UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ

INSTITUT DE LA FRANCOPHONIE INTERNATIONALE

Reconnaissance des formes

Option : Systèmes Intelligents et Multimédia (SIM)

Promotion : XXIII

RAPPORT DE TP1 DE RECONNAISSANCE DES FORMES

Classification automatique des documents à l'aide de l'apprentissage et du traitement automatique de langage naturelle

Rédigé par groupe 2 :
ADOUM Okim Boka
CISSE Abdoulaye

Encadrant :
Dr. Ho Tuong Vinh

Année académique : 2019 - 2020

Table des matières

Table des figures

1 Introduction

La recherche accorde ces dernières années, beaucoup d'importance au traitement des données textuelles. Ceci pour plusieurs raisons : un nombre croissant de collections mises en réseau et distribuées au plan international, le développement de l'infrastructure de communication et de l'Internet. Les traitements manuels de ces données s'avèrent très coûteux en temps et en personnel, ils sont peu flexibles et leur généralisation à d'autres domaines est presque impossible ; c'est pour cela que l'on cherche à mettre au point des méthodes automatiques.

Le domaine de la fouille de textes (text mining) s'est développé pour répondre à volonté à la gestion par contenu des sources volumineuses de textes. A l'heure actuelle, de nombreux logiciels de classification de textes sont disponibles, ils ont fait l'objet de publications et leurs champs d'application s'élargit de jour en jour. En général, ces systèmes sont basés sur des algorithmes d'apprentissage automatique (approche statistique, approche syntaxique et approche connexionniste).

Nous nous intéressons ici plus particulièrement aux algorithmes d'apprentissage et nous avons utilisé et comparé deux algorithmes : le Support Vector Machine (SVM) et l'algorithme de Gradient Boosting Decision Tree(GBDT). Pour pouvoir utiliser de tels algorithmes, il est nécessaire de transformer les données, initialement en format texte, en une représentation numérique. Nous avons choisi pour ce faire, la méthode de sélection des termes les plus pertinents. Une fois ce prétraitement terminé, nous pouvons effectuer la classification à l'aide de nos algorithmes ; ensuite nous ferons une étude comparative entre les deux méthodes.

Dans un premier temps nous allons décrire le projet, faire une étude comparative des différentes méthodes , secundo lister les outils qui nous ont permis de réaliser les taches, réaliser une analyse et enfin nous allons élaborer une comparaison entre les différents résultats, tertio la conclusion et les perspectives. Nous mettrons par ailleurs en évidence les difficultés rencontrées dans la réalisation de ce projet.

1.1 Contexte

Notre travail repose sur un contexte de la classification automatique de documents, classification supervisée puisque elle opère à partir d'un ensemble de classes prédéfinies. L'approche proposée est originale car elle repose sur une représentation vectorielle des documents non plus axée sur les mots contenus mais sur une représentation plus sémantique de ceux-ci. Dans cet espace, un coefficient de concordance permet d'obtenir des résultats prometteurs de classification puisque les textes sont classés en fonction de leur proximité conceptuelle.

1.2 Motivation

Les motivations principales de la réalisation de travail sont :

— faciliter la classification des documents aux seins des organisations publiques, privées et non gouvernementales ;

— Appliquer nos connaissances d'apprentissage automatique dans le traitement des données textuelles ;
— Tester quelques algorithmes de traitement automatique du langage naturel (NLP)

1.3 Objectifs

Nous aimerions mettre en place un modèle permettant de catégoriser automatiquement les documents et les trie en classes de documents prédéfinies.

FIGURE *1 – Classification de document*

Elle permet d'organiser des corpus documentaires, de les trier, et d'aider à les exploiter dans des secteurs tels que l'administration, l'aéronautique, la recherche sur internet, les sciences.

1.4 Contribution

Nous nous sommes plus situé sur un contexte de projet scolaire en faisant l'analyse des différents travaux pratiques existants et les résultats des recherches menés ces deux dernières années pour réaliser notre projet. Pour ceux nous envisageons d'autres occasions pour approfondir encore d'avantage nos résultats.

2 Analyse du sujet

Notre sujet porte sur la classification et la catégorisation de documents qui est l'activité du traitement automatique des langues naturelles qui consiste à classer de façon automatique des ressources documentaires, généralement en provenance d'un corpus. Cette classification peut prendre une infinité de formes. On citera ainsi la classification par genre, par thème, ou encore par opinion. La tâche de classification est réalisée avec des algorithmes spécifiques, mis en œuvre par des systèmes de traitement de l'information. C'est une tâche d'automatisation d'un processus de classement, qui fait le plus souvent appel à des méthodes numériques (c'est-à-dire des algorithmes de recherche d'information ou de classification de type mathématique).

