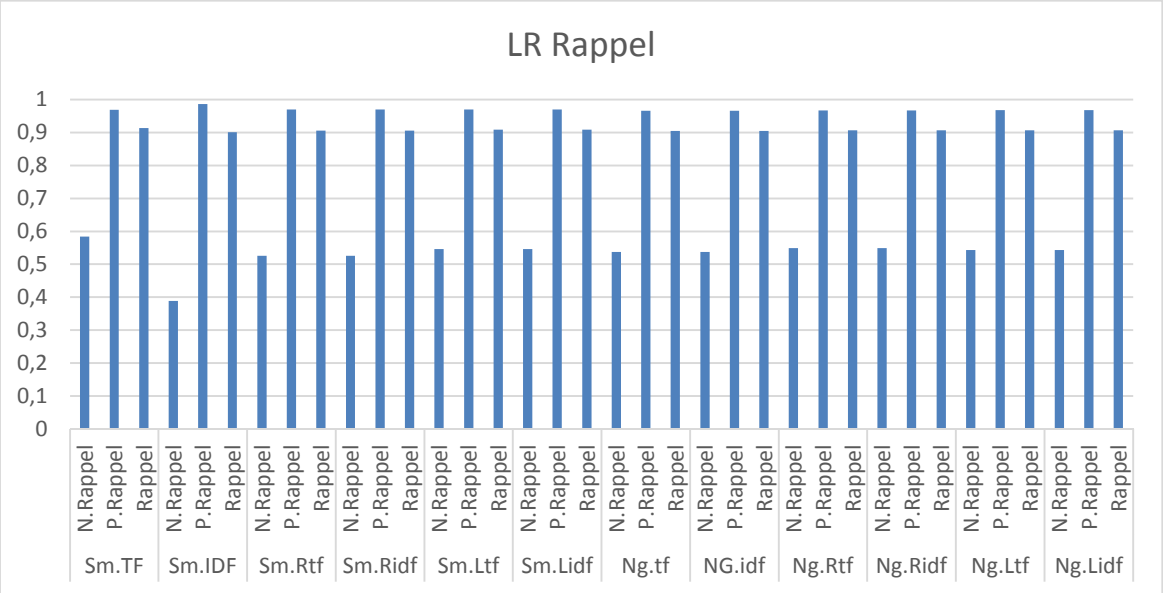


Figure 5.28: Le rappel de l'algorithme LR.

Pour l'algorithme « naïve bayes » le rappel se varie entre 84% et 89%, avec six modèles donnent un rappel de 89% mais si on s'intéresse par le rappel positif/négatif, le modèle en utilisant la représentation sac de mot avec tokenisation et tf.idf comme pondération a fait ses preuves pour la classe « négative » avec un rappel de (76,2%) et les deux modèles (sac de mot avec racinisation et tf, sac de mot avec lemmatisation et tf) pour la classe positive avec un rappel de (99,3%). Le modèle utilisant la représentation sac de mot avec tokenisation et tf.idf présente le meilleur rappel moyen.

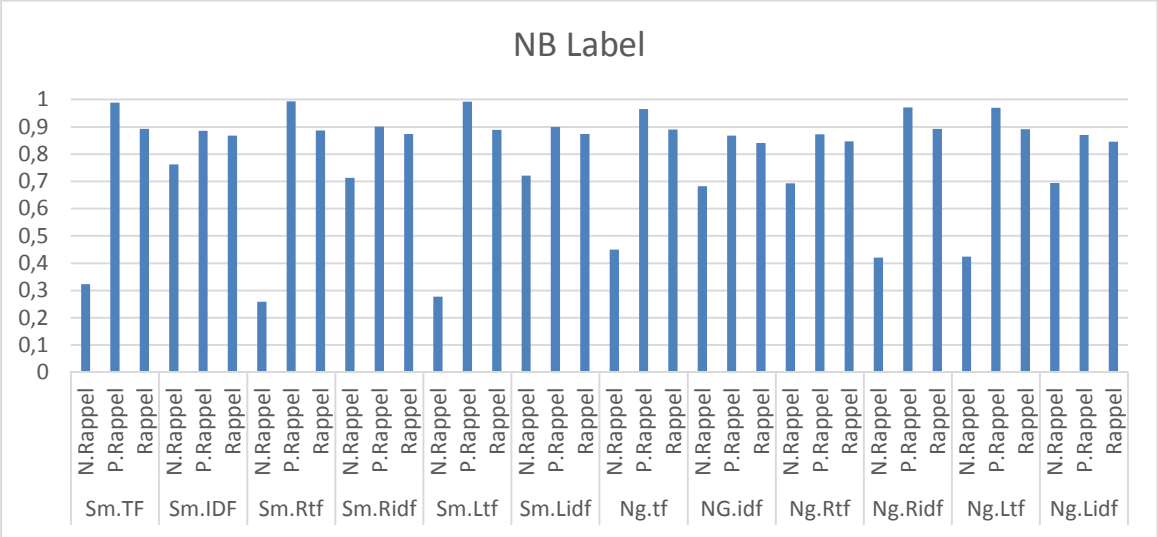


Figure 5.29: Le rappel de l'algorithme NB.

