Université Nice Sophia Antipolis

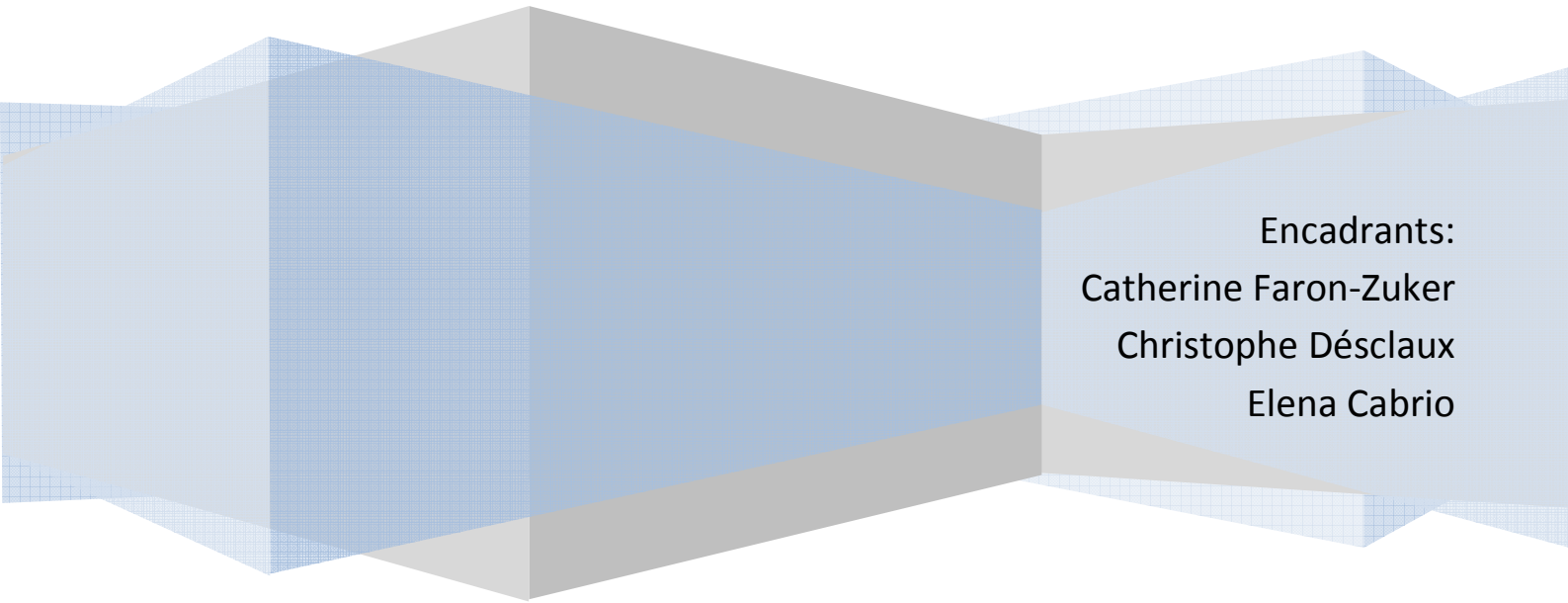
POLYTECH NICE-SOPHIA

Rapport de Projet de Fin d'Etudes

Catégorisation automatique de news à l'aide de techniques d'apprentissage supervisé

Ameni Bouaziz

M2 IFI/KIS



Encadrants:
Catherine Faron-Zuker
Christophe Désclaux
Elena Cabrio

Table de Matières

1. Introduction

Aujourd'hui avec le développement d'internet nous sommes en présence d'une quantité énorme d'information électronique. Le coût de plus en plus faible du stockage fait en sorte que cette quantité augmente continuellement. Parallèlement à cela, les sources de données ne cessent de se diversifier, aujourd'hui pour accéder à une information on peut avoir recours à des journaux électroniques, à des sites spécialisés, à des réseaux sociaux… Pour pouvoir gérer cette quantité grandissante de données et en tirer le plus d'information possible il devient nécessaire de l'organiser et de la catégoriser. C'est dans ce cadre que s'inscrit le projet ZONE, c'est un projet qui a pour but de développer un serveur d'annotation sémantique de news afin de les classer en différentes catégories, Ainsi un client intéressé par un type particulier d'information pourra trouver facilement toutes les news qui le concernent quelles que soient leurs provenances. Le principe de l'application est le suivant : grâce à un serveur abonné à différents flux RSS, on extrait et annote les textes des news, chaque texte se voit donc attribuer une ou plusieurs catégories selon son contenu, les clients n'ont donc qu'à soumettre des requêtes SPARQL pour récupérer et filtrer les flux RSS de news.

Dans le but d'optimiser la catégorisation dans le serveur d'annotations de ZONE, les résultats de plusieurs méthodes sont combinés (Wiki Meta, Open Calais, …), notre projet consiste à implémenter une nouvelle méthode de classification basée sur la fouille de données et utilisant le principe d'apprentissage supervisé et particulièrement SVM.

Ce rapport décrit le travail réalisé au cours de ce projet, la première partie présente un état de l'art de la question de classification de documents, la partie suivante explique les différentes étapes de l'algorithme implémenté, et la dernière partie décrit les expériences réalisées ainsi que les résultats obtenus et leurs analyses.

2. Etat de l'art

2.1. Définition de la classification

La classification automatique de documents est un problème connu en informatique, il s'agit d'assigner un document a une plusieurs catégories ou classes. Le problème est différent selon la nature des documents en question, en effet la classification de textes diffère de la classification de documents images, vidéo ou encore son. On peut aussi imaginer des classifications selon des paramètres associés aux documents tels que par exemple l'auteur, la date de parution… Dans le cadre de ce projet et dans la suite de rapport nous nous baserons sur la classification de documents de type texte selon leur contenu. Toute référence ultérieure à la classification renvoie donc à cette notion.

2.2. Applications de la classification

La classification automatique est une technique utilisée dans plusieurs domaines. Sa capacité prédictive la rend rapide et efficace. Parmi les applications où la classification est utilisée, nous trouvons le filtrage de spam, en effet il s'agit de traiter les messages électroniques textuels, identifier leurs caractéristiques et les classer en deux groupes messages désirés ou non désirés. [1]
Une autre application est la détermination automatique du sujet d'un texte pour le classer automatiquement afin de notifier des personnes intéressées par ce sujet de la présence d'un nouveau texte…

2.3. Etapes de la classification

La classification automatique de documents doit être obligatoirement précédée par une phase de préparation de données appelée « préprocessing ».