L'activité de classification de documents est essentielle dans de nombreux domaines économiques : elle permet d'organiser des corpus documentaires, de les trier, et d'aider à les exploiter dans des secteurs tels que l'administration, l'aéronautique, la recherche sur internet, les sciences. Les données non structurées sous forme de texte sont partout : e-mails, chats, pages Web, médias sociaux, tickets d'assistance, réponses aux sondages,.. Le texte peut être une source extrêmement riche d'informations, mais en extraire des informations peut être difficile et long en raison de sa

nature non structurée. De ce fait, nous avons décider d'appliquer la classification de texte pour structurer le texte de manière rapide et économique pour améliorer la prise de décision et automatiser les processus. A chaque instant t que la structure reçoit un nouveau fichier d'une catégorie quelconque, l'algorithme sera capable de l'affecter à sa catégorie correspondante. La classification de textes peut se faire de deux manières différentes :

1. **Classification manuelle** : dans la première, un annotateur humain interprète le contenu du texte et le classe en conséquence. Cette méthode peut généralement fournir des résultats de qualité mais elle est longue et coûteuse.
2. **Classification automatique** : applique l'apprentissage automatique, le traitement du langage naturel et d'autres techniques pour classer automatiquement le texte de manière plus rapide et plus rentable.

Il existe de nombreuses approches de la classification automatique des textes, qui peuvent être regroupées en trois types de systèmes différents :

— Systèmes basés sur des règles
— Systèmes basés sur l'apprentissage automatique
— Systèmes hybrides

Le système doit être en mesure de classifier tout type de document en fonction de sa structure. Nous prévoyons atteindre un pourcentage de 95%. Nous utiliserons le NLP et le machine learning pour accomplir notre tâche.

3 État de l'art

3.1 Catégorisation des documents

Le problème de la catégorisation peut se résumer en une formalisation de la notion de similarité textuelle, soit en d'autres termes : trouver un modèle mathématique capable de représenter, pour ensuite comparer, la sémantique des textes. Si cette notion de similarité sémantique est un processus souvent intuitif pour l'homme, elle résulte d'un processus complexe et encore mal compris du cerveau. Le but de la recherche sur la classification automatique de textes est donc de trouver un algorithme permettant d'assigner un texte à une classe avec le plus grand taux de réussite possible, sans toutefois assigner un texte à trop de classes. Dans un tel contexte, une mesure de similarité textuelle permet d'identifier le ou les documents les plus proches de la requête formulée dans le cadre de la recherche d'informations ou bien la ou les catégories les plus proches du document à classer. Si la plupart des approches existantes utilisent un même modèle pour la représentation des documents, il n'en est pas de même pour les formules adoptées comme élément de mesure.

3.2 Définition du modèle de catégorisation textuel de référence

L'objectif d'un sytème de catégorisation est d'approximer la fonction de catégorisation exacte qui associe à chaque couple de (document / classe) une valeur (vraie / fausse), en fonction de

l'appartenance ou non du document à la classe :

$$\breve{\Phi} : D \times C \rightarrow \{T, F\}$$

On nommera ϕ , l'approximation d'une telle fonction dont l'objectif repose essentiellement sur une notion de similarité entre documents et classes. Si nous nous intéressons seulement aux systèmes dont la représentation des documents (resp.classes) appartient à un même espace1, il est possible de représenter cette notion de similarité au sein d'un modèle de catégorisation textuel (MCT) définissant : – une formalisation de la "sémantique" d'un texte. – Une formalisation de la notion de classe. – Une relation entre textes et classes. – Une politique de catégorisation. Dans notre problématique de catégorisation, il n'est pas nécessaire de définir d'autres relations. En effet, comparer les textes entres eux et les classes entre elles n'est utile que dans le cas d'approches non supervisées. Le modèle que nous définissons s'inspire du modèle de similarité textuel et en utilisant le model Support Vector Machine (SVM) et l'algorithme de Gradient Boosted Decision Trees (GBDT).

4 Solutions

En guise de solution, nous avons opté pour le CANAVAS et l'ordre de :

4.1 CANAVAS

Pour la réalisation de ce travail nous présentons le CANAVAS de notre projet comme suit.

Titre : Classification automatique des documents à l'aide de l'apprentissage et du traitement de langage naturelle		
1. Déclaration du problème:	**2. Résultat/classification**	**3. La provenance de nos données :**
Classification Automatisée des Documents à l'aide de l'apprentissage automatique et du traitement automatique du langage naturel. Ce projet viendra en aide aux organisations publiques, privées et non gouvernementales afin de faciliter la classification des documents en fonction de leur structure.	Le système doit être en mesure de classifier tout type de document en fonction de sa structure. Nous prévoyons atteindre un pourcentage de 95%. Nous utiliserons le NLP et le machine learning pour accomplir notre tâche.	Les données textuelles sur lesquelles nous allons travailler proviennent de ce site http://qwone.com/~jason/20Newsgroups/
4. Modèles	**5. Evaluation de la performance des modèles**	**6. Préparation de données**
Notre jeu de données est constituée de 18.828 documents réparties en 20 catégories. nous allons utiliser l'algorithme TF-IDF pour extraire les caractéristiques des fichiers; Pour la classification nous utiliserons quelques algorithmes de machine learning pour créer notre premier modèle et nous utiliserons quelques algorithmes de NLP pour créer un second modèle	Pour évaluer la performance de nos modèles, nous allons utiliser la matrice de confusion. De même nous allons utiliser la courbe ROC (received operating characteristic) pour visualiser les performances du classificateur.	Pour atteindre ces objectifs, les étapes suivantes seront nécessaires: Etape 1 : Acquisition d'un dataset approprié pour la classification des documents Etape 2 : Subdiviser en deux parties Test et entraînement Etape 3 : Extraction des caractéristiques des documents. Etape 4 : Classification des documents. Etape 5 : Evaluer la performance de l'algorithme Etape 6 : Utilisation du modèle avec une interface Web.