Le rappel de l'algorithme SVM est presque le même pour touts les modèles utilisant la représentation sac de mot (87%), et il diminue pour les autres modèles utilisant la représentation ngram. Par contre le rappel pour la classe négative est très élevé (41-55%) dans les modèles avec représentation ngram « surtout avec racinisation (55%) » par rapport aux représentation sac de mot (15%). Le rappel de la classe positive est presque parfait (99%) pour les représentations avec sac de mots, et moyen pour les représentations ngram.

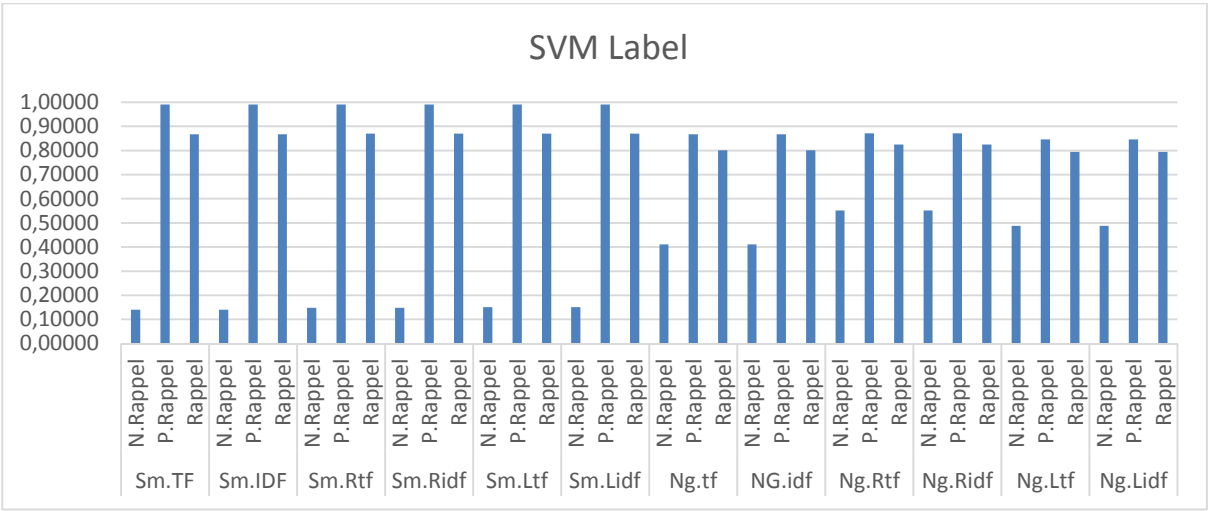


Figure 5.30: Le rappel de l'algorithme SVM.

La précision globale ainsi que la précision négative de l'algorithme « régression logistique » sont presque constantes pour touts les modèles avec une légère augmentation pour la représentation par sac de mot avec tokenisation et TF. Le modèle sac de mot avec tf.idf a la meilleure précision négative (83%) comparant aux autres modèles où elle est presque stable et aussi la meilleure précision moyenne.

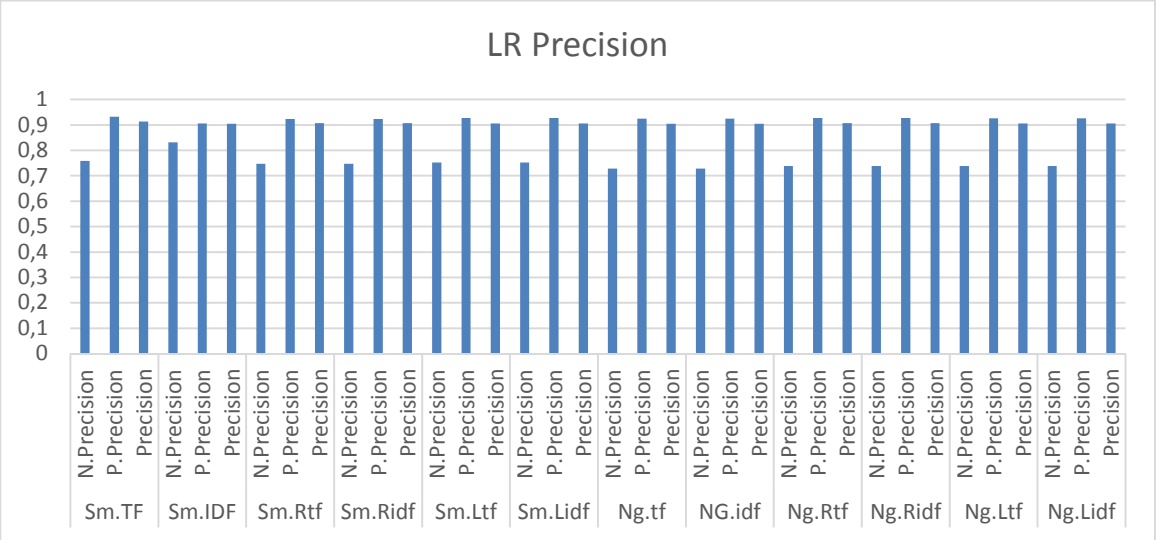


Figure 5.31: La précision de l'algorithme LR.

