UNIVERSITE DE YAOUNDE I
****
ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE DE YAOUNDE
****
DEPARTEMENT DE GENIE INFORMATIQUE

UNIVERSITY OF YAOUNDE I
****
NATIONAL ADVANCED SCHOOL OF ENGINEERING
****
DEPARTMENT OF COMPUTER ENGINEERING

Mise sur pied d'une plateforme de gestion intelligente et automatique de la qualité de données. Cas pratique : capture et classification des données de Wariba Classifieds

Mémoire de fin d'études/Master of Engineering

Présenté et soutenu par
MBA MBOGNE Stéphane Brandon

En vue de l'obtention du :
Diplôme d'Ingénieur de Conception de Génie Informatique

Sous la direction de :
Thomas BOUETOU BOUETOU, Professeur
Philippe NGIMBIS NZOGHE

Devant le jury Composé de :

Président :	**Thomas TAMO TATIETSE, Professeur, UYI**
Rapporteur :	**Thomas BOUETOU BOUETOU, Professeur, UYI**
Examinateur :	**Toussile WILSON, Chargé de cours, UYI**
Invité :	**Philippe NGIMBIS NZOGHE, Directeur Technique, WARIBA Classifieds**

Année Académique 2016-2017
Mémoire Soutenu le 13 Juillet 2017

Dédicaces

Je dédie ce mémoire à :

- mes parents parce qu'ils m'ont donné la vie et ont fait de moi qui je suis
- mes frères et soeurs pour leur contribution à mon édification personnelle
- ma grande famille pour son hospitalité
- à mes camarades de classe
- à toute l'équipe **Hello World**

Remerciements

J'addresse mes remerciements

- Au **Pr Thomas TAMO TATIETSE** pour l'immense honneur qu'il nous fait en présidant ce jury ;
- Au **Dr Wilson TOUSSILE** pour l'honneur qu'il nous fait en examinant ce mémoire
- Au **Pr Thomas BOUETOU BOUETOU** notre encadreur académique, pour sa disponibilité dans le suivi de la réalisation de ce travail ;
- A la société **WARIBA CLASSIFIEDS SAS** par la voix de son Directeur Général, **M. Nino NJOPKU**, qui nous a fourni un cadre de travail agréable pour la réalisation de notre stage ingénieur ;
- A notre encadrant professionnel **M. Philippe NGUIMBIS NZOGHE** Directeur Technique au sein de l'entreprise WARIBA CLASSIFIEDS SAS, pour son expertise et ses conseils précieux et généreux ;
- A tous les enseignants de l'**ENSP**, plus particulièrement ceux du Département du Génie Informatique, pour toutes les notions qu'ils nous ont inculquées ;
- A tout le personnel de WARIBA CLASSIFIEDS SAS pour son accueil chaleureux

Je remercie le Seigneur, Dieu, revelé en Jésus Christ bâtisseur de toute âme et de toute oeuvre noble, parce qu'il m'a apporté le soutien et la force nécessaire à la réalisation de cette oeuvre.

Résumé — De plus en plus de plateformes internet se sont tournées vers le web 2.0, avec un contenu essentiellement généré par les utilisateurs. Cette tendance donne l'avantage d'une rapidité dans la circulation de l'information qui devient de plus en plus diverse et intéressante pour les internautes. Cependant des problèmes liés à la qualité des informations qui circulent sur ces plateformes deviennent de plus en plus fréquents car elles sont ouvertes à tout le monde. Les arnaques, les propositions d'adhésion aux organismes et pratiques occultes, les contenus adultes sont très rencontrés sur ces plateformes. Alors pour y remédier, la solution très souvent est de mettre en place un système de modération manuelle où des personnes devront vérifier et valider les contenus publiés. Cela s'avère à la fois fatiguant pour les modérateurs et coûteux pour les entreprises, notamment les startups. Le présent travail propose une solution qui permet la gestion intelligente et automatique des contenus générés par les utilisateurs, afin d'éviter les dérives. Le système mis sur pied appelé MLA (Machine Learning Application) est capable de : détecter les contenus indésirables (suivant un ensemble de critères d'indésirabilité) et de classer automatiquement des contenus crées dans des catégories bien définies. Une autre partie toute aussi importante de notre travail, est la recherche de contenus dupliqués ou similaires, ce dans le but d'empêcher que les sites aient l'air spammés et de poser des actions commerciales. Cette contribution à l'automatisation des tâches est importante, puisqu'elle permet à l'entreprise d'optimiser l'utilisation de ses ressources humaines et produire dans l'ensemble de meilleurs résultats.

Mots clés : Apprentissage artificiel supervisé, classification supervisée, similarité, qualité de données.

Abstract — More and more internet plateforms are turning to web 2.0, with their content essentially produced by users. This trend has the advantage of speeding up the dissemination of information, which is becoming diverse and interesting for internet users. However, problems related to the quality of the information circulating on these platforms are becoming frequent because they are open to everyone. Scams, proposals for membership in hidden organizations and practices, adult content are very much encountered on these platforms. So to remedy this, the solution very often is to set up a manual moderation system where people will have to check and validate the published contents. This is both tiring for moderators and costly for businesses, especially startups. The present work proposes a solution which allows the intelligent and automatic management of the contents generated by the users, in order to avoid drifts. The system set up called MLA (Machine Learning Application) is capable of : detecting unwanted content (according to a set of undesirability criteria) and automatically classifying created content into well defined categories. Another important part of our work is the search for ads with duplicate or similar content, in order to put up commercial actions and prevent the site to look spam-my. This contribution to the automation of tasks is important, as it allows the company to optimize the use of its human resources and produce better overall results.

Key words : Supervised artificial learning, supervised classification, similarity, data quality.

Table des matières

Liste des tableaux

Table des figures

Liste des sigles et acronymes

API	Application Programming Interface
CSV	Comma-separated Values
EDI	Environnement de Développement Intégré
IDF	Inverse Document frequency
LSH	Locality sensitive hashing
MLA	Machine Learning Application
ORM	Object-Relational Mapping
SGBD	Système de Gestion de Base de Données
SVM	Séparateurs à Vaste Marge ou Support Vector Machine
Tf-Idf	Term Frequency - Inverse Document frequency

Introduction

Contexte

Wariba Classifieds est une entreprise spécialisée dans les sites de petites annonces et dispose actuellement de deux sites à son actif : kerawa.com et leportail.ci. Ce sont des sites de mise en relation entre vendeurs et acheteurs, leur principe étant que toute personne voulant vendre ou acheter un bien et/ou service publie son offre/demande sur le site afin de pouvoir être contacté par de potentiels intéressés. Chacun des sites a un grand volume d'annonces publiées (environ 120 000 pour leportail et plus de 200 000 pour kerawa) et cela à une fréquence de plus en plus croissante (jusqu'à 500 annonces publiées par jour par site).

Une telle activité (21 annonces par heure) est principalement due au fait que les sites sont disponibles sur plusieurs plateformes : navigateur desktop et mobile ainsi que via leurs applications mobiles Android. Les utilisateurs ont ainsi plusieurs moyens de publier leurs annonces, ce de manière libre et totalement gratuite. On a donc affaire à des sites de haut trafic dont le contenu est généré par les utilisateurs à destination d'autres utilisateurs. En tant que tel ceux-ci rencontrent de nombreux problèmes quant à la qualité des informations qui circulent sur les sites, mais surtout de celles qui sont stockées en bases de données. On rencontre beaucoup d'annonces qui présentent un contenu indésirable ainsi que celle qui sont dupliquées.