2.3.1. Préprocessing

2.3.1.1. Extraction de données

L'extraction de données est une tâche qui s'est développée dans le domaine de Traitement Automatique des Langues (TAL). Elle consiste à identifier et extraire d'un texte les éléments pertinents contenant des informations dont la nature est spécifiée à l'avance. Elle vise donc à transformer un texte de son format initial (une suite de chaines de caractères) à une représentation structurée et donc un format qui soit compréhensible par l'ordinateur. Elle se fait en reconnaissant dans le texte des unités lexicales particulières.
Il existe plusieurs techniques d'extraction de données telles que :

- **Les outils terminologiques** : nous nous intéressons aux termes présents dans un texte, ces outils peuvent être linguistiques, statistiques ou mixtes, les méthodes linguistiques se basent sur des patrons lexicaux-syntaxiques et sur le découpage des textes en unités syntaxiques. Les méthodes statistiques font des calculs sur les fréquences et les distributions des mots dans les textes.
- **Les méthodes d'extraction de relation terminologique** : ici nous nous intéressons plutôt à la relation entre les différents termes et structures des textes. Là aussi plusieurs approches peuvent être utilisées, on distingue les méthodes structurelles, contextuelles et distributionnelles

  Les méthodes structurelles se basent sur la structure syntaxique interne des termes.

  Dans le cas de méthodes contextuelles c'est plutôt le contexte dans lequel apparait le terme qui est pris en compte.

  Pour les méthodes distributionnelles, il s'agit de trouver des relations sémantiques entre les termes d'un texte à partir des statistiques de leur apparition et leur distribution.
- **La reconnaissance des entités nommées** : le plus souvent il s'agit de noms de personnes, de lieux, de noms d'organisations, etc. Selon le domaine traité par le texte d'autres entités peuvent

être ajoutées à la liste précédente, par exemple dans un texte médical il est important de reconnaître les noms de maladies ou de symptômes.
Plusieurs études sont faites sur les entités nommées, sur la façon de les identifier et de les reconnaitre dans un texte. Pour identifier un terme comme étant une entité nommée, on peut soit s'intéresser au terme lui-même (par exemple s'il commence par une lettre majuscule alors il y a une forte chance qu'il représente une entité nommée) soit à son entourage (par exemple s'il est précédé ou suivi d'un mot qui lui donne une qualification). [2]

Les applications de cette tâche ne se limitent pas à la classification de documents, en effet nous la retrouvons dans la veille technologique, dans les interfaces Homme machines, etc…

2.3.1.2. Calcul de poids

Les données collectées du texte n'ont pas toutes la même valeur informative, Certaines données sont plus importantes que d'autres pour la classification car elles apparaissent de façon répétée dans le texte, d'autres le sont moins à cause de leur caractère commun dans la langue. Pour permettre une meilleure classification, il faut refléter ce degré d'importance dans l'algorithme. C'est ainsi que nous attribuons à chaque mot un poids.
IL existe plusieurs méthodes de calcul de poids, en voici deux des plus connues :

- TF (Term Frequency) : C'est la fréquence d'apparition d'un mot dans le texte, elle est égale aux nombre d'occurrences de ce mot divisé par le nombre total de mots du texte.
- TF – IDF (term frequency, inverse document frequency): C'est une mesure statistique qui permet d'évaluer l'importance d'un mot dans un texte relativement à tout un corpus. Le poids ne dépend pas seulement de la fréquence du mot dans le texte en question mais aussi de la fréquence du mot dans tout le corpus. Ainsi un mot qui se répète dans tous les documents devient inutile pour la classification.
  TF_IDF = TF × IDF
  Où IDF = log $\left(\frac{nombre\ total\ de\ documents}{nombre\ de\ documents\ contenant\ le\ mot}\right)$

2.3.2. Classification

C'est à cette étape que se fait l'assignation du document à la classe à laquelle il appartient. La détermination de la classe se fait grâce à des algorithmes de classification qui exploitent les données extraites dans l'étape précédente et qui donnent en sortie la classe correspondante. Ces algorithmes se basent sur leur expérience passée où ils ont appris comment classer les textes, c'est de là que vient leur nom « Algorithmes d'apprentissage ». Il existe deux types de ces algorithmes :

- Les algorithmes d'apprentissage supervisé
- Les algorithmes d'apprentissage non supervisé.

Ces deux types seront détaillés dans la section suivante.

2.4. Les Techniques de classification automatique

2.4.1. Apprentissage non supervisé

2.4.1.1. Principe

L'apprentissage non supervisé consiste à apprendre à classer sans supervision. Au début de processus nous ne disposons ni de la définition des classes, ni de leurs nombres. C'est l'algorithme de classification qui va déterminer ces informations. Nous ne disposons pas non plus de données en entrée qui sont déjà classées, c'est aussi à l'algorithme de découvrir par lui-même la structure plus ou moins cachée des données et de former des groupes d'individus dont les caractéristiques sont communes. [3]
L'apprentissage non supervisé est utilisé dans plusieurs domaines tels que :

- Médecine : Découverte de classes de patients présentant des caractéristiques physiologiques communes.
- Le traitement de la parole : construction de système de reconnaissance de la voie humaine.
- Archéologie : regroupement des objets selon leurs époques.
- Traitement d'images
- Classification de documents

2.4.1.2. Exemples

Dans la littérature il existe plusieurs types d'algorithmes d'apprentissage non supervisé tels que les algorithmes de partitionnements et les algorithmes de classification hiérarchique :

- **Le partitionnement**: consiste au regroupement des données suivant leur degré de similarité. L'algorithme le plus célèbre appartenant à cette classe est K-means : c'est un algorithme qui permet de partitionner un ensemble de données automatiquement en K clusters. Il consiste tout d'abord à choisir k points qui représentent les centres des groupes à créer, puis à affecter les autres points aux centres les plus proches. Cette affectation est faite par le calcul de distance entre les points. Plusieurs distances peuvent être définies telles que la distance euclidienne ou la distance de Manhattan. Par la suite nous procédons à une étape de raffinement des groupes de façon itérative, le raffinement se fait par le recalcul des centres des groupes après chaque itération et par une réaffectation des points aux groupes. L'algorithme s'arrête quand aucun point ne bouge. [4]
- **La classification hiérarchique** : il existe deux types de classification hiérarchique : Ascendante et descendent. La classification ascendante consiste à utiliser une matrice de similarité afin de partir d'une répartition fine vers un groupe unique. Donc, il s'agit de fusionner les groupes jusqu'à ce qu'on obtient un seul groupe englobant tous les autres. Cette classification peut être représentée par un arbre hiérarchique ou dendrogramme. La classification descendante se présente comme l'inverse de la classification ascendante. Donc il s'agit de décomposer un cluster unique en sous-groupes jusqu'à l'obtention des singletons. [5]

2.4.2. Apprentissage supervisé

2.4.2.1. Principe

Contrairement à l'apprentissage non supervisé, nous commençons ici par un ensemble de classes connues et définies à l'avance. Nous disposons aussi d'une sélection initiale de données dont la classification est connue. Ces données sont supposées indépendantes et identiquement distribuées. Elles nous servent pour l'apprentissage de l'algorithme. La classification se fait par l'algorithme selon le modèle qu'il a appris. [3][6]

2.4.2.2. Exemples

Il existe plusieurs algorithmes d'apprentissage supervisé, cette section présente quelques-uns des plus connus parmi eux, il s'agit de kNN, LLSF, les réseaux de neurones et Naive Bayes [7]:

- kNN : (k nearest neighbor), c'est une approche statistique de classification très connue. Il a été prouvé que c'est une des méthodes les plus performantes après des tests réalisés sur le corpus de données Reuters. Le principe de l'algorithme kNN est le suivant : étant donné un texte à classer, l'algorithme cherche les k voisins les plus proches parmi les documents utilisés au cours de la phase d'apprentissage, les catégories de ces k voisins les plus proches serviront à donner des poids aux catégories candidates de classification. C'est le degré de similarité entre le document test et le document voisin qui est utilisé comme poids de la catégorie de ce dernier, si plusieurs voisins partagent la même catégorie alors le poids attribué à cette catégorie est égal à la somme des degrés de similarité entre le document test et chacun des voisins appartenant à cette catégorie. Par cette méthode on peut obtenir une liste des poids attribués à chaque catégorie, le document test est classé dans une catégorie si le poids attribuée à celle-ci est supérieur à un seuil fixé à l'avance.