FIGURE *2 – Canavas*

4.2 Ordre de réalisation du projet

Pour mener à bien ce projet nous suivons la procédure suivante

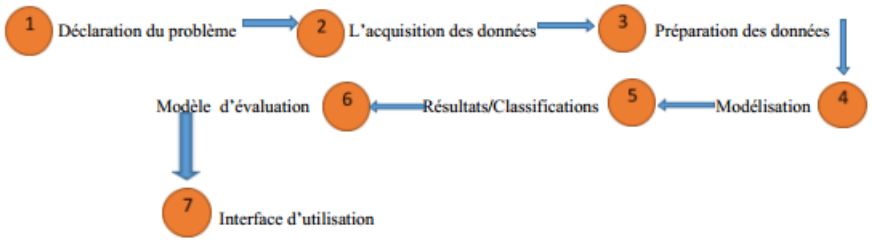

FIGURE *3 – phases de réalisation du projet*

4.3 Données

Notre dataset est de type texte, c'est un jeux de données du 20 Newsgroups. Cet jeux de données est une collection de documents de groupe de discussion. La collection de 20 Newsgroups de discussion est devenue un ensemble de données populaire pour les expériences dans les applications de texte des techniques d'apprentissage automatique, telles que la classification de texte et le regroupement de texte. Nous avons associé à ce dataset 20 Newsgroups le sac de mots anglais FastText crawl 300d 2M constitué de 2 million de vecteurs de mots pré-entraîné de l'ensemble de Crawl (600B tokens). Nous l'avons divisé en 2 parties dont 80% pour le training et 20% pour le test.

4.4 Environnement de développement, Librairies et langage

ENVIRONNEMENT	BIBLIOTHÈQUES	LANGUAGE	Algorithmes
Ubuntu 16.04	NLTK	python 3.6	GBDT
Editeur Spyder 3.3.3	Scikit-learn		SVM
	Numpy		
	Pickle		
	Pandas		

4.5 Modèles conceptuels & Algorithmes & Pipeline de traitement

4.6 Modèles conceptuels

Notre but est d'automatiser la tâche de classification des textes, nous avons une aperçu dans la figure ci-dessous du résultat que nous devront obtenir

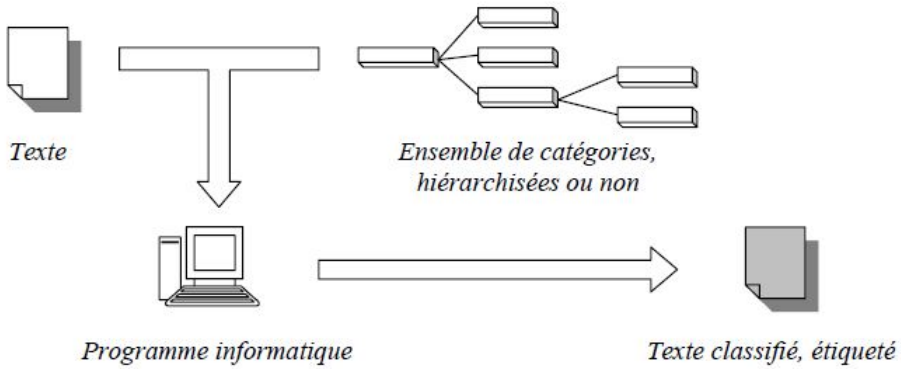

FIGURE *4 – La tâche de classification*

Le but de la catégorisation automatique de textes est d'apprendre à une machine à classer un texte dans la bonne catégorie en se basant sur son contenu. Habituellement, les catégories font référence aux sujets des textes, mais pour des applications particulières, elles peuvent prendre d'autres formes.En effet, on peut résoudre, par des techniques de catégorisation, des problèmes tels que l'identification de la langue d'un document, le filtrage de courrier électronique pertinent

ou indésirable, ou encore la désambiguïsation de termes. Un autre aspect du problème qui varie selon les applications est la présence ou non d'une contrainte concernant le nombre de catégories assignables à un document donné. Il se peut qu'on désire qu'un même texte ne soit associé qu'à une seule catégorie ou bien on peut permettre que plusieurs catégories accueillent un même document. Aussi, une précision supplémentaire est à faire : dans le cadre de la catégorisation de textes, l'ensemble de catégories possibles est déterminé à l'avance. Il est à noter que le problème consistant à regrouper des documents selon leur similarité, mais lorsque les groupes à former sont a priori inconnus, en est un à part entière connu sous le nom de regroupement («clustering») de textes.

Dans l'esprit de l'apprentissage automatique, on vise à construire des systèmes qui vont apprendre par eux-mêmes à classer les documents. On met l'accent sur l'automatisation, par apprentissage, de la création du classificateur. À l'aide d'un ensemble de textes déjà associés à des catégories, on entraîne le système. Cette banque de textes libellés doit être préalablement construite par un humain.