La précision négative pour l'algorithme naïve bayes a atteint ses meilleurs résultats en représentation sac de mot avec racinisation et tf, et connue des bas et des hauts dans les autres modèles. Le modèle avec représentation sac de mot avec tokenisation et tf.IDF a la meilleure précision positive. Et une précision globale qui se varie entre 84% et 89% pour cet algorithme.

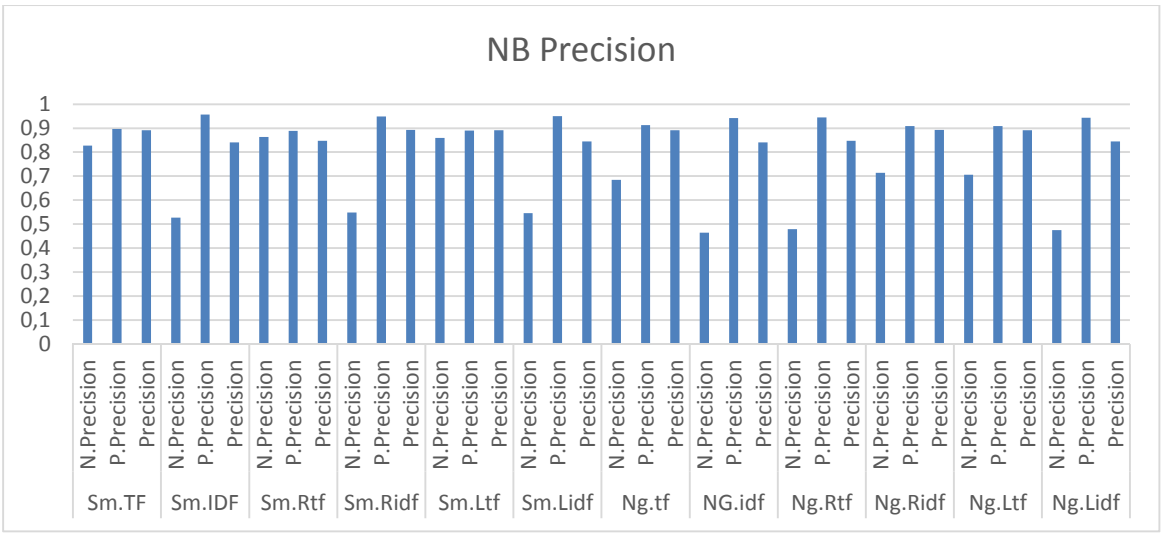


Figure 5.32: La précision de l'algorithme NB.

La précision positive pour l'algorithme SVM est presque stable pour les modèles avec représentation sac de mot (87%) et un peu élevé avec la représentation Ngram (90%-92%). Et le contraire est vrai pour la précision négative. La précision globale est presque constante dans touts les modèles.

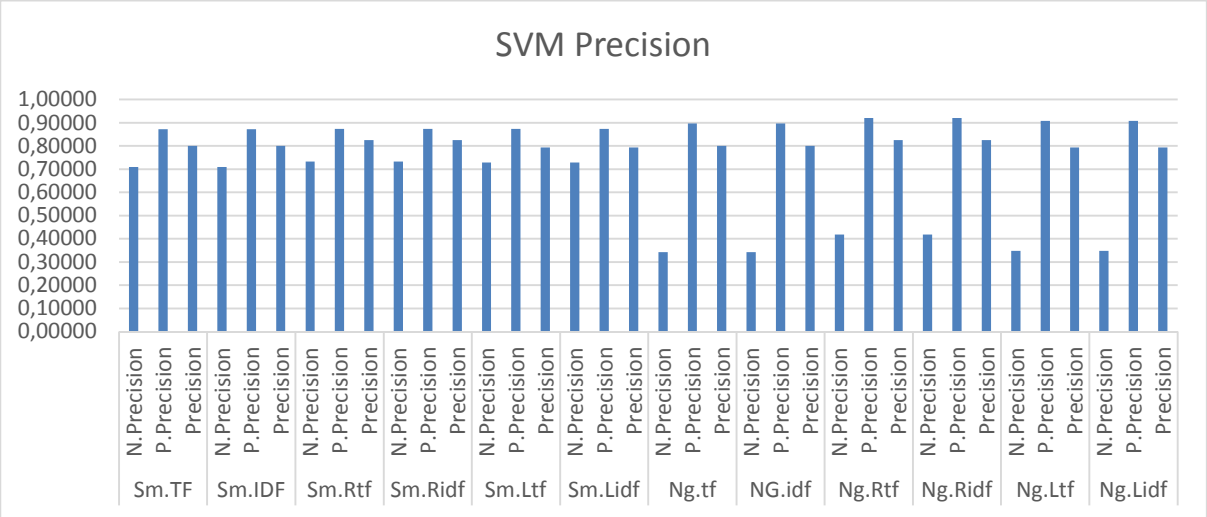


Figure 5.33: La précision de l'algorithme SVM.