- LLSF (linear least square fit) : c'est une approche de mapping développée par Yang[], il s'agit d'écrire les données d'apprentissage sous la forme de paires de vecteurs entrée/sortie, le vecteur d'entrée est composé des mots du texte accompagnés de leurs poids respectifs, alors que le vecteur de sortie est composé des différentes catégories avec leur poids binaires (1 si le texte appartient à une catégorie et 0 sinon), la résolution de l'équation suivante permet d'obtenir la matrice des coefficients de régression mot-catégorie.

  $$\mathrm{F_{LS}} = \min_{F} ||FA - B||^2$$

  Où A et B sont deux matrices qui représentent les données d'apprentissage et dont les colonnes sont composées des paires des vecteurs entrée/sortie. La matrice solution F_{LS} permet de donner pour tout texte un vecteur de catégories/poids. Comme pour kNN, un seuil est fixé et le document à classer appartient à toute catégorie ayant un poids supérieur au seuil fixé.

- Les réseaux de neurones : c'est une structure constituée de suite successive de couches de nœuds et qui permet de définir une fonction de transformation non linéaire des vecteurs

d'entrées (composés dans le cas de classification des mots pondérés de leur poids) en vecteur de catégories. La disposition des neurones dans le réseau ainsi que le nombre de couches utilisées ont une influence sur le résultat de classification.
Comparés aux autres méthodes de classification par apprentissage supervisé, les réseaux de neurones ont l'inconvénient que le coût d'apprentissage est assez élevé.

- NB (Naive Bayes) : c'est une méthode de classification probabiliste. Elle consiste à utiliser les probabilités jointes des mots et des catégories pour estimer la probabilité d'une catégorie sachant un texte à classer. Le caractère « naïf » de cette approche est dû au fait que les mots sont considérés indépendants, c'est-à-dire que la probabilité conditionnelle d'un mot sachant une catégorie est supposée indépendante des probabilités conditionnelles des autres mots sachant la même catégorie, cette assomption rend NB très efficace par rapport aux autres approches bayésiennes. Plusieurs versions de NB sont proposées dans la littérature, le model mixte multinominal par exemple a permis d'avoir de bonnes performances [].

2.4.2.3. SVM

Les « supports vectors machines » appelés aussi « maximum margin classifier» sont des techniques d'apprentissage supervisé basées sur la théorie de l'apprentissage statistique ou automatique. Les SVM sont relativement nouveaux, ils sont apparus en 1995 suite aux travaux de Vapnik. SVM traite d'un problème de classification bi classes. [7][8]
Le principe général de la classification par SVM peut être expliqué par la figure suivante:

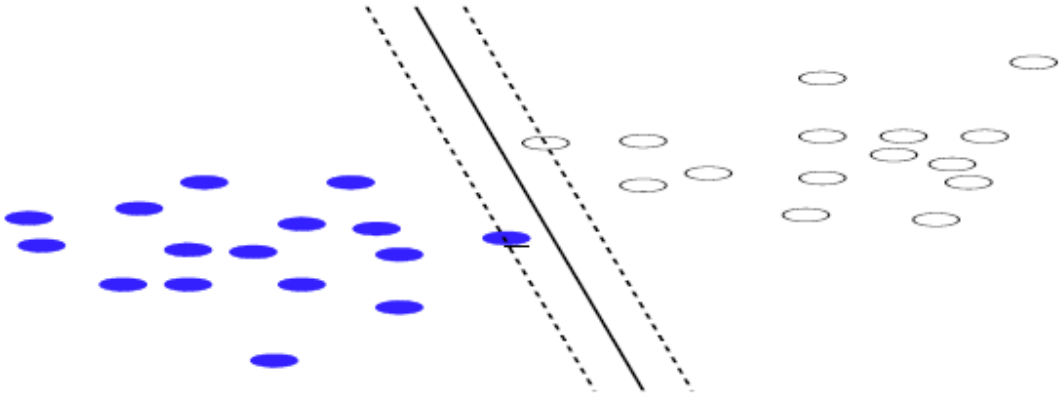


Figure 1: Principe de SVM

Le but de SVM est de determiner si un élément appartient à une classe ou pas. Nous disposons d'un ensemble de données et nous cherchons à séparer ces données en deux groupes. Le premier est l'ensemble de données appartenant à une classe, ces données sont étiquetées par (+) et un autre ensemble qui contient les éléments qui n'appartiennent pas à la classe donc étiquetées (-). L'algorithme SVM permet de trouver un hyperplan séparateur entre ces deux groupes. Pour optimiser la séparation SVM cherche l'hyperplan pour lequel la distance entre la frontière des deux groupes et les points les plus proches est maximale, c'est le principe de maximisation de la marge.

Dans certains cas les données sont non linéairement séparables. Il n'existe donc pas un hyperplan séparateur. Deux options sont possibles pour palier à ce problème :

- Définir une constante qui permet de tolérer une marge d'erreur (marge souple).Nous pouvons distinguer une constante pour les classes positive (Cp) et une constante pour les classes négatives (Cn). Les deux constantes sont en relation par un coefficient j. Soit **Cp = j × Cn**.
- Chercher une autre séparation non linéaire : Pour faciliter les calculs de cette frontière non linéaire nous utilisons une fonction noyau qui nous permet de passer dans un autre espace vectoriel de plus grande dimension où une frontière linéaire existe. Parmi les fonctions noyaux nous pouvons citer les noyaux polynomiaux, gaussiens… [6]

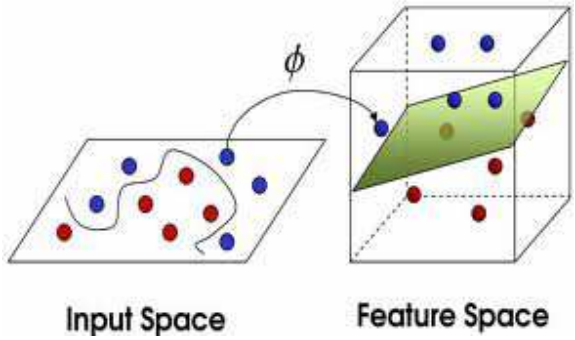


Figure 2: Exemple de fonction noyau SVM

SVM présente une solution au problème de classification binaire. Pour le problème à plusieurs classes nous faisons appel à SVM multi classes qui est une extension de SVM. Deux méthodes sont utilisées :