Pour pallier à ce problème de qualité d'information sur les sites, un système de modération manuelle a été mis en place notamment sur le portail. Une équipe de modération d'environ 3 personnes s'occupe de parcourir régulièrement (depuis une interface d'administration) les annonces nouvellement créées afin de les valider ou les invalider. Cela représente une charge de 3h de travail par personne par jour. Il est donc question pour nous à l'issue de notre travail de l'annuler afin que ceux-ci puissent effectuer des tâches plus prioritaires.

Problématique

Outre les problèmes techniques liées au haut trafic sur les sites Wariba Classifieds, des problèmes tout aussi importants sont dénotés et ont fait l'objet de notre travail. Ce sont les suivants :

Problème des annonces indésirables. La liberté et la gratuité des services offerts par Wariba Classifieds sur ses différents sites font à ce que beaucoup d'utilsateurs créent des annonces déplacées qui provoquent l'irritation d'autres utilisateurs. Ils se plaignent de la présence sur le site d'annonces à caractère ésotérique ou qui on trait à la sorcellerie, le spiritualisme et autres pratiques occultes. D'autres déplorent des arnaques principalement liées aux voyages à l'étranger. De plus, l'entreprise en elle-même ne souhaite pas avoir pour le moment des annonces ayant trait aux rencontres adultes, au sexe ou à la pornographie.

Problème de catégorisation. Afin de rendre l'information facilement navigable pour l'utilisateur, et leur permettre de facilement obtenir les informations dont ils ont besoin, des catégories ont été définies et chaque annonce se retrouve dans la (seule) catégorie qui lui correspond. Ainsi on dénote des catégories voiture, motos, terrain à vendre et bien d'autres. Le problème ici est que beaucoup d'annonces crées par les utilisateurs sont mal catégorisées par les utilisateurs qui les créent, soit par manque de vigilance, soit tout simplement par négligence de l'importance de la notion de catégorie. On se retrouve ainsi avec des annonces de vente de terrain par exemple qui se retrouvent dans la catégorie automobile.

Détection des annonces similaires et dupliquées. Le but ici est de savoir quelles sont les annonces dans l'ensemble des données de Wariba Classifieds qui ont des contenus identiques ou similaires à une annonce donnée dans le but d'une part de bloquer les annonces dupliquées, d'autre part de mener des actions marketing et commerciale futures par la recherche automatique d'annonces similaires à une annonce donnée.

Des problèmes présentés ci-dessus, nous remarquons la nécessité d'une gestion intelligente et automatique de la qualité des données qui sont produites par les utilisateurs.

Objectifs

La problématique ainsi définie, l'objectif principal de notre travail est de bâtir une plateforme qui permettra la gestion de la qualité des annonces sur les sites Wariba. Elle devra répondre aux attentes suivantes :

- Elle devra être capable de différencier une annonce indésirable (spam) de celle qui ne l'est pas étant données les contraintes évoquées dans la problématique.
- Pour toute annonce sur l'un des sites Wariba (notamment lorsqu'elle est créée), elle devra automatiquement être affectée à la catégorie qui lui correspond le mieux dans le site en question. Il est pour cela important de noter que chaque site a ses catégories qui lui sont propres.
- La plateforme devra, pour toute annonce sur l'un des sites Wariba, pouvoir automatiquement détecter si une annonce déjà créée lui est identique ou similaire

Plan du mémoire

Le mémoire est structuré autour de cinq chapitres. Dans le premier chapitre nous fairons une **analyse de la problématique** posée dans l'introduction afin de définir les axes de recherche qui seront explorés dans le deuxième chapitre qu'est l'**état de l'art**. Au chapitre 3 nous présenterons les **méthodologies** choisies pour parvenir à la résolution des problèmes concernés par chacun des axes de recherche de l'état de l'art. Ensuite, au chapitre 4 nous présenterons l'**implémentation** faîte des solutions trouvées avant de faire un point chapitre 5 sur les **résultats obtenus**.

Chapitre 1

Analyse de la problématique

Les trois problèmes étant posés, nous avons débuté notre travail par une première étape d'analyse de ces besoins afin d'identifier les solutions potentielles qui peuvent être implémentées pour atteindre nos objectifs.

1.1 Détection de spams

Pour le cas de la détection d'annonces indésirable, l'objectif est de pouvoir détecter automatiquement les annonces dont le contenu ne respecte pas les règles des sites Wariba. Il faudrait détecter :

- Les annonces qui ont trait à l'ésotérisme et toute autre forme de pratique magique, mystique ou spirituelle.
- Les annonces à caractère érotique, faisant référence au sexe, aux rencontres adultes ou à la pornographie
- Les annonces d'anarque principalement des offres d'emplois accrédités à des organismes internationaux tel que l'ONU, des propositions de voyage/travail à l'étranger, proposition de gain d'argent facile (très souvent via internet), des recrutements dit "massifs", des bourses d'études. Pour la plupart des annonces de ce type publiées sur les sites, ce sont des arnaques.

Résoudre ce problème consiste à dire pour une annonce, étant donné son contenu (titre et description) si celle-ci est une annonce indésirable ou pas. Ce qui revient à prédire pour une annonce dont on a le contenu, si celle-ci est un spam ou pas.

Il s'agit d'un problème de **classification** et puisque nous connaissons déjà à l'avance les catégories dans lesquelles la classification doit se faire ainsi que les critères d'appartenance à ces catégories, on a donc affaire à un problème de **classification supervisée**.

Un aspect important de ce problème c'est que la solution mise sur pied doit être en mesure

de détecter une annonce spam même si elle ne l'a jamais vu avant. Ceci implique de l'**apprentissage artificiel** (en anglais machine Learning) dans le but de pouvoir prédire pour une annonce totalement nouvelle quelle étiquette (*spam* ou *non spam*) doit lui être attribuée.

1.2 Catégorisation des annonces

Notre plateforme devra pouvoir automatiquement affecter une annonce de l'un des sites Wariba à la catégorie qui lui correspond le mieux dans le site. Sur leportail.ci, on dénombre 58 catégories différentes et kerawa.com en a 88. Voici quelques exemples de catégories que l'on peut retrouver sur chacune des plateformes.

leportail.ci	kerawa.com
Billetteries	Accessoires de mode
Chaussures	Assurances
Cours particuliers ou Formation	Bureaux - locaux commerciaux
Electroménager	Equipement industriel
Fourniture de Bureau	Formations professionnelles
Instrument de musique	Montres - Bijoux - Lunettes
Livres	Motos
Locations	Offre de stage
Motos	Ordinateurs
Vins et gastronomie	Téléphones - tablettes tactiles
Voitures	Voitures

TABLE 1.1 – Exemples des catégories sur les sites leportail.ci et kerawa.com de WARIBA Classifieds

La résolution de ce problème consiste aussi, comme la détection de spam, à prédire pour une annonce dont on a le contenu à laquelle des catégories elle appartient. Il s'agit d'un problème de **classification**. Les catégories étant connues d'avance c'est donc un problème de **classification supervisé**, tout comme le problème de détection de spam. Avec la nuance qu'ici, le nombre de catégorie est plus élevé.

1.3 Similarité et Annonces dupliquées

Le souci est de pouvoir détecter de manière automatique, pour une annonce dont on a le contenu si il existe une ou plusieurs autres avec un contenu identique ou similaire ; ce sur l'ensemble des données du site où l'annonce a été publiée. Cela consiste à retrouver les

annonces qui lui sont les plus 'proches' et correspond ainsi à un problème de plus proche voisin.

1.4 Conclusion

Les problématiques ayant été identifiées, nous les avons traduites d'un point de vue technique afin d'identifier les solutions qui peuvent être implémentées dans le but de les résoudre. En somme, nous nous rendons compte que pour les trois besoins exprimés au départ, la recherche de solution devra se faire sur deux principaux axes : la **classification supervisée** et la **recherche de plus proches voisins**. Dans la partie qui suit, nous détaillerons les résultats de nos recherches.