Le modèle de l'algorithme régression logistique en utilisant la représentation sac de mot avec tokenisation et Tf est sans doute le meilleur. Donc la racinisation et la lemmatisation et même la représentation n-gram n'ont pas servi à améliorer la performance de l'algorithme régression logistique.

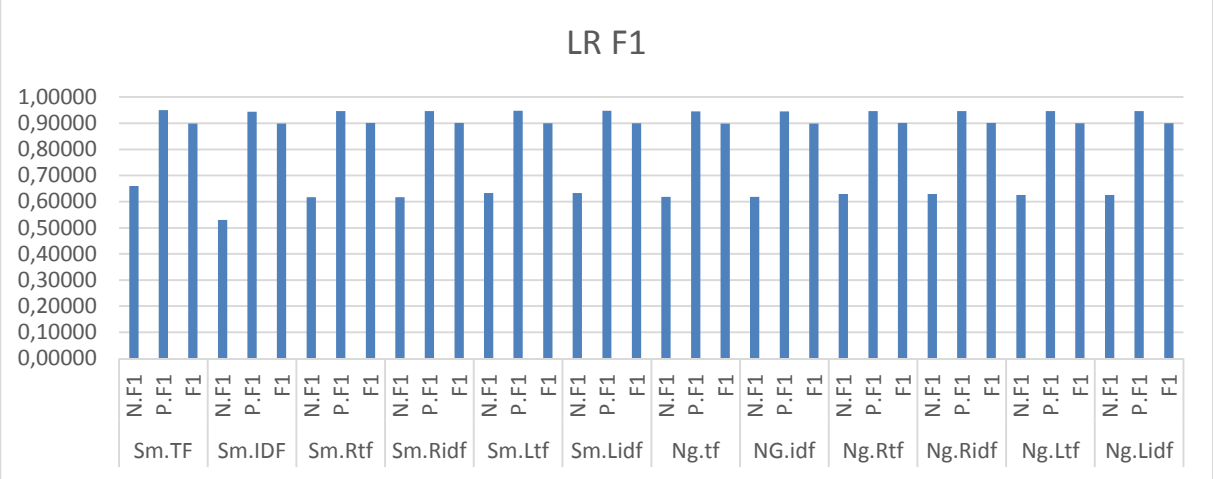


Figure 5.34: F-mesure de l'algorithme LR.

Les meilleurs modèles de l'algorithme Naïve bayes sont ceux où nous avons utilisé la racinisation ou lemmatisation avec la pondération tf.Idf dans le cas des deux représentations.

Donc les techniques de prétraitements lemmatisation et racinisation jouent un rôle pour l'amélioration des performances de l'algorithme Naïve Bayes.

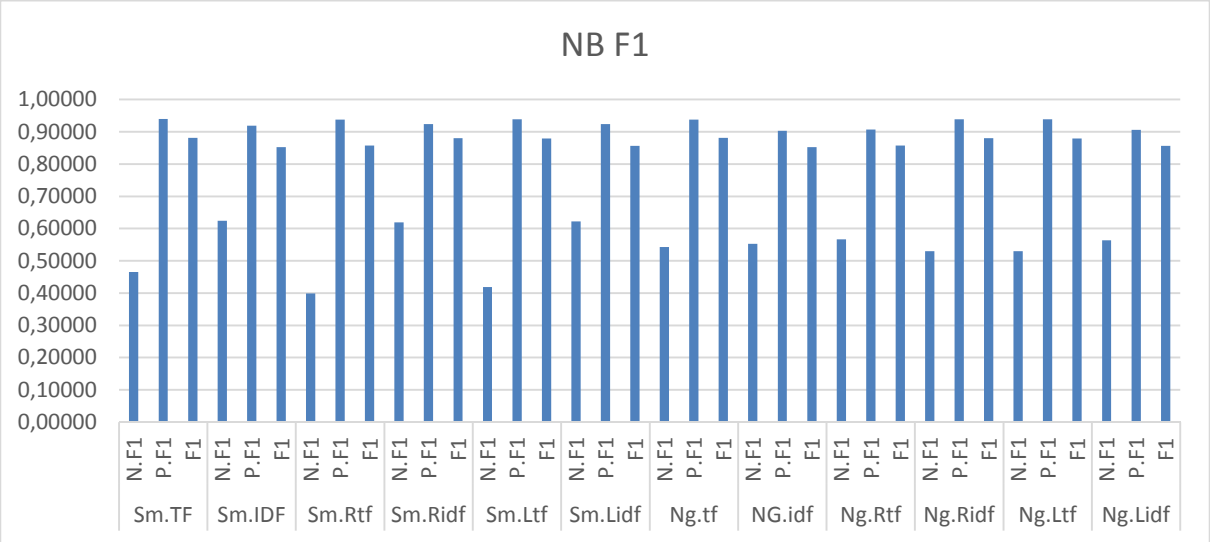


Figure 5.35: F-mesure de l'algorithme NB.