- One–versus-all : dans ce cas on construit autant de classifieurs SVM que de catégories, pour un texte donné, parmi tous les classifieurs qui ont donné un résultat positif, nous considérons celui qui a retourné la plus grande marge.
- One-Versus-one : si n est le nombre de catégories, alors on construit tous les classifieurs SVM possibles en prenant les catégories deux à deux, donc n (n – 1)/2, pour un texte donné, il appartiendra à la catégorie citée le plus souvent après application de tous ces algorithmes. [9]

3. Travail réalisé

Le but du projet est d'intégrer la classification par apprentissage supervisé dans le serveur d'annotation de ZONE. Pour cela nous nous sommes basés sur SVM et nous avons implémenté un algorithme en deux parties : apprentissage et classification. La figure suivante montre les principale étapes de cette implémentation :

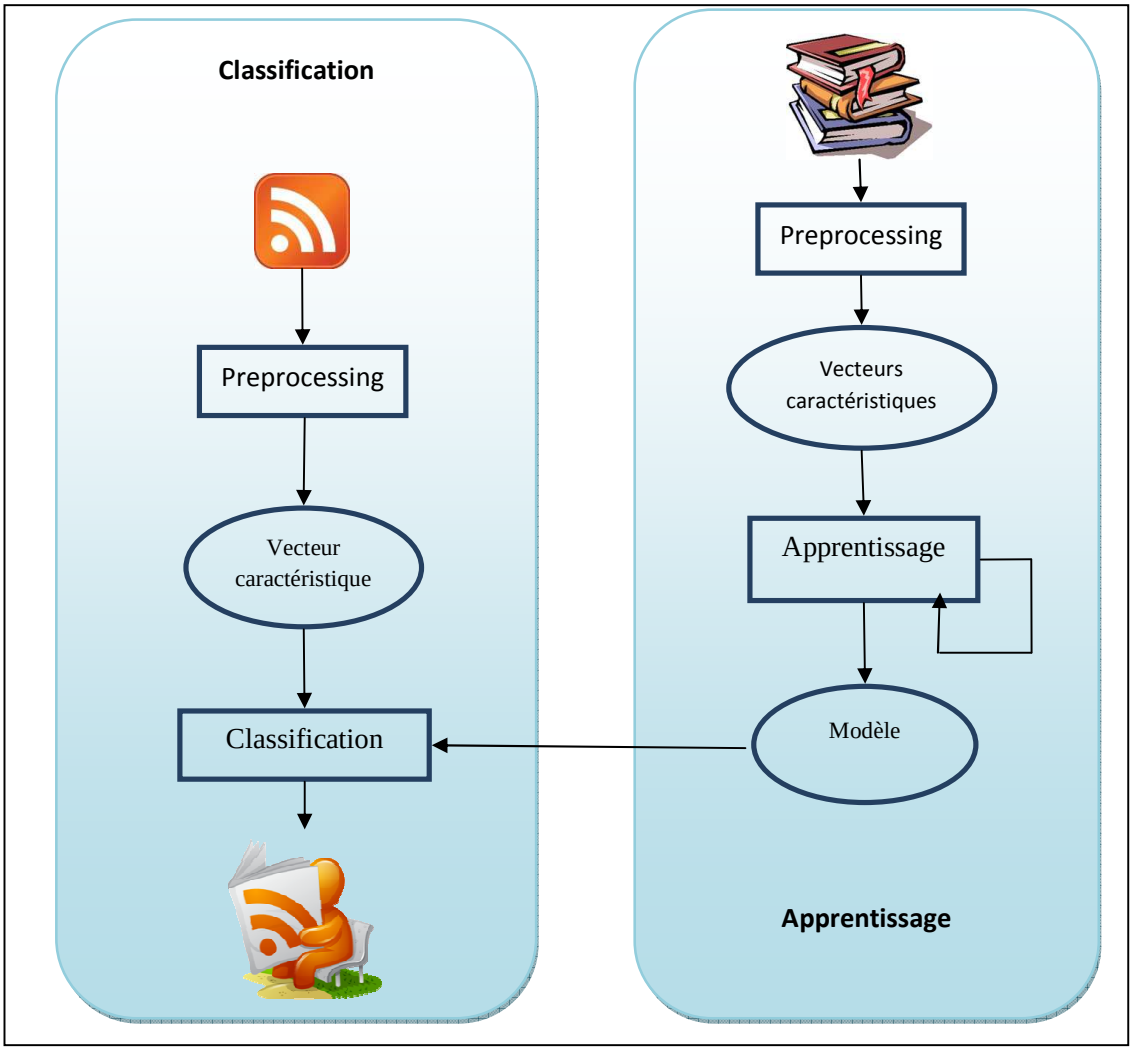


Figure 3: Processus de classification des flux RSS

- La phase de preprocessing consiste à utiliser les techniques de TAL pour transformer les textes en vecteurs de caractéristiques nécessaires pour SVM.
- La phase de l'apprentissage consiste à prendre en entrés les vecteurs et produire en sortie un modèle de classification.

- La phase classification consiste à prendre le vecteur caractéristique du texte et l'utiliser pour la classification selon le modèle appris.

3.1. Implémentation SVM choisie

L'algorithme réalisé dans ce projet a été développé en se basant sur SVM light. C'est une librairie de SVM écrite en langage C et conçue par **Thorsten Joachims**, chercheur à l'université de **Dortmund.** Elle est parmi les implémentations les plus connues de SVM. Elle présente plusieurs avantages :

- Extensible;
- Il supporte les ensembles d'apprentissage de grande taille ;
- Paramétrable et fournie les fonctions noyaux connues dans SVM.
- Il donne la main à l'utilisateur de définir son propre noyau suivant ces besoins ;
- Est un algorithme optimisé en temps d'exécution et en utilisation de mémoire.
- permet de résoudre trois types de problèmes qui sont : la reconnaissance des modèles, la régression et l'apprentissage par tri.

Le code source de SVM Light est disponible sur le site officiel de joachims [10]. Il peut être compilé pour toutes les plateformes.
Comme le code de ZONE est écris en JAVA nous avons choisi d'utiliser JNI SVM Light, une extension de SVM Light qui propose une interface JAVA pour accéder aux différente fonctionnalités de la librairie écrite en C. Cette interface a été développée par **Martin Theopard**, [11] un chercheur dans le groupe de ADReM de l'université d'Antwerp en Belgique.