Ensuite, la machine tente d'apprendre la tâche de classification en observant le travail fait par l'humain. Elle essaie de généraliser les liens entre les textes et les catégories en analysant des exemples. Après la phase d'entraînement, le classificateur peut procéder lui-même au classement de nouveaux textes. Souvent , on s'inspire de la stratégie bien connue «diviser-pour-régner» et on découpe le problème en plusieurs sous-problèmes plus faciles à résoudre. Plus précisément, on construit un classificateur binaire pour chaque catégorie, qui va déterminer si, oui ou non, un document en fait partie. En fait, l'apprentissage de la distinction de deux catégories s'avère plus facile que l'apprentissage sur l'ensemble des catégories. De plus, comme les catégories peuvent éventuellement se chevaucher, le document soumis à plusieurs classificateurs sera jugé par chacun d'eux et pourra être associé à plus d'une catégorie.

En résumé, l'objectif global devient donc de créer un constructeur automatique de classificateurs. À prime abord, la tâche est plus complexe qu'avec la première approche, où elle relève d'un expert chargé de produire les règles de classification. Cependant, en bout de ligne, lorsqu'un tel constructeur de classificateurs est mis au point, il peut être transposé d'un corpus de textes à un autre, d'un domaine à un autre, sans nécessiter beaucoup de travail supplémentaire, mis à part peut-être l'ajustement de quelques paramètres. L'effort en recherche des dernières années semble avoir été mis plutôt de ce côté. La figure 5 présente le processus général d'entraînement d'un tel système, tandis que la figure 6 illustre le processus de classification d'un nouveau document.

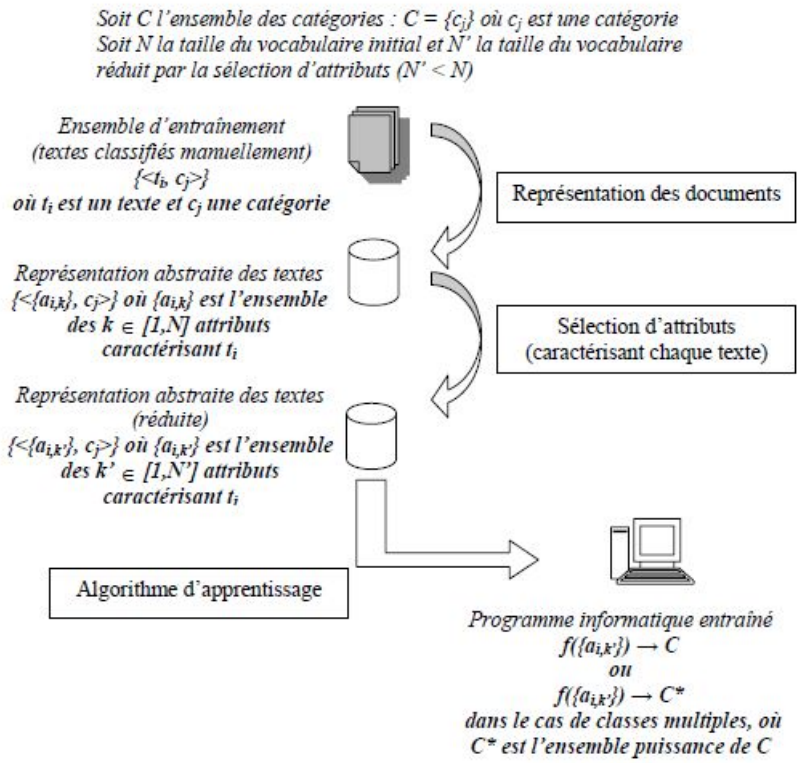

FIGURE 5 – *Entraînement d'un système de classification automatique de textes*

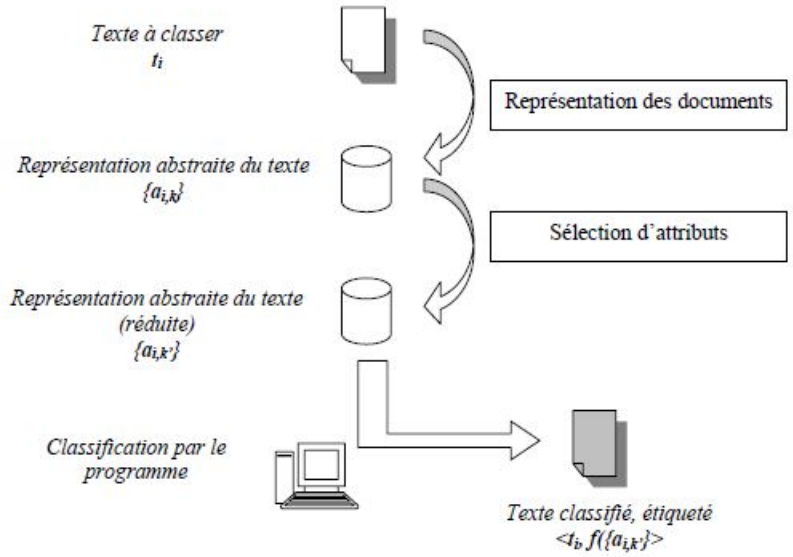

FIGURE 6 – *Classification d'un nouveau document*

4.7 Algorithmes

Nous allons appliquer principalement deux algorithmes d'apprentissage supervisé à savoir le **Le Support Vector Machine (SVM) et le Gradient Boost Decision Tress (GBDT)**, si nous avons fait le choix de ces deux algorithmes c'est du aux résultats de notre état de