Pour l'algorithme SVM nous notons que la représentation ngram a soulevé la f-mesure par rapport à la représentation sac de mot. Et la meilleure performance par un modèle est obtenu en utilisant la représentation Ngram avec racinisation.

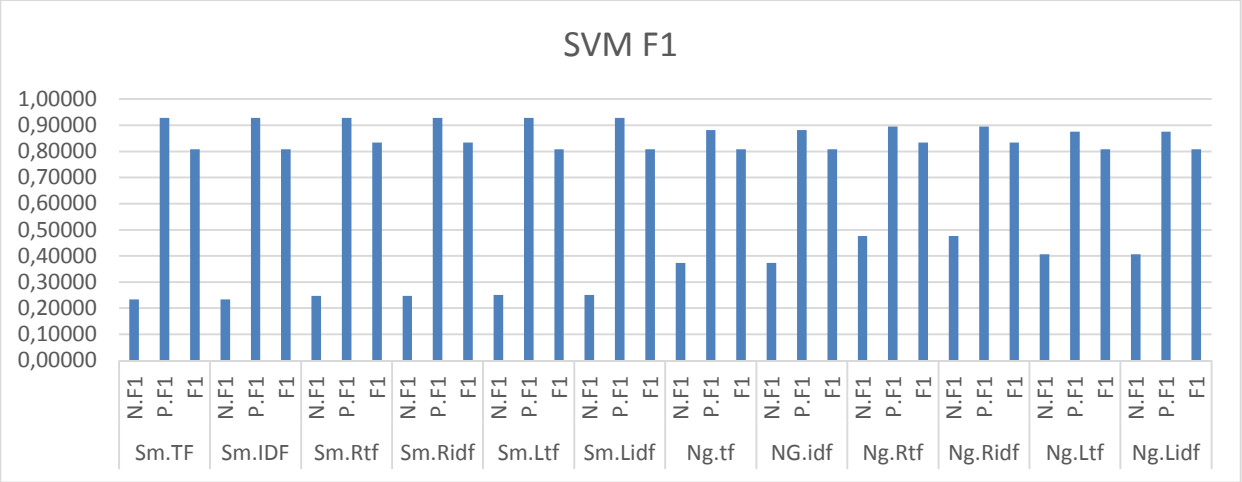


Figure 5.36: F-mesure de l'algorithme SVM.

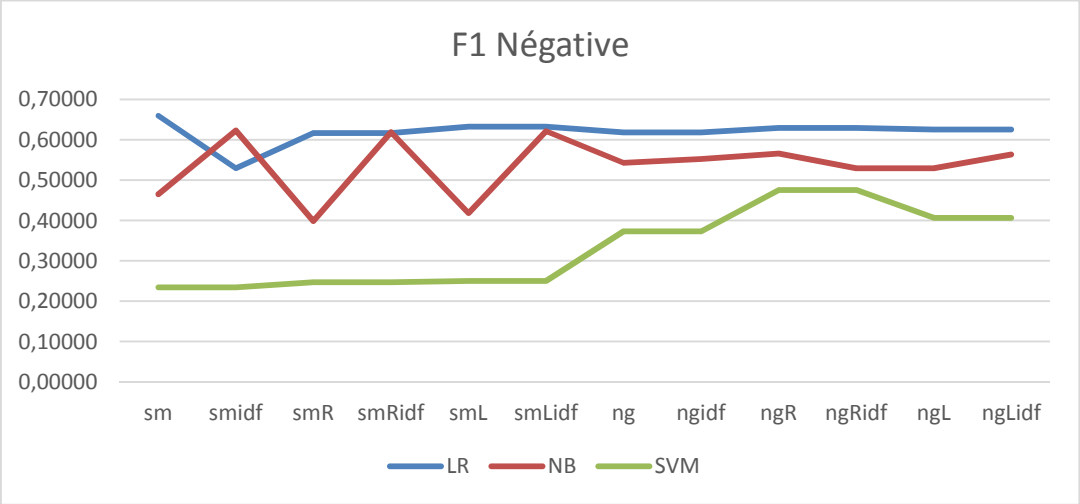


Figure 5.37: Comparaison de f-mesures de la classe « 0 » des trois classifieurs.