3.2. Preprocessing

Cette phase est la première phase de l'algorithme implémenté. Elle consiste à préparer les données sous un format que les fonctions d'apprentissage et de classification de SVM Light comprennent.
Ce format se présente comme suit :
<target> <Dim> : <Val> <Dim> : <Val> <Dim> : <Val> <Dim> : <Val> …
Où
<target> correspond au label d'une catégorie (1 ou -1)
<Dim> un entier qui représente une dimension du problème étudié.
<Val> : le poids de la dimension correspondante.

Dans notre cas nous avons construit un dictionnaire de mots de tous les textes utilisés à la phase d'apprentissage. Pour un texte donné les valeurs de Dim correspondent aux rangs de ses mots dans le dictionnaire. Les Val sont les poids correspondants calculés par TF et TF_IDF.
Le préprocessing est très important dans le processus de classification car la construction du modèle est basée sur les données préparées. En effet des données mal préparées risquent de donner un modèle non performant. Ainsi nous avons procédé à un nettoyage des textes pour lemmatiser les mots et supprimer les stop words et les symboles de ponctuations.
Le graphe suivant donne une vue globale du préprocessing :

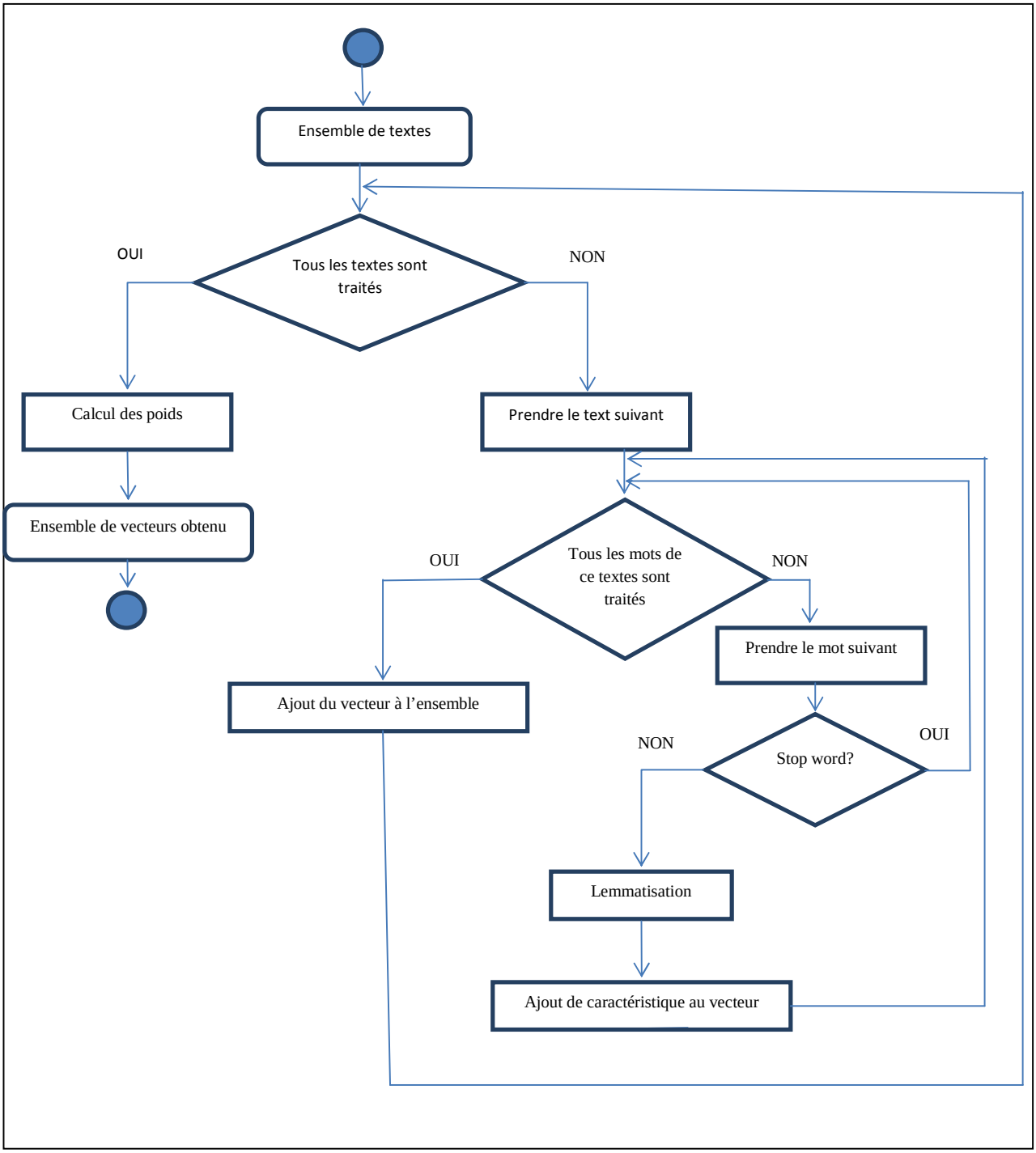


Figure 4: Algorithme de preprocessing

3.2.1. Elimination de stop words

Cette phase consiste à supprimer tous les mots standards dans un texte, ce sont des mots très communs et utilisés dans pratiquement tous les textes. Leur présence peut dégrader la performance de l'algorithme de classification en termes de coût et en termes de précision de la classification.
Nous pouvons prendre comme exemple pour l'anglais I, am, with, what, you …
Et pour le français il s'agit d'éliminer par exemple :

- Les pronoms pronominaux : il, elle, je, lui, ceux….
- Les auxiliaires : avoir, être et toutes leurs conjugaisons possibles
- Les opérateurs de conjonction : avec, et, ou, aux…
- L', c' …

Pour les deux versions française et anglaise nous avons écrit un petit programme qui permet de de déterminer si un mot est un stop word ou pas en référant à un fichier qui contient la liste de tous les stop words.

3.2.2. Tokonization et Lemmatisation

La tokonization est l'extraction des mots d'un texte. La lemmatisation est la réduction d'un mot à sa forme la plus simple, par exemple transformer tous les verbes conjugués à l'infinitif.
Ce traitement permet de normaliser les mots qui dérivent de la même racine et de les traiter comme s'ils étaient le même afin de leurs attribuer un poids unique.