Chapitre 2

Etat de l'art

Dans cette partie, nous présentons les solutions envisageables afin de répondre aux attentes de Wariba. Elle se divise en deux principales parties : la **classification supervisée** et la **recherche de plus proche voisin**. Ces parties correspondent aux deux axes de recherche identifiés pendant la phase d'analyse de la problématique. Après ces deux axes, nous décrirons des méthodes de calcul de distance entre textes. Mais avant de présenter ces trois parties nous ferons un point sur l'**extraction de caractéristique**.

Les algorithmes utilisés pour faire de la classification supervisée ou la recherche de plus proches voisins travaillent avec des valeurs numériques. Or les annonces ont un contenu essentiellement textuel. Une première étape avant de pouvoir utiliser ces algorithmes est de transformer le texte en vecteur. Nous présenterons des techniques utilisées dans la littérature pour pouvoir le faire.

2.1 Extraction de caractéristiques

Il existe trois principale méthodes de transformation de corpus textuel en vecteur de réels :

- **Occurence des mots** : qui consiste à construire un vecteur à partir d'un vocabulaire de mots et du nombre de fois que chaque mot apparaît dans le texte qu'on veut transformer
- **Fréquence de mots** : qui lui est construit à partir d'un vocabulaire et de la fréquence de chaque mot du vocabulaire dans le texte
- **Tf-Idf** : qui a pour objectif de produire une représentation du texte où les mots spécifiques au texte à transformer se voient donné une importance plus grande dans le vecteur final.

2.1.1 Occurrence des mots

Pour transformer un texte en vecteur on peut tout simplement compter le nombre d'occurrence de chaque mot du vocabulaire dans le texte en question [Buitinck 2013]. Une première étape c'est de constituer le vocabulaire qui contient l'ensemble des mots susceptible d'apparaître dans les textes à transformer. Ensuite, pour chaque texte, son vecteur est construit en fonction de si oui ou pas chaque mot du vocabulaire appartient au texte. La dimension des vecteurs obtenus est le nombre de mots du vocabulaire.

Les **annonces sur les sites kerawa sont écrites en langue française** et celle-ci comporte jusqu'à 80 000 mots [OQLF] ; ce qui reviendrait à construire des vecteurs de dimension 80 000, or très souvent beaucoup de ces mots ne sont pas utilisés. De plus les mots de langue française n'incluent pas toujours ceux employés par les utilisateurs locaux, à l'exemple des noms de lieux ou de plats. Une astuce alors employée est de bâtir notre vocabulaire à partir d'un ensemble de texte suffisamment représentatif du problème qu'on veut résoudre.

Ainsi pour l'ensemble des textes :

T1 : voiture à vendre à Makepe ;

T2 : restaurant spécial taro à Melen.

On aura les représentations suivantes

	à	Makepe	Melen	restaurant	spécial	taro	vendre
T1	2	1	0	0	0	0	1
T2	1	0	1	1	1	1	0

2.1.2 Fréquence de mots

[Buitinck 2013] Le principe de base est le même qu'avec les occurrences de mots, on construit le vecteur en fonction des mots du vocabulaire présents dans le texte. La différence est qu'au lieu du nombre de mot c'est plutôt la fréquence qui est enregistrée pour ramener toutes les valeurs à la même échelle. Et pour les mêmes textes :

T1 : voiture à vendre à Makepe ;

T2 : restaurant spécial taro à Melen.

On obtient :

	à	Makepe	Melen	restaurant	spécial	taro	vendre
T1	$\frac{2}{4}$	$\frac{1}{4}$	0	0	0	0	$\frac{1}{4}$
T2	$\frac{1}{5}$	0	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	0

Ces deux techniques présentent cependant des limites. Les valeurs dans chaque dimension du texte vectorisé représentent implicitement l'importance qu'à le mot concerné dans la représentation vectorielle du texte [Luhn 1957]. Ainsi, d'après les vectorisations faites

jusque là, le mot **à** a une plus grande importance que le mot **Makepe** dans le Texte 1. Ce qui est un problème, parce que ce mot n'est qu'un mot usuel de la langue française, à l'exemple des articles et autres préposition. Il est donc important que dans l'extraction de caractéristiques, l'importance de ce type de mots soit réduite. Et c'est de là que naît la technique **tf-idf**.

2.1.3 Tf-Idf

Tf-Idf pour *Term frequency – Inverse Document Frequency*. Cette technique va du principe qu'il faut pénaliser les mots qui apparaissent dans beaucoup de textes [Rajaraman 2011]. On calcule toujours la fréquence du mot dans le texte comme dans le cas précédent mais aussi l' IDF (Inverse Document Frequency) selon la formule :

$$IDF(m) = \log(\frac{\text{Nombre total de textes}}{\text{Nombre de textes où le mot } m \text{ apparaît}}) \tag{2.1}$$

Le logarithme étant utilisé ici comme une heuristique.

L'IDF caractérise la spécifité du mot par rapport au document car étant inversément proportionnel au nombre de document dans lequel le mot apparaît [Jones 1972] []

Ainsi pour les textes

T1 : voiture à vendre à Makepe ;

T2 : restaurant spécial taro à Melen.

On a les IDF suivants :

à	Makepe	Melen	restaurant	spécial	taro	vendre
$\log \frac{2}{2}$	$\log \frac{2}{1}$	$\log \frac{2}{1}$	$\log \frac{2}{1}$	$\log \frac{2}{1}$	$\log \frac{2}{1}$	$\log \frac{2}{1}$
0	0.30	0.30	0.30	0.30	0.30	0.30

Dès lors on peut calculer les valeurs de tf-idf selon la formule :

$$tf - idf(m, T) = f(m, T) * IDF(m) \tag{2.2}$$

$f(m, T)$ étant la fréquence du mot m dans le texte T

Les résultats finaux sont les suivants :

	à	Makepe	Melen	restaurant	spécial	taro	vendre
T1	0	$\frac{3}{40}$	0	0	0	0	$\frac{3}{40}$
T2	0	0	$\frac{3}{50}$	$\frac{3}{50}$	$\frac{3}{50}$	$\frac{3}{50}$	0

2.2 Classification supervisée

L'apprentissage supervisé est un ensemble d'algorithmes qui à partir d'un ensemble d'exemples dit d'entraînement, essaient de trouver des hypothèses générales dans le but de faire des prédictions pour des instances futures. L'objectif est de bâtir un modèle qui va donner des labels de classes à des instances à prédire, à partir des caractéristiques de celles-ci [Kotsiantis 2007] .

Un problème d'apprentissage artificiel supervisé [Mehryar Mohri 2012] est défini par une base d'apprentissage B et une fonction f à estimer :
La base de d'apprentissage B est un ensemble de N couples entrée-sortie (x, c) avec $x \in X^p$ et $c \in C$, que l'on considère être tirées selon une loi sur $X^p \times Y$ fixe et inconnue. C est l'ensemble des classes auxquelles peuvent appartenir toute entrée x de X^p ; c'est un ensemble discret.

On suppose qu'il existe une fonction f tel que

$$\begin{array}{rccl} f: & X^p & \longrightarrow & Y \\ & x & \longmapsto & y = f(x) + w \end{array}$$

L'objectif de l'apprentissage est de la retrouver.

La méthode d'apprentissage supervisée utilise la base d'entraînement B pour déterminer une représentation de f qu'on peut noter g et appelée *fonction de prédiction*, *hypothèse* ou très souvent *modèle*. Ce modèle sera utilisé pour associer à une nouvelle entrée x_{new}, sa sortie $g(x_{\text{new}})$. Le but d'un algorithme d'apprentissage supervisé est donc de généraliser pour des entrées inconnues ce qu'il a pu « apprendre » grâce aux données de la base d'entraînement B.