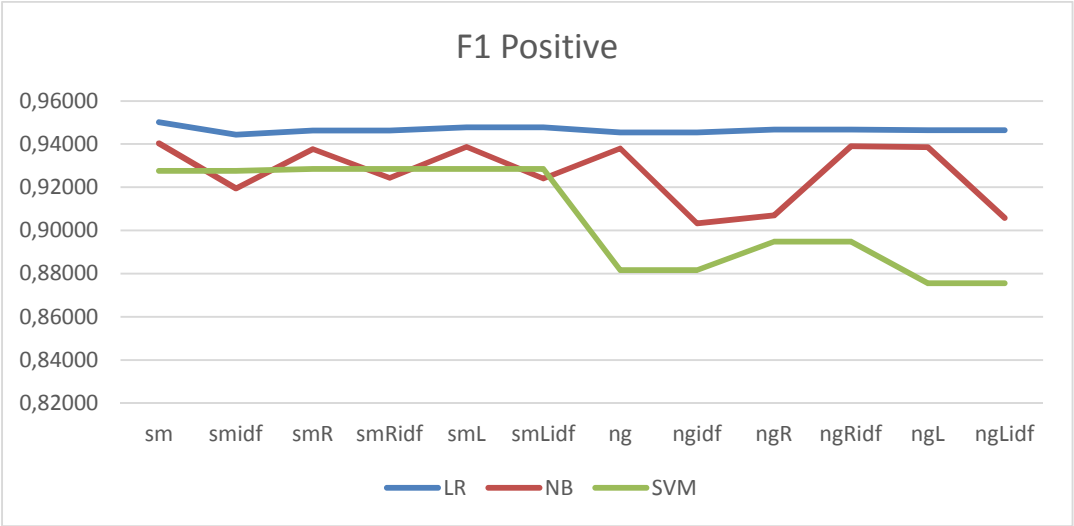


Figure 5.38: Comparaison de f-mesures de la classe « 1 » des trois classifieurs.

Notant que F1 moyenne = (F.positive + F.négative) / 2, nous obtenons le tableau[1] suivant :

	SM	SMidf	SMR	SMRidf	SML	SMLidf	NG	NGidf	NGR	NGRidf	NGL	NGLidf
LR	0,80	0,74	0,78	0,78	0,79	0,79	0,78	0,78	0,79	0,79	0,79	0,79
NB	**0,70**	0,77	0,67	0,77	0,68	0,77	0,74	0,73	0,74	0,73	0,73	0,73
SVM	0,58	0,58	0,59	0,59	0,59	0,59	0,63	0,63	0,69	0,69	0,64	0,64

Tableau 5.7: F-mesures moyennes.

[1] La 1ere ligne du tableau présente les représentation et techniques abréviées (SM : sac de mot, R : racinsination, L : lemmatisation)

L'algorithme LR a les meilleurs résultats dans presque touts les représentations, sauf dans la représentation sac de mot avec tf.idf où l'algorithme NB a pu le dépasser. Nous voyons aussi que l'algorithme NB, dans les représentations sac de mot avec racinisation et lemmatisation avec tf.idf, se rapproche des résultats de l'algorithme LR. Concernant l'algorithme SVM nous voyons que ses résultats, surtout avec la représentation sac de mot, sont très loin des résultats des autres algorithmes.

La représentation sac de mot dans notre cas est la meilleure représentation, avec laquelle les algorithmes LR et NB ont donné les meilleurs résultats sauf pour l'algorithme SVM où les performances de ce dernier ont été élevé avec la représentation Ngram.

La tokenisation seulement, comme technique de prétraitement, a pu donner des résultats élevés avec deux algorithmes LR et NB. Et pour la racinisation, cette technique a augmenté les performances de l'algorithme NB avec représentation sac de mot, et l'algorithme SVM avec représentation Ngram. La lemmatisation à son tour a pu donner des bons résultats juste avec l'algorithme LR et NB avec représentation sac de mot.

La technique de pondération TF a donné les meilleurs résultats avec l'algorithme LR, contrairement pour le cas de l'algorithme NB où la technique tf.idf a donné les meilleurs résultats. Pour le cas du SVM l'utilisation de l'un ou l'autre ne change rien sur les résultats.

Finalement nous pouvons dire que l'algorithme LR a fait ses preuves avec presque touts les représentations et surtout avec la représentation sac de mot avec tokenisation seulement et Tf.