3.2.2.1. Traitement des textes anglais

Pour les textes anglais, il existe plusieurs outils de TAL qui réalisent parfaitement ce traitement par exemple openNLP et CoreNLP. Dans notre programme nous avons utilisé CoreNLP. C'est un outil qui permet de parser, tokonizer, lemmatiser et d'extraire les entités nommés d'un texte. Il a été créé à l'université de Stanford [12]. Il permet de transformer un texte en un fichier XML de la forme suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="CoreNLP-to-HTML.xsl" type="text/xsl"?>
<root>
<document>
<sentences>
        <sentence id="1">
                ……..
        </sentences>
        <coreference>
<mention representative="true">
<sentence>2</sentence>
<start>3</start>
<end>6</end>
<head>5</head>
</mention>
```

```
<mention>
<sentence>2</sentence>
<start>1</start>
<end>2</end>
<head>1</head>
</mention>
</coreference>
</coreference>
</document>
</root>
```

La partie sentences contient toutes les phrases du texte en entrée.
Chaque phrase est représentée dans une balise sentence comme suit

```
<tokens>
        <token id="1">
        <word>Stanford</word>
        <lemma>Stanford</lemma>
        <CharacterOffsetBegin>0</CharacterOffsetBegin>
        <CharacterOffsetEnd>8</CharacterOffsetEnd>
        <POS>NNP</POS>
        <NER>ORGANIZATION</NER>
        </token>
 …
        <parse>(ROOT (S (NP (NNP Stanford) (NNP University)) (VP (VBZ is) (VP (VBN located) (PP (IN in)
(NP (NNP California))))) (. .)))
        </parse>

<basic-dependencies>
<dep type="nn">
<governor idx="2">University</governor>
<dependent idx="1">Stanford</dependent>
</dep>
                …