La **classification supervisée** est un cas particulier d'apprentissage supervisé. Nous allons présenter quelques principes utilisés, pour résoudre les problèmes de classification. Dans notre revue de la littérature, nous présenterons trois principales méthodes de classification supervisée :

- **Le classifieur de Bayes Naïf** : qui est une approche de classification probabiliste basée sur la théorème de Bayes
- **Le classifieur par k plus proches voisins** : dont le principe est de donner la classe d'une entrée à prédire en fonction de celles de ses plus proches voisins dans la base d'apprentissage
- **Machine à vecteur support** : qui a pour principe de transformer l'espace de la base d'apprentissage en un espace de plus grande dimension et de diviser cet espace par des hyperplans selon les différentes classes existantes.

2.2.1 Classifieur de Bayes Naïf

[Liefooghe 2013] Considérons la base d'apprentissage à N éléments suivante $B = \{(x, c)\}$, $B \subset X^p \times C = \{c_1, c_2, ..., c_K\}$

$x = (x_1, x_2, ..., x_p)$ est un vecteur de caractéristiques encore appelée **variable explicative** avec $\forall i \in \{1, 2, ..., p\}, x_i \in \mathbf{R}$.

L'objectif du classifieur de bayes est d'apprendre $f = p(c|x)$, c'est à dire la probabilité d'appartenir à la classe c sachant qu'on a les caractéristiques x.

On a la formule de Bayes :

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)} \tag{2.3}$$

Le modèle naïf consiste à faire l'hypothèse que les descripteurs sont indépendants conditionnellement à la classe, d'où :

$$p(x|c) = \prod_{i=1}^{p} p(x_i|c) \tag{2.4}$$

La formule de Bayes donne donc :

$$p(c_k|x) = \frac{p(c_k)\prod_{i=1}^{p} p(x_i|c_k)}{p(x)} = \frac{p(c_k)\prod_{i=1}^{p} p(x_i|c_k)}{\sum_{j=1}^{K} p(c_j)\prod_{i=1}^{p} p(x_i|c_j)} \tag{2.5}$$

où j est l'indice de la classe ($j \in \{1, 2, ..., K\}$), i l'indice de la variable explicative et k une classe.

La classe prédite est celle qui maximise la probabilité conditionnelle $p(c_k|x)$ c'est à dire :

$$g(x) = argmax_{c_k \in C} \frac{p(x|c_k)p(c_k)}{p(x)} \tag{2.6}$$

Quelques observations : $p(c_k|x) = \frac{p(x|c_k)p(c_k)}{p(x)}$

- $p(c_k|x)$ croît quand $p(c_k)$ croît : plus c_k est probable, plus il y'a des chances qu'elle soit la classe
- $p(c_k|x)$ croît quand $p(x|c_k)$ croît : si $(x_1, x_2, ..., x_n)$ arrive souvent quand c_k est la classe, alors il y'a des chances que c_k soit la classe
- $p(c_k|x)$ décroît quand $p(x)$ croît : si $(x_1, x_2, ..., x_p)$ est courant, il nous apprend peu sur c_k

2.2.2 Classifieur par k plus proches voisins

[Tretyakov 2004] Pour la base d'apprentissage à N éléments suivante $B = \{(x, c)\}$, $B \subset X^p \times C = \{c_1, c_2, ..., c_K\}$, supposons qu'il existe une distance entre les vecteurs $x \in X^p$. On serait ainsi capable de dire si deux vecteurs sont "proches" l'un de l'autre ou pas. C'est sur cette base que fonctionne le classifieur par k plus proches voisins : pour définir la classe à laquelle appartient une nouvelle entrée x_{new} on se base sur celle de ses plus proches voisins, dans la base d'apprentissage B.
Pour une nouvelle entrée x_{new}, il faut déterminer ses k plus proches voisins parmi les exemples d'apprentissage de B. De là on a

$$g(x_{\text{new}}) = argmax_{c_k \in C} \sum_{i \in V_k(x_{\text{new}})} w_i(x_{\text{new}}) \tag{2.7}$$

où $V(x_{\text{new}})$ est l'ensemble des k plus proches voisins de x_{new} qui sont de la classe c_k. $w_i(x_{\text{new}})$ est une valeur inversément proportionelle à la distance qu'il y'a entre x_{new} et l'élément i de V_k

Dans cette méthode, il n'ya pas de réelle phase d'"entraînement". Néanmoins, pour ne pas avoir à parcourir la base d'apprentissage B à chaque fois qu'il faut prédire la classe d'une entrée, une indexation de cette base est faîte afin de rendre la recherche des plus proches voisins peu coûteuse en temps. Nous présenterons ces techniques dans la partie liée aux **Similarités et détection de contenus dupliquées**.

2.2.3 Machine à Vecteur Support

Encore appelées SVM (*Séparateurs à Vaste Marge* ou *Support Vector Machine*) elles ont été introduites par Vapnik [Vladimir Vapnik 1998]. Leur principe est le suivant : les entrées $x \in X^p$ sont non-linéairement transformées dans un espace $\mathcal{H}$ de très grande dimension. Dans ce nouvel espace, on part du principe que les classes de $C = \{c_1, c_2, ..., c_n\}$ peuvent être séparées par des hyperplans [Richard O. Duda 2001]. L'entrainement consistera donc à trouver les hyperplans séparateurs des différentes classes et trouver les conditions d'appartenance aux différentes classes sur la base des hyperplans trouvés.

[Clémençon 2011] A la base les techniques SVM sont utilisés pour des problèmes de classification binaire (c'est à dire $\text{card}(C) = 2$). Elles font appel à une fonction ϕ transformant l'espace d'entrée X^p en un espace $\mathcal{H}$ de plus grande dimension et muni d'un produit scalaire $\langle ., . \rangle$. L'apprentissage s'effectue alors à partir des $(\phi(x), c)$ dans l'espace $\mathcal{H}$, de dimension plus grande certes, mais dans lequel on espère que les données soient "davantage linéairement séparables".

Du point de vue pratique, le calcul des projections $\phi(x)$ n'est pas utilisé dans la méthode, seuls les produits scalaires $\langle \phi(x), \phi(x') \rangle$, $(x, x') \in X^p \times X^p$ sont requis. Or, ceux-ci sont

donnés par un noyau K, via la relation :

$$K(x, x') = \langle \phi(x), \phi(x') \rangle \tag{2.8}$$

Le noyau peut être :
- Linéaire : $K(x, x') = \langle x, x' \rangle$
- Gaussien radial : $K(x, x') = \exp(-\sigma \|x - x'\|)$
- Polynomial : $K(x, x') = (\alpha + \beta \langle x, x' \rangle)^{\delta}$
- Sygmoïde : $K(x, x') = \tanh(\alpha + \beta \langle x, x' \rangle)$

Pour une nouvelle entrée x_{new}, on aura $g(x_{\text{new}})$ de la forme :

$$g(x_{\text{new}}) = \text{sgn}(\langle w, \phi(x_{\text{new}}) \rangle + b) \tag{2.9}$$

où $w \in \mathcal{H}$ et $b \in \mathbf{R}$ sont des paramètres ajustés lors de la phase d'apprentissage. L'hyperplan séparateur a pour équation $\langle w, \phi(x_{\text{new}}) \rangle + b = 0$ et est obtenu en maximisant la **marge** (distance entre l'hyperplan et les échantillons les plus proches). Ceci revient à résoudre le problème d'optimisation suivant

$$\text{minimiser} \frac{1}{2} \|w\|^2 + \frac{C}{m} \sum_{i=1}^{n} \xi_i \tag{2.10}$$

sous les contraintes : $\forall i \in \{1, ..., n\}$,

$\xi_i \geq 0$ et $c_i(\langle w, \phi(x_{\text{new}}) \rangle + b) \geq 1 - \xi_i$

La résolution de ce problème constituera la phase d'**entraînement du classifieur**.