l'art. Mais avant cela un pré-traitement est nécessaire. Le pré-traitement va nous permettre de représenter le texte d'une manière vectorielle afin que nous puissions appliquer nos algorithmes de classification. Une représentation vectorielle dans laquelle chaque texte est représenté par un vecteur de n termes pondérés. À la base, les n termes sont tout simplement les n différents mots apparaissant dans les textes de l'ensemble d'entraînement. Cette approche est aussi appelée **bag-of-words (BOW)**. Ensuite nous allons calculer le poids d'un terme par la fonction **TF-IDF (acronyme pour «term frequency inverse document frequency»)**. **Le TF-IDF** donne plus d'importance aux mots qui apparaissent souvent à l'intérieur d'un même texte, ce qui correspond bien à l'idée intuitive que ces mots sont plus représentatifs du document. Mais sa particularité est qu'elle donne également moins de poids aux mots qui appartiennent à plusieurs documents, pour refléter le fait que ces mots ont un faible pouvoir de discrimination entre les classes. Le poids d'un terme tk dans un document dj est calculé ainsi :

$$\text{tfidf}(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|Tr|}{\#(t_k)}$$

où

- (tk, dj) est le nombre d'occurrences de tk dans dj
- |Tr| est le nombre de documents d'entraînements
- (tk) est le nombre de documents d'entraînements dans lesquels tk apparaît au moins une fois

La taille impressionnante du vocabulaire peut s'avérer un obstacle à l'utilisation d'algorithmes plus complexes. Pour cela, appliquons une technique de réduction de dimension du vocabulaire qu'est la sélection d'attributs («feature selection»). **La feacture selection** prend les attributs (ou mots) d'origine et conserve seulement ceux jugés utiles à la classification, selon une certaine fonction d'évaluation, les autres sont rejetés.

Une fois ces étapes réalisé, nous pourrons passer à l'étape de la classification. Commençons par le **SVM**.

4.7.1 Le Support Vector Machine (SVM)

Les Machines à Support Vectoriel («Support Vector Machines» ou (SVM)) forment une classe d'algorithmes d'apprentissage qui peuvent s'appliquer à tout problème qui implique un phénomène fonction(f) et qui, à partir d'un jeu d'entrées x, produit une sortie y = f(x), et où le but est de retrouver f à partir de l'observation d'un certain nombre de couples entrée/sortie. Le problème revient à trouver une frontière de décision qui sépare l'espace en deux régions, à trouver l'hyperplan qui classifie correctement les données et qui se trouve le plus loin possible de tous les exemples. On dit qu'on veut maximiser la marge, la marge étant la distance du point le plus proche de l'hyperplan.

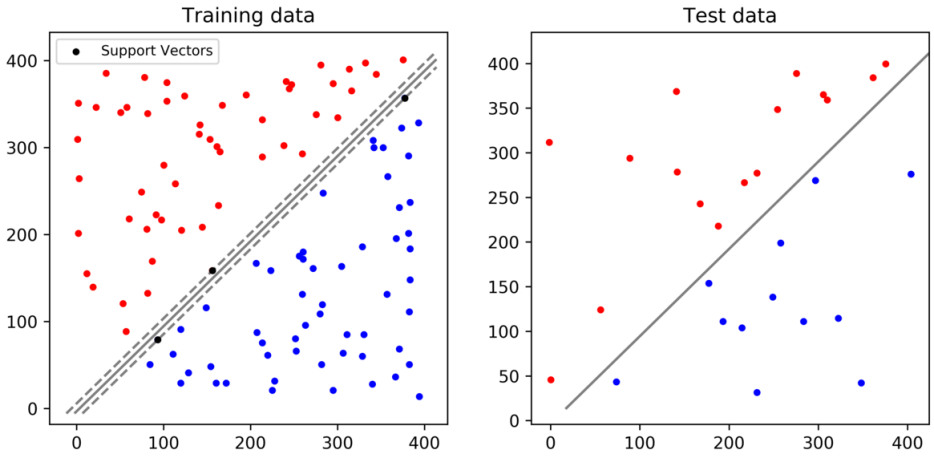

FIGURE 7 – *La tâche de classification avec le SVM*

Nous implémentons cette fonction à partir du **Package Scikit-learn pour Python - SVC** en utilisant le noyau polynomial qui est recommandé pour le traitement de texte

4.7.2 Le Gradient Boost Decision Tree (GBDT)

Le GBDT est une technique d' apprentissage automatique pour les problèmes de régression et de classification , qui produit un modèle de prédiction sous la forme d'un ensemble de modèles de prédiction faibles, généralement des arbres de décision . Il construit le modèle par étapes comme le font les autres méthodes de boosting , et il les généralise en permettant l'optimisation d'une fonction de perte arbitrairement différenciable .