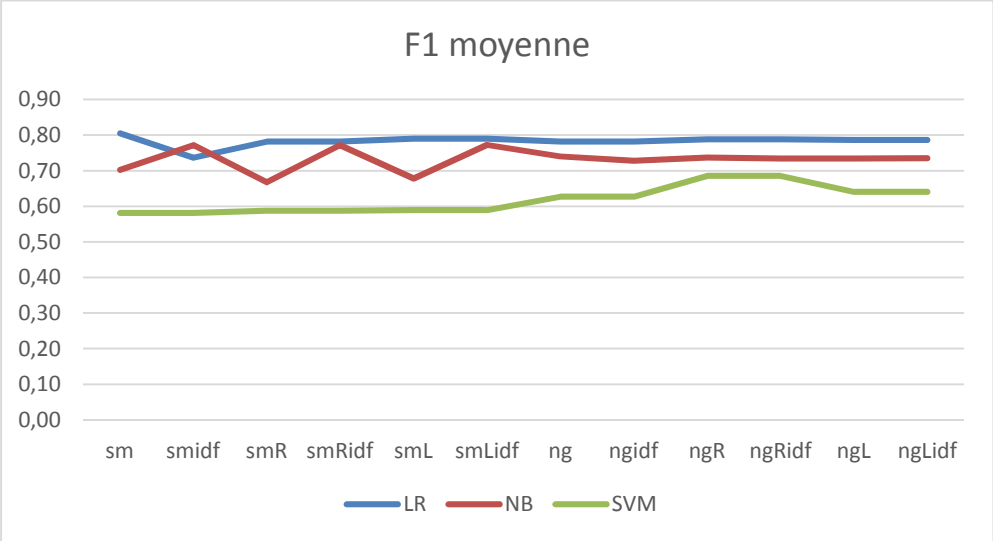


Figure 5.39: Comparaison de f-mesures des trois classifieurs.

5.9 Conclusion

Nous avons remarqué que le classifieur "Régression Logistique" donne les meilleurs résultats pour les deux classes. En seconde position, il vient le classifieur "Naive Bayes" et en dernier lieu le classifieur "SVM". Nous avons observé que le classifieur "Régression

Logistique" donne de bons résultats avec toutes les représentations utilisées. Cela démontre que la taille du vecteur d'entrainement n'a pas d'influence sur les résultats de ce classifieur. Il en résulte que ce classifieur s'adapte bien avec toutes les représentations. Les résultats obtenus avec le classifieur "Naive Bayes" et la représentation sac de mot fluctuent entre un minimum (correspondant à la pondération TF) et un maximum (correspondant à la pondération Tf.IDF), et se stabilisent avec la représentation N-gram. Pour le cas du classifieur "SVM" les résultats sont mauvais avec la représentation sac de mot et s'améliorent avec la représentation ngram pour la classe "négatif", et l'inverse pour la classe "positif". Les résultats des classifieurs "Naive Bayes" et "SVM" dépendent du type de représentation.

Conclusion et perspectives

6.1 Conclusion

Le sujet de recherche de ce mémoire est l'analyse des sentiments effectuée sur des données massives (Big Data). Pour réaliser ce travail nous avons utilisé un outil dédié aux big data appelé "Apache SPARK". Il s'avère être un outil très performant permettant la réalisation d'une étude comparative entre plusieurs classifieurs appliqués sur un corpus volumineux d'Amazon.

Nous avons comparé la performance des trois classifieurs avec un apprentissage supervisé: régression logistique, naïve bayes et SVM, avec deux méthodes de représentation: Sac de mot et Ngram, et trois techniques de prétraitement à savoir la tokenisation, la racinisation et la lemmatisation, afin d'extraire les sentiments à partir des avis des clients sur les aliments du site Amazon.

Cette étude montre que les deux premiers classifieurs donnent de très bons résultats, et avec un degré moins pour le classifieur SVM.

Le classifieur "Régression Logistique" est très performant avec toutes les représentations utilisées. Le classifieur "Naive Bayes" donne de très bon résultats surtout avec la représentation sac de mot et en particulier avec la pondération Tf.idf. Pour le cas du classifieur "SVM" nous avons remarqué qu'il présente un défaut avec la représentation sac de mot, et ses performances s'améliorent avec la représentation N-gram et atteignent le maximum avec la technique de racinisation.

Si nous regardons de plus près les résultats, nous constatons que le classifieur "LR" conduit aux meilleurs résultats pour les deux classes à la fois. Et pour le classifieur "SVM" il mène aux mauvais résultats pour les deux classes.

6.2 Perspectives

Nous estimons que les objectifs de ce mémoire ont été pleinement atteints. Afin d'améliorer les prédictions il nous reste beaucoup de questions à régler qui demandent des analyses beaucoup plus profondes parmi lesquelles le problème de construction d'une liste de mots vides plus adéquate à l'analyse des sentiments ainsi que le problème de valorisation des adjectifs négatifs et positifs pour les mettre en exergue.