La solution w s'exprime

$$w = \sum_{i=1}^{n} \alpha_i c_i \phi(x_i), \tag{2.11}$$

les indices i pour lesquels $\alpha_i \neq 0$ sont ceux pour lesquels l'égalité est réalisée dans la contrainte, les x_i correspondants sont appelés vecteurs supports. . Les coefficients α_i désignent les solutions du problème quadratique dual :

$$\text{maximiser} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{1 \leq i, 1 \leq n} \alpha_i \alpha_j c_i c_j K(x_i, x_j) \tag{2.12}$$

sous les contraintes : $\forall i \in \{1, ..., n\}$,

$$0 \leq \alpha_i \leq \frac{C}{n} \text{et} \sum_{i=1}^{n} \alpha_i c_i = 0 \tag{2.13}$$

Le paramètre C contrôle la complexité du classifieur dans la mesure où il détermine le coût d'une mauvaise classification : plus C est grand, plus la règle obtenue est complexe (le nombre de point pour lequel on veut minimiser l'erreur de classification croît).

Classification Multi classes

Comme nous l'avons souligné plus haut, les techniques SVM sont utilisés pour des problèmes de classification binaire, card$C = 2$ en général $C = \{-1, 1\}$. Néanmoins, il est possible de généraliser ce principe pour faire de la classification avec card$C > 2$. Il existe deux principales méthodes pour le faire [Chih-Wei Hsu]

- **Un contre un** on considère toutes les paires de classes possibles $(c_k, c_{k'})$ et on fait l'entraînement pour chacune de ces paires. On aura ainsi C_K^2 modèles $g_{k,k'}$. Pour toutes nouvelle entrée x_{new}, sa classe sera celle qui aura été le plus prédit par les $g_{k,k'}$.
- **Un contre tous** Pour chaque modalité c_k, on apprend un classifieur permettant de discriminer entre les populations $c = c_k$ et $c \neq c_k$. A partir des estimations des probabilités a posteriori, on affecte la classe estimée la plus probable.

2.3 Similarité et détection de contenu dupliqué

La recherche des plus proches voisins consiste à retrouver pour un point donné, les k points qui lui sont le plus proche dans un ensemble d'autres points. Ainsi pour un ensemble B_s de N_s éléments $x_s \in X^p$, il s'agit pour une nouvelle entrée x_{new} de trouver dans B_s les k éléments qui sont les plus proches de x_{new} selon une distance définie. Il existe plusieurs techniques pour déterminer ces plus proches voisins.

2.3.1 Algorithme dit naïf de recherche de plus proche voisin

Encore appelé algorithme naïf, son principe est de parcourir l'ensemble des N_s points de de B_s et pour chaque point, on vérifie qu'il est plus proche ou non qu'un des plus proches voisins déjà sélectionné. S'il est plus proche, on l'insère dans l'ensemble des k plus proches voisins, sinon on ne fait rien. L'algorithme est le suivant :

input : Tableau T des N_s points de B_s et q le vecteur dont on veut extraire les k plus proches voisins
output: Tableau T_k des k plus proches voisins

for $i \leftarrow 1$ **to** k **do**
  Mettre le point $T[i]$ *dans* T_k;
end
for $i \leftarrow k+1$ **to** N_s **do**
  if *la distance entre* q *et* $T[i]$ *est inférieure à la distance d'un des points de* T_k *à* q **then**
    Supprimer de T_k *le point le plus éloigné de* q *Mettre dans* T_k *le point* $T[i]$
  end
end

Algorithme linéaire de plus proche voisin

Cette méthode bien que facile à implémenter présenter le désavantage de s'éxécuter en temps linéaire $O(n)$. A chaque fois qu'il faudra trouver les plus proches voisins d'une entrée q il est nécessaire de parcourir toute la base B_s.

2.3.2 Division de l'espace en arbre K^d

Cette méthode consiste à partionner l'espace de l'ensemble des N_s points de B_s afin de rendre le processus de recherche de plus proche voisin moins coûteux en temps [Bentley 1975]. Les points sont ainsi stockées dans une structure de donnée adéquate

appelée $K^d - Tree$ (arbre K^d) qui divise par l'espace de manière hiérarchique par des hyperplans. Le nom $K^d - Tree$ vient du fait que la structure de donnée est construite pour des vecteurs de dimension K. C'est un arbre où chaque noeud est un point de B_s par lequel passe un (au plus) hyperplan séparateur. Chaque noeud possède un **fils gauche** et **un fils droit** qui . Il est construit à l'aide d'un algorithme recursif selon le principe suivant :

Function `Kdtree`$(T : Tableau, d : int, I_d : int, I_f : int)$ **:**

Data: Tableau T des N_s points de $B_s \subset X^p$ contenant initialement tous les éléments de B_s, d la profondeur du partitionnement initialement égale à 0, I_d Indice de départ initialement égal à 0, I_f Indice de fin initialement égal à $N_s - 1$

Result: Structure de donnée $K^d - Tree\ K - d$

axe $\leftarrow d\%k$

On ordonne la plage [I_d, I_f] du tableau T suivant la dimension axe qui vient d'être choisie

$median = I_f - I_d + 1//2$; `/* On calcule l'indice de l'élément médian */`

return *Noeud(*
$point = T[median]$,
$fils_{droit} =$ `Kdtree`$(T, d + 1, I_d, median)$,
$fils_{gauche} =$ `Kdtree`$(T, d + 1, median + 1, I_f)$
)

end

Construction d'un arbre K^d

L'instruction axe $\leftarrow d\%k$ permet de choisir une des p dimensions en fonction de la profondeur, afin de parcourir de manière cyclique toutes les dimensions.

Une fois l'abre construit, la recherche de plus proche voisin peut se faire de manière aisée avec une complexité de $O(\log(n))$ [Bentley 1975]. Pour trouver les k plus proches voisins d'un vecteur q,

- On commence par le noeud racine et on parcourt l'arbre de manière récursive comme pour faire une insertion
- Chaque noeud, on calcule la distance avec les fils gauche et droit afin d'identifier de quel coté de l'hyperplan séparateur se trouve q et on continue le parcours dans le noeud enfant associé.
- On itère jusqu'à atteindre une feuille. Et de là, on sélectionne les points de la feuille atteinte
- Ensuite on fait un parcours inverse de l'arbre, remontant noeud après noeud et l'on décide si l'on doit parcourir ou non les branches non visitées. Pour cela, à chaque

noeud visité, il faut regarder si la branche non visitée peut contenir des voisins plus proches. Si c'est le cas, on visite ces branches.

La méthode de partitionnement par un arbre K^d souffre néanmoins du problème de dimension (malédiction de la dimension ou curse of dimensionality) [R. 1957] : plus K est grand, moins cette technique est performante et a même tendance à atteindre un temps de recherche linéaire de $O(n)$[Sariel Har-Peled 1998] [T. 2003].

2.3.3 Locality-Sensitive Hashing

Locality Sensitive Hashing (LSH) est une méthode de recherche approximative dans des espaces de grande dimension. Elle répond entre autre au problème de "malédiction de la dimension" rencontré avec les méthodes de partionnement de l'espace [Piotr Indyk 1998]. Son principe est de réduire la dimension des données à traiter. Celles-ci sont "hashées" et regroupées en paquets de telle sorte que les entrées avec des hashs similaires se retrouvent dans le même paquet avec une probabilité élevée [Gionis 1999]. Il est important de noter que les fonctions de hash utilisées sont différentes de celles utilisées en cryptographie. Dans le cadre des LSH, les fonctions de hash tendent à maximiser la probabilité de colision entre éléments similaires.