Il y a deux arbres dans le graphique. x est un échantillon d'entrée. Après avoir traversé les deux arbres, l'échantillon X tombe sur les nœuds foliaires de deux arbres. Chaque nœud feuille correspond à la caractéristique unidimensionnelle LR(Linear Regression). Ensuite, toutes les caractéristiques LR correspondant à l'échantillon sont obtenues en parcourant l'arbre. Le nouveau vecteur propre est 0/1. Par exemple, le graphique ci-dessus a deux arbres, l'arbre de gauche a trois nœuds de feuille, l'arbre de droite a deux nœuds de feuille et la dernière caractéristique est un vecteur à cinq dimensions. Pour l'entrée x, supposons qu'il tombe sur le premier nœud de l'arbre de gauche, encodant [1,0,0], et sur le deuxième nœud de l'arbre de droite encodant [0,1], donc l'encodage global est [1,0 , 0,0,1], qui est tapé dans le LR pour la classification.

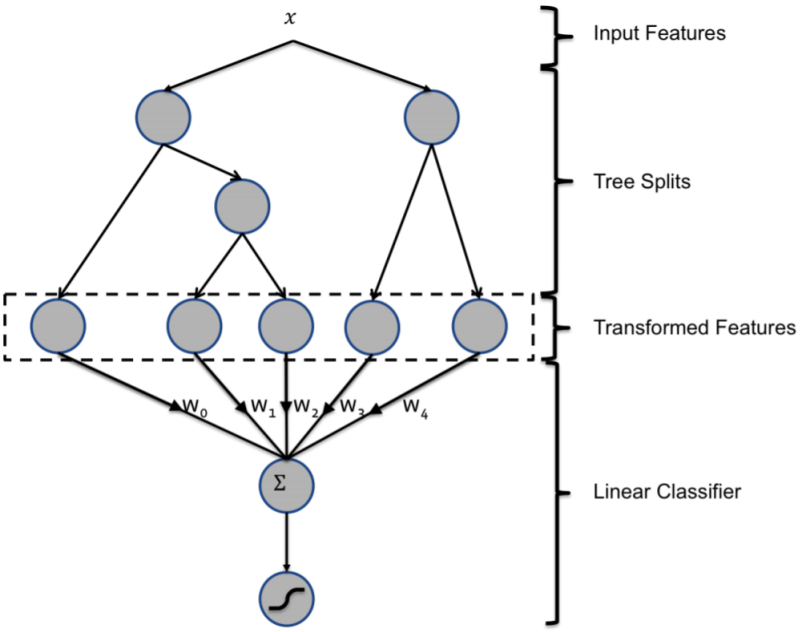

FIGURE *8 – Le classificateur GBDT*

Pour l'implementation, nous allons utiliser **le package sklearn.ensemble import GradientBoostingClassifier**

4.8 Pipeline de Traitement

Se référer au point 4.2 Ordre de réalisation du projet

5 Implémentation et Expérimentation

5.1 Implémentation

Nous avons définis trois classes réparties comme suit :

1. **data_process.py** , dans cette classe nous avons éffectué la division du jeu de données ,le prétraitement en définissant les fonctions de **stopword, le bow, embedding, le tf-idf**
2. **gbdt.py** , cette classe nous permettra d'utiliser nos classificateurs **SVM et le GBDT** ainsi que la fonction d'evaluation (**Matrice de confusion**) de nos classificateurs
3. **main.py** dans cette classe principale , nous avons importer nos deux classes (le gbdt.py et le data-process.py) en appliquant le picke ainsi que les meilleurs paramétres de tf-idf

5.2 Expérimentation & Analyse de résultats

A la fin de l'entraînement de notre modèle, ci-dessous les différents résultats qui quantifient la perte dans la phase de l'entraînement de notre jeux de données. Ces resultats sont capturés selon l'ordre d'entraînement des différents modèles et le formats pré-traitement du jeux de données et sa sauvegarde dont nous avons les fichiers normale (Fastext) et les fichiers inversés TF-IDF.

5.2.1 GBDT + Fastext

Ci-dessous les résultats du modèle GBDT. Elle est faite en 20 itérations avec une précision d'environs 0.7445.

```
----------------start train----------------
      Iter       Train Loss   Remaining Time
         1       41327.7462           13.08m
         2       38194.2898           13.00m
         3       35836.3390           12.85m
         4       33874.6764           12.72m
         5       32136.5894           12.59m
         6       30599.7151           12.49m
         7       29245.7928           12.36m
         8       28021.8375           12.23m
         9       26909.7797           12.10m
        10       25888.2870           11.98m
        20       18754.8878           10.71m
        30       14779.3147            9.42m
        40       12161.6465            8.09m
        50       10290.3550            6.74m
        60        8875.1812            5.40m
        70        7777.1916            4.05m
        80        6865.8081            2.70m
        90        6133.6982            1.35m
       100        5525.2708            0.00s
acc 0.7445
```

FIGURE *9 – Résultat de la précision du GBDT*

5.2.2 Précision par type de classe, cas de GBDT + Fastext

La capture ci-dessous, nous montre les précisions des différents type de classe (catégories) de dossiers. Elles sont plus ou moins variables du à la répartition des différents mots des différents classes de dossiers en fichiers normale (Fastext) en nombre inégal lors de la phase de pré-traitement du jeux de données.