        </basic-dependencies>
</tokens>
```

Une phrase est un ensemble de mots (token). Un mot est caractérisé par :

- word : le mot tel qu'il apparaît dans le texte.
- Lemma : la racine du mot
- CharacterOffsetBegin : la position du premier caractère du mot dans la phrase.
- CharacterOffsetEnd : La position du dernier caractère du mot dans la phrase.
- Pos : Part of speech
- NER : le type de l'entité nommée représentée par le mot
- parse permet de construire un arbre syntaxique pour la phrase.
- basic-dependencies permet de construire les dépendances entre les phrases.

CoreNLP peut prendre plusieurs options pour générer une ou plusieurs parties du fichier XML. Ces options sont :

- ssplit : décomposer un texte en un ensemble de phrases.
- Tokenize : décomposer une phrase en mots
- Lemma : transformer le token en sa racine
- parse : construit la partie parse du fichier XML.
- Pos : construit la partie pos du fichier XML
- Ner : construit la partie ner du fichier XML
- etc

Dans notre programme nous avons utilisé l'interface java du coreNLP. Cette interface permet de définir les options de coreNLP et fournit des méthodes pour parser le XML généré. Ainsi nous avons pu récupérer toutes les lemmes.

3.2.2.2. Traitement des textes français

Cette tâche était plus difficile car CoreNlp ne permet supporte pas la langue française par défaut. Nous avons pu lui ajouter un fichier pour permettre la tokenization, mais pour la lemmatisation nous avons été obligés de chercher une solution en dehors de CoreNlp, cette solution consiste à parcourir un fichier contenant tous les mots français et les lemmes qui leur correspondent, ce fichier a été écrit par Anne Vilnat, professeure de l'université de Paris Sud, il est disponible dans sa page [].

3.2.3. Calcul des poids et préparation du vecteur caractéristique

C'est la dernière étape du préprocessing, nous avons implémentés les algorithmes TF et TF-IDF pour réaliser ce calcul, les résultats obtenus par TF-IDF étaient meilleurs donc nous l'avons considéré dans les tests finaux, nous avons choisi le design pattern « Factory » pour permettre de basculer facilement d'un algorithme à un autre et d'en rajouter d'autres. Pour TF-IDF le corpus de documents utilisé est celui des documents d'apprentissage, si dans le texte à classer un mot n'est jamais apparu dans le corpus d'apprentissage nous lui attribuons un poids négatif et nous l'ignorons dans le processus de classification. Comme indiqué dans la section 4.2, les vecteurs caractéristiques des textes sont composés des rangs de leurs lemmes dans le dictionnaire (construit également à partir de tous les lemmes du corpus d'apprentissage) accompagnés des poids correspondants. Pour les textes d'apprentissage, la valeur de target est égale à 1 quand le texte fait partie de la catégorie étudiée et -1 sinon, pour les autres textes cette notion n'est pas utilisée.

3.2.4. Exemple de preprocessing

Dans cette partie, nous donnons un exemple de trois documents qui illustrent le travail réalisé dans la partie préprocessing du programme, les trois textes sont en anglais, le premier et le deuxième appartiennent à la catégorie du Data Mining, alors que le troisième traite du domaine de la biologie.7
Initialement les trois textes sont :

Document1
Data mining is a powerful new technology with great potential to help companies focus on the most important information in the data they have collected about the behavior of their customers and potential customers.

Document2
It discovers information within the data that queries and reports can't effectively reveal.

Document 3
Biology is a natural science concerned with the study of life and living organisms

Première étape: extraction des mots à partir des textes:

Document1
Data/ mining/ is/ a/ powerful/ new/ technology/ with/ great/ potential/ to/ help/ companies/ focus/ on/ the/ most/ important/ information/ in/ the/ data/ they/ have/ collected/ about/ the/ behavior/ of/ their/ customers/ and/ potential/ customers/./

Document2
It/ discovers/ information/ within /the/ data/ that/ queries/ and/ reports/ can/'t/ effectively/ reveal/./

Document 3
Biology /is /a /natural /science /concerned /with /the /study /of /life /and/ living /organisms/. /

Deuxième étape : supprimer les stops words

La liste des stops words des 3 documents :

Document1
is/ a/with/to / on/ the/ most / in/ the/ they/ have / about/ the / of/ their/ and /./

Document2
It/ within /the/that/ and/'t/./

Document 3
is /a /with /the / of /and/. /

Troisième étape : Lemmatisation

Document1

Data/ mining/ power/ new/ technology/ great/ potential/ help/ company/ focus/ important/ information/ data/ collect/ behavior/ customer/ potential/ customer/

Document2
Discover/ information/ data/ query/ report/ can/ effective/ reveal/

Document 3
Biology /nature/science /concern /study /live/live/organism/

Quatrième étape : construction du dictionnaire

Rang	**Lemme**
1	Data
2	Mining
3	Power
4	new
5	Technology
6	Great
7	Potential
8	help
9	company
10	focus
11	important
12	information
13	collect
14	behavior
15	customer

Rang	**Lemme**
16	Discover
17	Query
18	Report
19	Can
20	Effective
21	Reveal
22	Biology
23	Nature
24	Science
25	concern
26	Study
27	live
28	organisme

Quatrième étape : calcul des poids

- Nombre de document = 3
- Nombre de mots dans document 1 = 18
- Nombre de mots dans document 2= 8
- Nombre de mots dans document 3= 8
- Nous calculons le poids de chaque mot dans ces textes.
  Exemple poids du mot « live » dans le document 3 :
  TF26 = 2/8= 0.25
  IDF26 = log (3/1) = 0.48

TF_IDF26= 0.25*0.48 = 0.12

Cinquième étape: preparation du vecteur:

Document 1

label	1_p1	2_p2	3_p3	4_p4	5_p5	6_p6	7_p7	.	.	.	.			15_p15

Document 2

label	16_p16	12_p12	1_p1	17_p25	18_p18	19_p19

Document3

label	22_p22	23_p23	24_p24	25_p25	26_p26	27_0.12	28_p28

Sixième étape: affectation de label aux vecteurs:

Si nous faisons l'apprentissage sur le classe Data Mining alors l'étiquetage sera comme suit

Document 1

1	1_p1	2_p2	3_p3	4_p4	5_p5	6_p6	7_p7	.	.	.	.			15_p15

Document 2

1	16_p16	12_p12	1_p1	17_p25	18_p18	19_p19

Document3

-1	22_p22	23_p23	24_p24	25_p25	26_p26	27_0.12	28_p28

3.3. Apprentissage

Pour déterminer les vecteurs de supports, SVM procède à des calculs de produits scalaires des vecteurs caractéristiques des textes, comme ces vecteurs n'ont pas la même taille, SVM light offre une fonctionnalité de normalisation pour les harmoniser.

L'apprentissage et la génération du modèle se fait en appelant la fonction « learn » de SVM light. C'est à cette étape qu'il est possible de préciser les différents paramètres de SVM, par défaut le noyau est linéaire et le taux d'erreurs autorisées est nul.

3.4. Classification

Cette phase consiste à utiliser le modèle de classification obtenu de la phase d'apprentissage pour classer les nouveaux textes. Les textes sont soumis au même traitement de préprocessing pour obtenir le vecteur caractéristique. Par la suite nous appelons la fonction « classify » de SVM Light sur ce vecteur, la valeur retournée par cette fonction correspond à la marge qui sépare le texte de la frontière entre les deux catégories, si elle est positive, le texte est considéré comme appartenant à la catégorie étudiée et si elle est négative le texte est considéré comme non appartenant à la catégorie étudiée.

Pour le cas multi classes, nous avons préparés autant de modèles de classification que de catégories étudiées, et par la suite nous avons soumis les textes à classer aux différents algorithmes de classification générés par ces modèles, le texte est considéré appartenant à toutes les catégories dont le modèle correspondant a donné une marge positive.

3.5. Intégration dans ZONE

L'architecture de l'application ZONE est basée sur le système de plugins, chacun des plugins est responsable d'une méthode de classification. Un plugin dédié à SVM a été développé et a permis d'intégrer facilement notre programme dans le code source de ZONE. Nous nous sommes basés sur l'outil de gestion de source github pour cela. Les tests de la classification des news en français ont été réalisés directement sur les flux RSS annotés par le serveur de ZONE.

4. Expérimentation et Résultats

4.1. Corpus de test

4.1.1. Anglais

Pour tester notre programme nous avons besoins d'un ensemble de jeu de données pour faire l'apprentissage de l'algorithme et d'un autre pour la classification. Ces deux ensembles doivent être classées en différentes catégories à l'avance pour :

- Pouvoir attribuer les labels aux textes dans la phase apprentissage
- Pouvoir vérifier les résultats de la classification.

Cette problématique a fait l'objet de plusieurs études et il existe des corpus de textes prêts pour cela. Nous avons considéré dans le cadre de ce projet le corpus Reuters-21578. Ce corpus a été adapté au problème de catégorisation suite aux travaux de Moschitti [13]. Il en résulte la décomposition des textes en deux dossiers « training » pour l'apprentissage et « test » pour la classification. Chacun de ces deux dossiers contient 90 sous dossiers représentants les 90 catégories du corpus. Et chacun de ces sous dossiers contient au moins un texte. Le pourcentage des textes d'apprentissage et de test pour une catégorie donnée est pratiquement le même, la décomposition respecte ainsi la condition de distribution identique requise. Le nombre total de textes dans le dossier training est de 11413 et celui dans le dossier test est de 4024.

Toutefois cette décomposition a fait l'objet de plusieurs études et de désaccord entre les scientifiques, plusieurs autres sont proposées (115 ou 135 catégories).