Pour cette méthode, on définit $\mathcal{F}$ une *famille LSH* [Rajaraman 2011] [Gionis 1999] pour l'espace métrique $\mathcal{M} = (X^p, d)$ (d une distance), le seuil $\mathcal{R} > 0$ et le facteur d'approximation $c > 1$. $\mathcal{F}$ est une famille de fonctions $h : \mathcal{M} \to S$ qui fait correspondre un element de l'espace métrique à un paquet $s \in S$. Pour tous points $x, y \in \mathcal{M}$ et pour toute fonction $h \in \mathcal{F}$ choisie aléatoirement, les conditions suivantes sont vérifiées :

— si $d(x, y) \leq \mathcal{R}$, alors $h(x) = h(y)$ avec une probabilité au moins égale à P_1
— si $d(x, y) \geq c\mathcal{R}$, alors $h(x) = h(y)$ avec une probabilité au plus égale à P_2

$\mathcal{F}$ est intéressante si $P_1 > P_2$ c'est à dire les fonctions de hachage permettre aux points proches d'entrer fréquemment en collision tandis que les points éloignés doivent entrer rarement en collision. Une famille $\mathcal{F}$ vérifiant ces conditions est qualifiée de $(\mathcal{R}, c\mathcal{R}, P_1, P_2)$-sensitive.

Dans le cadre de la recherche de plus proche voisin, on définit une famille $\mathcal{G}$ de L fonctions de hachage g, où chaque g est la concatenation de t fonctions $h_1, ..., h_t$ de $\mathcal{F}$ i.e $g(p) = [h_1(p), ..., h_t(p)]$. Ainsi le haché d'une entrée x par g correspond à l'identifiant du paquet dans lequel est supposé se trouver x. Par la suite on construit L tables de hachage, correspondant chacune à une fonction de hachage g i.e la i^eme table de hachage contient les points de B_s hachés par la fonction g_j.

Lorsqu'il faut trouver les k plus proches voisins d'un vecteur q,

— hacher le point q avec chacune des L fonction de hachage
— pour chacune des L fonctions de hachage g, on trouve les points qui ont un haché similaire à celui de q. Ces points sont dans le même paquet que q pour la fonction

g correspondante.
— de ces points, retourner les k plus proches à l'aide d'un calcul effectif de distance entre le point q et chacun des points qui sont retournés à l'étape précédente

Exemple de famille LSH

Un exemple de famille LSH utilisée est la suivante :
— Choisir de manière aléatoire C plans dans l'espace. Ces plans divisent l'espace B_s en sous régions.
— Hasher chaque point de B_s en une séquence binaire (de O et 1) de taille C de telle manière que le i^eme bit de la séquence est 0 si le point est d'un côté du i^eme plan et 1 s'il est de l'autre.
— De manière intuitive, les points qui sont proches l'un de l'autre seront du même côté de beaucoup de ces plans. Comme résultat leurs représentations binaires auront plus de chance d'être similaires
— A partir des hashs obtenus un prétraitement est fait afin que la recherche de points à partir de leur hash soit efficiente.

Forêt LSH : une variante de LSH

Avec la méthode LSH, la complexité se situe dans la recherche d'éléments ayant un haché similaire à celui de q. Une approche consiste donc à utiliser une structure d'abre binaire [Mayank Bawa] [Garg] tel que :
— Chaque noeud correspond à l'une des fonctions de hachage g de $\mathcal{G}$
— On se déplace de l'un ou l'autre côté de l'arbre en fonction de la valeur du haché au noeud courant
— Lorqu'on cherche les points similaires à une entrée q les meilleurs candidats sont ceux qui se trouvent dans la même feuille que q

Néanmoins, une limite avec cette manière de parcourir l'arbre est que dépendamment de la fonction g qu'on choisit pour le noeud racine et les autres noeuds, une entrée x peut être proche de q mais se trouver dans une feuille de l'arbre éloigné de celle de q. Pour donner à l'algorithme une meilleur chance de trouver les plus proches voisins, on construit un ensemble d'arbre. Pour chacun de ces arbres, l'ordre dans lequel les fonctions de hachage sont séquencées est choisi de manière aléatoire. Il s'agit de l'algorithme de **Forêt LSH**

2.4 Distance entre textes

La recherche d'éléments similaires ou dupliquées est intrinsèquement liée à une notion de distance entre ces éléments. On ne peut définir la similarité entre deux vecteurs x et y que si une mesure de similarité est définie. Dans cette partie nous ferons un point sur les distances utilisables dans le cadre la recherche de plus proche voisin [**?**].

2.4.1 Distance euclidienne

Etant donné les vecteurs de l'ensemble B_s sur lequel la recherche de similarité sera faite, la distance euclidienne entre deux vecteurs x et y est définie tel qu'il suit :

$$d = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2} \tag{2.14}$$

2.4.2 Distance de Jaccard

Elle est introduite en 1901 par Paul Jaccard [Jaccard 1901]. Elle est utilisée pour comparer la similarité entre deux ensembles A_1 et A_2 . Pour la calculer, on définit l'indice de Jaccard qui est le rapport entre le cardinal de l'intersection des ensembles considérés et le cardinal de l'union des ensembles [Kosub]. Son expression est :

$$J(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|} \tag{2.15}$$

Pour le cas d'espèces des représentation vectorielles de textes x, $y \in B_s$, $|A_1 \cap A_2|$ est le nombre de colonnes i pour lesquelles x_i et y_i sont non nuls ; $|A_1 \cup A_2|$ est le nombre total de colonne de x (ou y puisqu'ils sont de la même dimension) A partir de l'indice de Jaccard, on peut ainsi calculer la distance de Jaccard tel qu'il suit :

$$d_J(A_1, A_2) = 1 - J(A_1, A_2) \tag{2.16}$$

2.4.3 Distance Cosinus

C'est une mesure de distance entre vecteurs de même dimension qui est basé sur le calcul du cosinus de l'angle entre ces vecteurs [Wael H. Gomaa 2013]. Pour l'obtenir, on définit la similarité cosinus qui est le cosinus de l'angle entre les vecteurs x et y ($x, y \in X^p$ dont on veut connaître la similarité et qui se calcule tel qu'il suit :

$$\text{sim}(x, y) = \frac{x \cdot y}{||x|| * ||y||} = \frac{\sum_{i=1}^{p} x_i * y_i}{\sqrt{\sum_{i=1}^{p}(x_i)^2}\sqrt{\sum_{i_1}^{p}(y_i)^2}} \tag{2.17}$$

A partir de cette définition de la similarité cosinus la distance cosinus se trouve tel qu'il suit :

$$d_{\text{sim}}(x, y) = 1 - \text{sim}(x, y) = 1 - \frac{\sum_{i=1}^{p} x_i * y_i}{\sqrt{\sum_{i=1}^{p}(x_i)^2}\sqrt{\sum_{i_1}^{p}(y_i)^2}} \tag{2.18}$$

2.5 Conclusion etat de l'art

Dans cette partie nous présenterons pour chacun des axes de recher de notre état de l'art, les solutions les plus adéquates à utiliser pour l'implémentation de la solution à proposer.

2.5.1 Extraction de caractéristique

De part sa définition et son procédé d'obtention, nous avons retenu la méthode Tf-Idf pour faire la transformation des données textuelles en vecteurs.