	precision	recall	f1-score	support
0	0.75	0.72	0.73	200
1	0.83	0.83	0.83	200
2	0.93	0.86	0.90	200
3	0.63	0.76	0.69	200
4	0.63	0.56	0.59	200
5	0.72	0.64	0.68	200
6	0.68	0.72	0.70	200
7	0.86	0.85	0.85	200
8	0.48	0.44	0.46	200
9	0.90	0.86	0.88	200
10	0.86	0.82	0.84	200
11	0.73	0.73	0.73	200
12	0.88	0.82	0.85	200
13	0.61	0.64	0.62	200
14	0.86	0.92	0.89	200
15	0.78	0.83	0.81	200
16	0.69	0.69	0.69	200
17	0.95	0.92	0.93	200
18	0.58	0.64	0.61	200
19	0.60	0.62	0.61	200
avg / total	0.75	0.74	0.74	4000

FIGURE *10 – Résultat de la précision du GBDT*

5.2.3 GBDT + TF-IDF

Le modèle GBDT appliqué aux fichiers de type TF-IDF, donne les résultats ci-dessous. le modèle est entrainé avec une précession total de 0.9485.

----------start train----------

Iter	Train Loss	Remaining Time
1	23424.1179	43.58m
2	18383.0523	43.41m
3	14932.0975	42.96m
4	12337.5801	42.53m
5	10317.4388	42.15m
6	8712.3711	41.75m
7	7425.0040	41.32m
8	6372.3906	40.89m
9	5515.3502	40.47m
10	4808.7258	40.04m
20	1810.9575	35.73m
30	1236.4123	31.22m
40	1083.0770	26.68m
50	1003.2810	22.11m
60	943.3836	17.55m
70	901.3651	13.09m
80	860.1399	8.67m
90	829.2766	4.32m
100	802.1868	0.00s

acc 0.9485

FIGURE *11 – Résultat de la précision GBDT + TF-IDF*

5.2.4 Précision par type de classe, cas de GBDT + TF-IDF

Ci-dessous les précisions par type de catégories de dossiers qui sont variables du à la répartition des différents mots des différents classes de dossiers en fichiers inversés TF-IDF dans le sac des mots en nombre inégal . Cas de l'algorithme GBDT sur les fichiers inversés TF-IDF.

	precision	recall	f1-score	support
0	0.98	0.97	0.98	200
1	1.00	1.00	1.00	200
2	0.99	1.00	1.00	200
3	0.99	0.96	0.98	200
4	0.98	0.98	0.98	200
5	0.97	0.95	0.96	200
6	0.96	0.97	0.97	200
7	1.00	1.00	1.00	200
8	0.67	0.71	0.69	200
9	0.95	0.95	0.95	200
10	1.00	0.98	0.99	200
11	0.94	0.91	0.92	200
12	0.99	0.99	0.99	200
13	0.98	0.99	0.99	200
14	1.00	0.98	0.99	200
15	0.98	1.00	0.99	200
16	0.98	0.98	0.98	200
17	1.00	0.99	0.99	200
18	0.78	0.83	0.81	200
19	0.85	0.78	0.81	200
avg / total	0.95	0.95	0.95	4000

FIGURE *12 – Résultat de la précision GBDT + TF-IDF*

5.2.5 Support Vector Machine (SVM)

Maintenant, pour essayer un algorithme de Support Vector Machine (SVM), sur l'ensemble de données pour voir qu'il a de meilleures performances. Comme précédemment, ajustez d'abord un SVM simple sans régler aucun paramètre. nous avons une précision de test de 82,24%. Voici l'analyse des performances de différents groupes de discussion :

	precision	recall	f1-score	support
alt.atheism	0.72	0.71	0.71	319
comp.graphics	0.79	0.70	0.74	389
comp.os.ms-windows.misc	0.73	0.77	0.75	394
comp.sys.ibm.pc.hardware	0.71	0.68	0.69	392
comp.sys.mac.hardware	0.82	0.82	0.82	385
comp.windows.x	0.84	0.77	0.80	395
misc.forsale	0.82	0.87	0.85	390
rec.autos	0.91	0.89	0.90	396
rec.motorcycles	0.92	0.97	0.94	398
rec.sport.baseball	0.90	0.91	0.90	397
rec.sport.hockey	0.86	0.98	0.92	399
sci.crypt	0.85	0.96	0.90	396
sci.electronics	0.81	0.62	0.70	393
sci.med	0.90	0.87	0.88	396
sci.space	0.83	0.96	0.89	394
soc.religion.christian	0.74	0.93	0.82	398
talk.politics.guns	0.70	0.93	0.80	364
talk.politics.mideast	0.92	0.93	0.92	376
talk.politics.misc	0.89	0.56	0.69	310
talk.religion.misc	0.82	0.39	0.53	251
avg / total	0.83	0.82	0.82	7532

FIGURE *13 – Résultat de la précision SVM*

Par rapport à GBDT + TF-IDF, les résultats de l'utilisation de SVM ont des scores plus uniformément répartis, il n'y a pas de 85%, ni de score élevé de 98%, 99%, 100% et ou celuis de GBDT + Fastext qui a un score de base de 58% et le score le plus élevé de 95%. Mais le score global pour SVM est moyennement bon que celui de GBDT.