Les tests réalisés dans ce projet ont porté sur les 10 catégories les plus significatives (contenant le plus de textes) de Reuters, il s'agit de « money-fx », « earn », « acq », « grain », « crude », « trade », « interest », « ship », « wheat », « corn ».

4.1.2. Français

Pour la langue française le corpus de test a été construit à la main à partir de news extraites du serveur ZONE. Là aussi nous avons respecté la condition de distribution identique de données de test et d'apprentissage, le corpus français est de plus petite taille et comporte 5 catégories, le tableau suivant résume sa composition :

	learning	test	différence (Learning+test)/test
santé	56	21	36%
informatique	274	104	36%
sport	721	281	35%
économie	32	12	36%
science	28	11	35%
total	1111	429	36%

4.2. Métriques d'évaluation

Trois grandeurs sont utilisées pour évaluer la classification faite par notre algorithme, il s'agit du « Rappel » de la « Précision » et de la « F-Measure », ces grandeurs sont définies comme suit :

Rappel : c'est le nombre de textes associés correctement à une catégorie divisé par le nombre total des textes appartenant réellement à cette catégorie.

$$\text{Rappel} = \frac{\text{nombre de documents bien classés à une catégorie i}}{\text{nombre de documents appartenant réellement à la catégorie i}}$$

Précision : c'est le nombre de textes correctement attribués à une catégorie divisé par le nombre total de textes attribués à cette même catégorie,

$$\text{Précision} = \frac{\text{nombre de documents bien classés à une catégorie i}}{\text{nombre de documentsclassés dans la catégorie i}}$$

Pour un problème multi classes, le rappel et la précision se calculent comme étant les moyennes respectives des rappels et des précisions de tous les classifieurs SVM utilisés.

F-measure: cette grandeur est une combinaison entre précision et rappel. Elle est définie comme suit :

$$\text{F} - \text{Measure} = \frac{2 \times \text{rappel} \times \text{précision}}{\text{rappel} + \text{précision}}$$

4.3. Résultats et interprétation

4.3.1. Anglais

Vu la grande taille du corpus Reuters, la partie préprocessing a été très longue, notamment à la phase calcul des poids des lemmes qui a duré environ 3 heures pour les 11413 textes d'apprentissage et environ 1 heure et 20 minutes pour les 4024 textes de test. Comme ce calcul se répète identiquement pour chaque test effectué, nous avons modifié l'algorithme pour stocker ces résultats dans fichier et les charger dans la mémoire à chaque utilisation, cela nous a permis un gain de temps énorme, en effet les tests d'apprentissage et de classification ne nécessitaient plus que quelques secondes.
La librairie SVM Light permet de calculer automatiquement le taux d'erreurs autorisées (la constante C), pour notre cas SVM light a trouvé pour cette constante la valeur 13.492.

Les résultats des premiers tests sont résumés dans le tableau suivant :

	Nombre de textes d'apprentissage	Nombre de textes de test	Nombre de textes classés	Nombre de textes bien classés	rappel	précision	F-measure
money-fx	538 (4,4 %)	179 (4,4 %)	46	36	0,20	0,78	0,32
earn	2877 (25,2 %)	1087 (27 %)	1071	1044	0,96	0,97	0,96
acq	1650 (14,4 %)	719 (17,8 %)	572	548	0,76	0,95	0,84
grain	433 (3,7%)	149 (3,7 %)	2	1	0,006	0,5	0,01
crude	389 (3,4 %)	189 (4,6%)	109	104	0,55	0,95	0,7
trade	396 (3,2%)	117 (2,9 %)	40	39	0,33	0,97	0,49
interest	347 (3 %)	131 (3,2 %)	36	36	0,27	1	0,42
ship	197 (1,7 %)	89 (2,2 %)	16	16	0,17	1	0,29
wheat	212 (1,8 %)	71 (1,7 %)	0	0	0	NA	NA
corn	181 (1,5 %)	56 (1,3 %)	0	0	0	NA	NA

Nous remarquons ici que les valeurs trouvées pour le rappel sont faibles à l'exception des catégories « earn » et « acq », ceci s'explique par le fait que ces deux catégories sont les plus grandes du corpus (respectivement 25 % et 14 %).
L'algorithme n'a pas pu classer aucun texte pour des catégories faiblement représentées telles que « wheat » ou « corn ».
La précision dans ce premier cas est en général élevée et a atteint même 100 % pour des catégories comme « interest » ou « ship » (nombre de textes faibles pour ces catégories).

Pour essayer d'améliorer ces résultats, nous avons joué sur le paramètre J (qui définit le rapport entre le taux de pénétration de la marge pour les textes positifs et le taux pour les textes négatifs), en prenant une valeur de J égale à 10, nous avons obtenus les résultats suivants :

	Nombre de textes d'apprentissage	Nombre de textes de test	Nombre de textes classés	Nombre de textes bien classés	rappel	précision	F-measure
money-fx	538 (4,4 %)	179 (4,4 %)	197	130	0,72	0,66	0,69
earn	2877 (25,2 %)	1087 (27 %)	1110	1057	0,97	0,95	0,96
acq	1650 (14,4 %)	719 (17,8 %)	635	593	0,82	0,93	0,87
grain	433 (3,7%)	149 (3,7 %)	138	115	0,77	0,83	0,8
crude	389 (3,4 %)	189 (4,6%)	187	151	0,79	0,80	0,79
trade	396 (3,2%)	117 (2,9 %)	123	82	0,70	0,66	0,68
interest	347 (3 %)	131 (3,2 %)	113	88	0,67	0,77	0,72
Ship	197 (1,7 %)	89 (2,2 %)	72	60	0,67	0,83	0,74
wheat	212 (1,8 %)	71 (1,7 %)	65	49	0,69	0,75	0,72
corn	181 (1,5 %)	56 (1,3 %)	45	32	0,57	0,71	0,63

La valeur du rappel s'est nettement améliorée pour toutes les catégories et on a atteint une moyenne de presque 70 % (mais les plus grandes catégories gardent toujours les meilleures valeurs pour le rappel).
Par contre on a perdu légèrement en précision, cela s'explique par l'augmentation du rappel et donc du nombre de textes classés dans la catégorie étudiée. Toutefois les valeurs de la précision restent dans les 70 % à 95 %.

Nous avons examiné de plus près la catégorie « corn » pour essayer des comprendre les raisons pour lesquelles des textes sont mal classés. Nous remarquons pour le cas où j=10 qu'il y a 13 textes qui sont classés comme appartenant à corn alors qu'ils ne le sont pas. Ces textes sont : 0010328 0011042 0012881 0010318 0010762 0010320 0012637 0011802 0011641 0010067 0010891 0011776 0012083.
Si nous prenons l'exemple du texte 0010328 nous remarquons que ce texte :

- Contient peu de mots (17 seulement)
- contient des mots qui se trouvent déjà dans les textes de la catégorie « corn » tels que « tonne», « export» (« export » se trouve par exemple dans des textes de cette classe comme par exemple 0009606 (3 fois), 0009622 (4 fois) et 0010141 (2 fois) et « tonne » se trouve par exemple dans 0009742 (7 fois), 0009780(4 fois), 0009888 (5 fois) et 0009926 (3 fois)...). Et comme le nombre de mots de ce texte est petit, ces deux mots ont un poids important ce qui a causé l'erreur de classification.
- le texte 0010328 appartient à une catégorie qui s'appelle « soy-meal ». Cette catégorie est proche de « corn », elles contiennent beaucoup de mots communs ce qui engendre une confusion lors du classement.

Si nous prenons l'exemple du texte 0009606 nous remarquons que ce textes doit être classé parmi les textes appartenant à la catégorie « corn », or ce n'est pas le cas. Ceci est expliqué par le fait que l'ensemble des textes de l'apprentissage de cette catégorie et très petit par rapport à l'ensemble de tous les textes de l'apprentissage (1,5 %). le modèle préparé ne couvre donc pas tous les mots caractéristiques de cette catégorie.

4.3.2. Français

5. Conclusion

Dans ce projet nous avons enrichi le serveur d'annotation de ZONE par une nouvelle méthode de classification de news basée sur un algorithme d'apprentissage supervisé qui est SVM. Le programme réalisé se base sur la librairie SVM light et se compose d'une partie apprentissage et d'une partie de classification. Ces deux parties sont nécessairement précédées d'une étape de préprocessing de textes qui consiste à extraire les lemmes des mots des textes et à calculer leur poids avec la méthode TF-IDF, et ce afin de préparer les vecteurs caractéristique des textes nécessaires pour SVM Light.
Pour valider notre implémentation nous avons utilisé le corpus Reuters pour l'anglais et un corpus construit manuellement pour le français. Nous avons calculé la précision, le rappel et le F-measure pour différentes catégories. Notre algorithme a montré des résultats satisfaisants.

Néanmoins il est possible d'apporter d'autres optimisations à notre programme en travaillant sur l'étape de préprocessing pour la sélection des vecteurs caractéristiques des textes. Plusieurs travaux de recherche dans ce sens existent et peuvent nous être utiles. [14] [15]

6. Références bibliographiques

[1] Kamilia MENGHOUR, Labiba SOUICI-MESLATI Selection de caractéristiques pour le filtrage de spam
[2] Sylvie Szulman, LIPN, Université Paris 13 Construction de ressources sémantiques à partir de textes
[3] Guillaume Cleuziou 2004 une méthode de classification non supérvisé pour l'apprentissage des règles et la recherche d'information
[4] Kiri Wagsta Claire Cardie 2001 Constrained K-means Clustering with Background Knowledge
[5]Jean-Pièrre Nakache Josian Confais Approche de classification pragmatique
[6] Cours_KIS_2012-2013_SVM
[7] Yiming yang Xin Lui , A re_examination of text categorization method
[8]Thorsten Joachims transductive interface for text classification using support vector machines
[9]J. Weston c. Watkins Multi class support vector machines
[10]http://svmlight.joachims.org/
[11] http://www.mpi-inf.mpg.de/~mtb/
[12]http://nlp.stanford.edu/software/corenlp.shtml
[13] http://disi.unitn.it/moschitti/corpora.htm
[14]Sam Scott Stan Matwin Feature Engineering for Text Classification
[15] Indra Mahadevan Selvakuberan Karuppasamy Rajaram Ramasamy
Resource Optimization in Automatic web page classification using integrated feature selection and machine learning