Références bibliographique

[Beranger, 2015].	F. Beranger « BIG DATA Collecte et valorsiation de masses de données ». Livre blanc Smile, 2015.
[Bruchez, 2015]	R. Bruchez. « Les bases de données NoSQL et le Big Data ». 2eme édition, Avril 2015.
[Buc&al 1992]	C. Buckley, G. Salton and J. Allan. Automatic Retrieval with Locality Information using Smart. Proceedings of the First Text Retrieval Conference. Gaithersburg, 1992.
[Cavnar & Trenkle, 1994]	Cavnar, W. et J. Trenkle. N-gram-based text categorization. In Proceedings of SDAIR- 94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US, 1994.
[Cointot & Eychenne, 2014]	Cointot, J. & Eychenne, Y. La révolution du big data. Livre de Dunod, 2014.
[Das & Chen, 2001]	Das, S., & Chen, M. Yahoo! for Amazon: Extracting Market Sentiment from Stock Message Boards. In: Proceedings of the Asia Pacific Finance Association Annual Conference (APFA), 2001.
[Dave & al. 2003]	Dave, K., Lawrence, S., & M., Pennock D. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews, 2003.
[Jaillet & all, 2003]	S.Jaillet, M.Teisseire, J.Chauche, V.Prince. « Classification automatique de documents : Le coefficient des deux écarts », 2003.
[Jalam & Teytaud, 2001]	Jalam, R. et O. Teytaud (2001). Identification de la langue et catégorisation de textes basées sur les n-grammes. In H. Briand et F. Guillet (Eds.), EGC, Volume 1 of Extraction des Connaissances et Apprentissage, 2001.
[Joachims, 1998]	T.Joachims « Transductive inference for text classification using support vector machines », 1998.

[Jolia-Ferrer, 2014]	L. Jolia-Ferrer. « Big data concept et mise en ouevre de Hadoop», 2014
[Gil, 2013]	G.Press « A very short history about big data », Article sur Forbes.com, 09/05/2013.
[Lavalley & all, 2009]	R. Lavalley, P. Bellot, M. El-Bèze « Interactions entre le calcul [Lee 1995] J. H. Lee. Combining Multiple Evidence from Different Properties of Weighting Schemes. Proceedings of the 18th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 1995). Washington, USA, 1995.
[Liu, 2006]	Liu, B. Web data mining; Exploring hyperlinks, contents, and usage data. Springer. Chap. 11: Opinion Mining, 2006.
[Monino & Sedkaoui, 2016]	J. Monino, S. Sedkaoui. « Big data, Open data et valorisation de données ». Volume 4, 2016.
[Na & Al, 2004]	Na, J.C., Sui, H., Khoo, C., Chan, S., & Zhou, Y. 2004. Effectiveness of Simple Linguistic Processing in Automatic Sentiment Classification of Product Reviews. Pages 49–54 of: Conference of the International Society for Knowledge Organization (ISKO), 2004.
[Náther, 2005]	Náther, P. (2005). N-gram based text categorization, institute of informatics, comenius university, 2005.
[Pang & Al, 2002]	Pang, B., Lee, L., & Vaithyanathan, S. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. Pages 79–86 of : Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002
[Paradis & Nie, 2005]	Paradis, F. et J. Nie (2005). Filtering contents with bigrams and named entities to improve text classification. In AIRS, 2005.
[Robertson & Walker, 1994]	Robertson, S. et S. Walker (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, pp. 232–241. Springer-Verlag New York, Inc, 1994.
[Sal, 1971a]	G. Salton. The SMART Retrieval System. Experiments in Automatic Document Processing. Prentice Hall, Englewood Cliffs. 1971.

[Sal, 1971b]	G. Salton. The Performance of Interactive Information Retrieval. Information Processing Letters. 1971.
[Sal, 1973]	G. Salton. Recent Studies in Automatic Text Analysis and Document Retrieval. ACM Journal. 1973.
[Sal, 1989]	G. Salton. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley. 1989.
[Sal & McG, 1983]	G. Salton and M. McGill. Introduction to Modern Information Retrieval. 1983.
[Sal & Buc, 1988]	G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management. 1998.
[Salton & McGill, 1983]	G.Salton & M.McGill « Introduction to Modern Information Retrieval» , 1983
[Salton & Buckley, 1988]	G.Salton, C.Buckley « Term-weighting approaches in automatic text retrieval », 1988.
[Sin, 1997]	A. Singhal. Term Weighting Revisited. PhD thesis. Department of Computer Science, Cornell University. 1997.
[Tan & al, 2002]	Tan, C., Y. Wang, et C. Lee (2002). The use of bigrams to enhance text categorization. Inf. Process. Manage. 2002.
[Tong, 2001]	Tong, R. M. 2001. An Operational System for Detecting and Tracking Opinions in Online Discussion. In: Proceedings of the Workshop on Operational Text Classification (OTC). 2001.
[Uschold & King, 1995]	M.Uschold et M.King «Towards a Methodology for Building Ontologies », 1995.
[Vapnik, 1995]	V.Vapnik « The Nature of Statistical Learning ». 1995.

Glossaire

NB	Naive Bayes
LR	Régression Logistique
SVM	Support Vector Machine
N.rappel	Le rappel de la classe « négatif »
P.rappel	Le rappel de la classe « positif »
N.précision	La precision de la classe “ négatif ”
P.précision	La precision de la classe “ positif ”
N.F1	La F-mesure de la classe « négatif »
P.F1	La F-mesure de la classe « positif »
F1	La F-mesure
NN	Le nombre des documents bien classés pour la classe ” négatif ”
NP	Le nombre des documents bien classés pour la classe ” positif ”