5.2.6 Matrice de confusion

Nous avons constaté que lors de la génération de matrice de confusion deux colonnes sont décales du faite que nous voulons le rendre lisible. Mais cella ne nous empêche constater que les confusions sont rares et nous avons remarqué aussi que la diagonale se détache nettement.

```
Confusion matrix:
[[226   0   0   0   0   0   0   0   0   0   0   1   3   1   0   2   0   1
    0  11]
 [  0 271   3   9   2   5   2   0   0   1   0   0   3   1   0   0   0   1
    0   0]
 [  0   6 257  14   5   6   1   0   0   0   0   0   2   0   1   0   0   0
    0   0]
 [  1   7  12 243   7   3  12   2   0   0   0   1  12   0   0   1   0   0
    0   0]
 [  0   2   3   8 233   1   4   0   1   0   0   0   3   0   1   0   0   0
    0   0]
 [  0  19  10   4   0 258   0   0   0   1   0   0   2   0   2   0   1   0
    0   0]
 [  0   0   3   7   5   0 257   7   2   0   0   1   7   0   1   0   0   0
    0   0]
 [  0   0   0   2   1   0   5 306   2   2   0   0   4   1   0   0   1   0
    0   0]
 [  0   0   0   0   1   0   3   3 285   0   0   0   1   0   0   1   0   0
    0   0]
 [  0   0   0   0   0   0   3   2   0 305   2   1   1   0   0   0   0   0
    1   0]
 [  0   0   0   0   0   0   1   0   1   0 300   0   0   0   0   0   0   0
    0   0]
 [  0   0   1   1   0   2   2   1   0   0   0 282   0   1   1   0   2   2
    1   1]
 [  0   2   2  11   3   1   5   5   1   0   1   1 278   2   1   0   0   0
    0   0]
 [  0   3   0   0   1   1   1   0   0   0   0   0   0 269   0   1   1   0
    0   0]
 [  0   5   0   0   1   0   0   0   0   0   2   0   2   0 294   0   0   0
    1   0]
 [  1   1   1   0   0   1   0   1   0   0   0   0   0   1   1 284   1   0
    0   1]
 [  0   0   1   0   0   0   0   0   1   3   0   0   0   0   0   1 233   1
    5   1]
 [  0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   2   0 293
    0   0]
 [  0   2   0   0   0   0   2   0   0   1   0   1   0   1   0   0   9   5
  213   2]
 [ 13   0   0   0   0   0   0   0   0   0   0   1   0   2   0   9   2   1
    2 141]]
```

FIGURE *14 – Résultat de la matrice de confusion*

6 Conclusion et Perspectives

Ce projet nous a permis de travailler sur les méthodes de classification de documents fondés sur l'analyse syntaxique et le calcul sémantique à base de vecteurs d'indexation. Il a été réalisé en plusieurs phase à savoir le pré-traitement du jeux de données en utilisant le sac de mots anglais FastText crawl 300d 2M constitué de 2 million de vecteurs de mots pré-entraîné de l'ensemble de Crawl (600B tokens) pour generer les fichiers dite normales (Faste) et les fichiers inversés (TF-IDF). Nous avons pu developper notre connaissance en gestion des projets machine learning en utilisant le CANAVAS comme support de travail. Nous avons explorer aussi les techniques de classification des documments textuaires dans son ensemble. A cela s'ajoute, grâce à ce projet, notre vision du naturel langage process combinés aux algorithmes tel que (le SVM et le GBDT), les résultats du GBDT + TF-IDF nous donne un pourcentage de 0.95% qui est meilleur que celui de GBDT + FastText qui est en moyenne égale à 0.75% . Néanmoins, il est a noté que nous avons testé ces algorithmes sur un seul jeux de données du au faite que plusieurs recherches scientifiques lié au domaine de traitement automatique de langage naturel ont été mené. Cela nous a permis de comparer nos résultats aux résultats des travaux précédents,d'où le choix de ce dataset. Nos travaux restent ouvert au deep learning qui est une piste à découvrir. Code source

Références

[1]https ://medium.com/@eaifundoffical/fusion-scheme-of-gbdt-and-lr-1d3b330ec46f, consulté le 15/11/2019

[2]https ://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html, consulté le 18/11/2019

[3]https ://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html, consulté le 18/11/2019

[4]https ://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html, consulté le 19/11/2019

[5]https ://docs.python.org/3/library/pickle.html, consulté le 19/11/2019

[6]SIMON RÉHEL, «CATÉGORISATION AUTOMATIQUE DE TEXTES ET COOCCURRENCE DE MOTS PROVENANT DE DOCUMENTS NON ÉTIQUETÉS», FACULTÉ DES SCIENCES ET DE GÉNIE UNIVERSITÉ LAVAL QUÉBEC, Janvier 2005

[7]Simon Jaillet, Maguelonne Teisseire, Jacques Chauche, Violaine Prince. «Classification automatique de documents»,LIRMM-CNRS - Université Montpellier 2, 2016

[8]LABIADALI, «SÉLECTION DES MOTS CLÉS BASÉE SUR LA CLASSIFICATION ET L'EXTRACTION DES RÈGLES D'ASSOCIATION»,MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES , juin 2017

[9]https ://github.com, consulté le 23/11/2019