2.5.2 Problèmes de classification supervisé

Détection des spams

Le problème de détection de spams est un problème de classification binaire où l'on a que deux classes : **spam** et **non spam**. Il est question de choisir un modèle à utiliser afin de résoudre ce problème. La démarche suivie est celle préconnisée par le projet scikit-learn[scikit learn]. Celui-ci regroupe un ensemble de chercheurs et universitaires qui implémentent diverses techniques d'apprentissage artificiel dans le langage python. En fonction du contexte dans lequel on se trouve, ils recommendent la démarche suivante présentée à la Figure 2.1

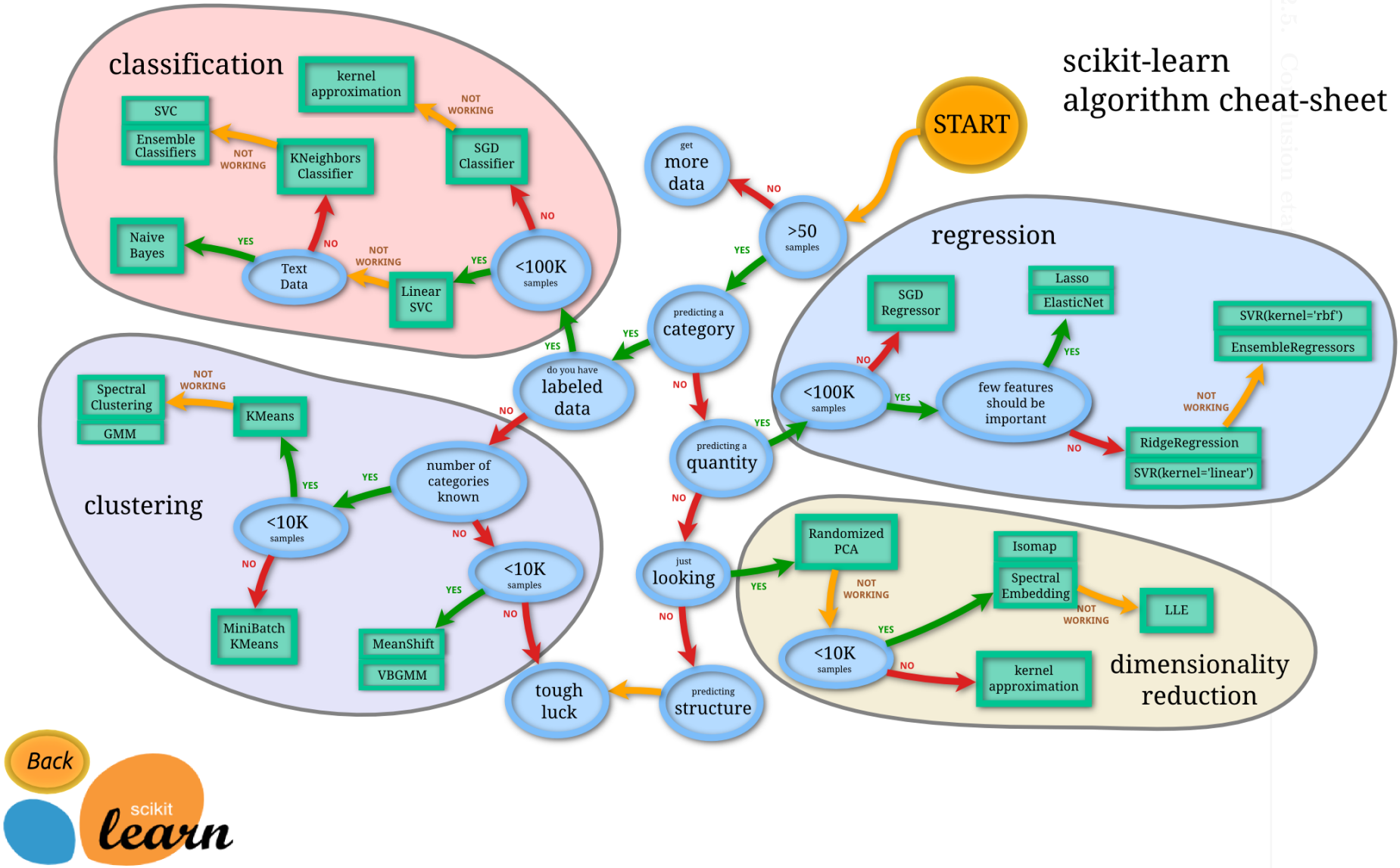

FIGURE 2.1 – Arbre de décision pour choix de modèle d'apprentissage (Projet scikit-learn)

Voici en exergue le parcours de l'arbre qui correspond à notre situation : faire de la catégorisation automatique à partir d'un jeu de données texte, étiqueté comportant moins de 100 000 éléments.

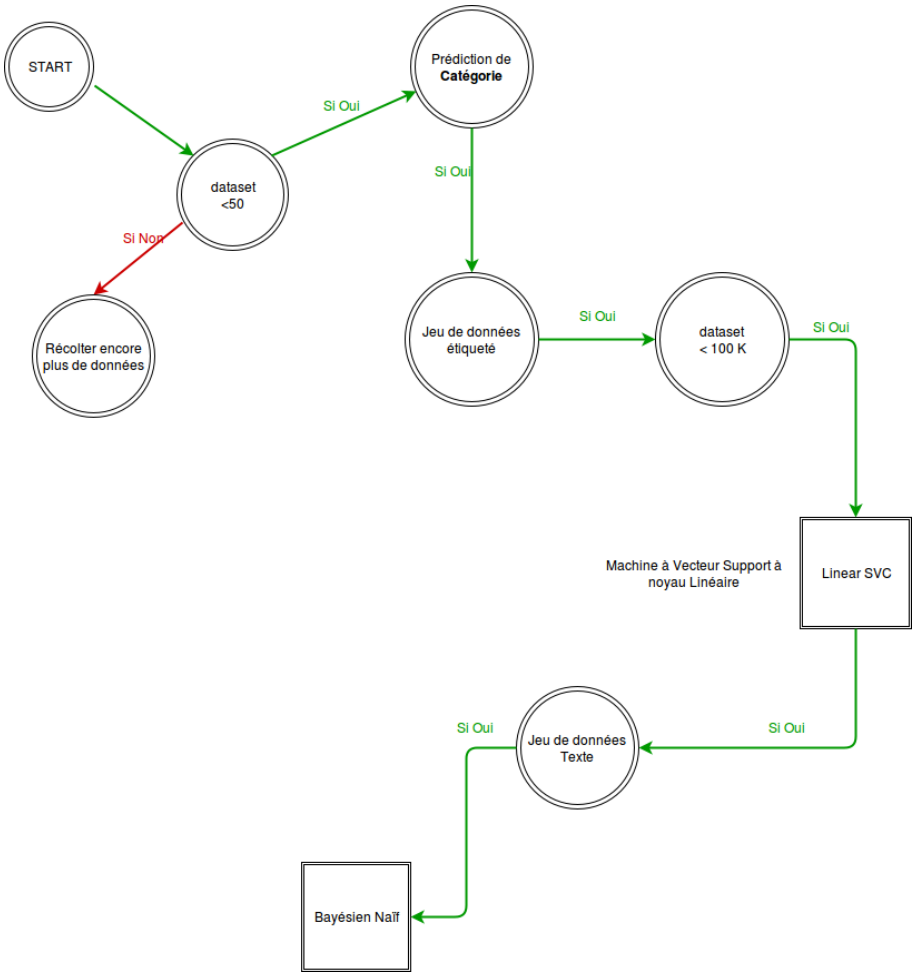

FIGURE 2.2 – Critères de choix du modèle de Bayes pour la détection de spams

Classification des annonces dans les catégories

En ce qui concerne la classification, la première approche serait d'adopter le modèle Bayésien Naïf suivant la recommandation du projet scikit-learn [scikit learn], mais elle s'avère avoir des limites. Vu le grand nombre de catégories existantes, celles qui n'ont pas suffisament d'exemples se voient sous-représentées dans la base d'entraînement. Ceci aura pour conséquence que ces classes seront rarements choisies par le modèle lors des prédictions, même pour les entrées de ces classes. Cela s'explique par le mode de fonctionnement du classifieur de Bayes, qui comme souligné dans la sous-section 2.2.1,

plus une classe est représentée dans la base d'apprentissage, plus il y'a des chances qu'elle soit la classe prédite.

Ce constat étant fait, nous avons décidé de nous appuyer sur la détection de similarité afin de faire une classification de type k-means. En effet, étant donné une entrée à prédire x_{new}, nous extrayons de notre base d'apprentissage l'ensemble V des K éléments qui lui sont les plus proches. Ensuite nous procédons à un système de vote, où chacun des éléments de V donnera un vote inversément proportionnel à la distance qui le sépare de l'entrée à prédire [J. 1977]. De manière arbitraire, nous avons fixé K à 7. La classe finale est alors :

$$c = argmax_{c_k \in C} \sum_{j \in V_k} \frac{1 - d_j}{\sum_{i \in V}(1 - d_i)} \tag{2.19}$$

avec où V_k est l'ensemble des éléments de V qui sont de la classe c_k et d_j la distance entre x_{new} et l'élément j de V_k. La distance d_j variant entre 0 et 1. Avec cette méthode, l'entraînement consistera en fait à l'indexation de la base d'apprentissage en vue de la recherche efficace d'éléments similaires.

2.5.3 Recherche de plus proche voisin

Les données à manipuler d'un important volume des centaines de milliers d'entrées et sont de types textes, c'est à dire une fois transformées donnent des vecteurs de très grande dimension (de l'ordre du millier), il est important de choisir une méthode de recherche de plus proche voisin qui correspond. La méthode **LSH** plus particulièrement **Forêt LSH** a été choisie, car elle correspond le mieux au problème que nous tentons de résoudre.

2.5.4 Calcul de distance

Le choix de la distance à utiliser, nous l'avons fait de manière expérimentale. Le principe a été de calculer par plusieurs méthodes, les distances entre un ensemble de textes afin de sélectionner la plus convenable. Voici un extrait de notre expérimation.

Pour l'ensemble de textes suivant (obtenus en contatenant les titres et descriptions d'annonces de leportail.ci) :

Pour ces textes, il faudrait une distance qui une fois calculée, donne comme résultat :

- Le texte 1 plus proche des textes 2 et 3 que des autres textes
- Le texte 1 plus proche des textes 4, 5 et 6 que du texte 7

Nous avons fait un calcul des distances entre ces textes par chacune des méthodes recensées dans notre état de l'art (section 2.4). Voici la distance par chacune des méthodes entre le premier texte et les cinq autres :

Les remarques suivantes peuvent être faites :

1	Chrysler crossfire vends Chrysler crossfire, de couleur noire, immatriculée FP, intérieur cuir, ordinateur de bord, jante alu, clim
2	Chrysler Chrysler crossfire en vente, année 2007, 50.000km, essence, couleur noire, immatriculation FP, cuir, ordinateur de bord, jante alu, climatisation, boite auto Prix de vente :1250000
3	Chrysler c 300 chrysler c 300,jante allu,boite auto,interieur cuir,faible consommation,bonne tenue de route,ecran a bord,full option,encore neuf,importé,bon état. – – – –
4	MERCEDES classe S500 MERCEDES classe S500 de couleur noire est en location, année 2009, cinq portes, 70.000 kilométrage, essence, auto, abs, phares xénon, jantes alu, climatisation, GPS ; cuir, ordinateur de bord, MP3
5	Toyota rav4 Particulier vend une Toyota rav4 de couleur grise intérieur cuir, boite automatique, année 2008, toit ouvrant, 75.000 kilométrage, essence, ordinateur de bord Prix de vente : 13.500.000 Fcfa
6	BMW X5 modèle 2002 Voiture 4x4 de couleur grise, intérieur cuir
7	TERRAIN DE 750m^2 Situé dans la commune de Yopougon résidentiel, un terrain dans un bon emplacement avec certificat de propriété. + 225 – – – –/ + 225 – – – –

TABLE 2.1 – Textes utilisés pour le test de distances (Nous avons masqués ici les numéros de téléphones présent dans les textes 3 et 7)

Distance	**Texte 2**	**Texte 3**	**Texte 4**	**Texte 5**	**Texte 6**	**Texte 7**
Euclidienne	7.071	9.055	9.486	9.486	7.071	10.392
Jaccard	0.113	0.1691	0.159	0.167	0.099	0.195
Cosinus	0.481	0.791	0.850	0.853	0.856	0.962

TABLE 2.2 – Résultats de test des différents types de distance entre texte

— Toutes les distances rendent compte du fait que le texte 1 est plus proche des textes 2 et 3 que des textes 4, 5 et 7
— Cependant un comportement indésirable est observé pour les distances de Jaccard et euclidienne du Texte 1 au texte 6. Ces deux distances donnent comme résultat que le texte 1 est plus proche du texte 6 que du texte 2. Or ceci ne reflète pas la réalité des faits.

Ceci est une illustration des limites que peuvent avoir les distances euclidienne et de Jaccard par rapport à la distance du cosinus.

Chapitre 3

Méthodologie

Dans ce chapitre, nous présentons les méthodologies utilisées pour mettre sur pied les solutions aux problèmes posés. Dans un premier temps, nous introduisons de manière formelle les étapes suivies pour la résolution du problème de classification supervisée de texte. Ensuite nous mettrons en lumière le procédé que nous proposons afin de parvenir à détection d'annonces dupliquées à leur création et ce sur une base de recherche qui est constamment mise à jour. Enfin il sera question d'expliciter la modélisation adoptée pour notre plateforme afin d'inclure.

3.1 Etapes de la classification supervisée

Dans notre contexte, la classification supervisée part d'un ensemble de textes auxquels des labels ont été donnés, labels correspondant à la classe à laquelle chacun de ces textes appartient respectivement. A partir de ces textes on utilise une méthode d'apprentissage artificiel pour la classification afin de produire un modèle qui nous permettra de faire des prédictions pour les textes nouveaux dont on aimerait connaître la classe. De ce fait, nous avons défini une démarche formelle qui comporte trois principales étapes :

- Etiquetage des données : ici les données d'apprentissage sont chargés et étiquetées selon les labels des classes à prédire.
- Choix et entraînement du modèle : cette étape prend en entrée les représentations vectorielles obtenues à la phase précédente et donne en sortie un classifieur. Pendant cette phase également le classifieur est évalué à partir des données fournies afin d'en déterminer la justesse.
- Prédiction : à ce niveau il s'agit de prédire les classes de nouvelles entrées à partir du modèle entraîné

3.1.1 Etiquetage des données

Le principe de l'apprentissage supervisé c'est de renseigner au modèle à entraîner des données représentatives du problème à résoudre. Dans le cadre de la classification supervisé on doit fournir au modèle des données de chacune des classes à prédire, d'où l'importance de cette étape. Elle consiste donner une étiquette à des données chargées pour que le modèle puisse savoir à quelle classe elles appartiennent.

3.1.2 Choix et entraînement du modèle

Pour chacun des problèmes de classification (Détection de spams et Catégorisation automatique), les données obtenues à l'étape précédente doivent être injectées dans le modèle pour entraînement. A l'issue de cet entraînement, on obtient un classifieur capable de prédire pour une nouvelle entrée si elle est spam ou non spam d'une part, ou à quelle catégorie elle appartient d'autre part.

3.1.3 Evaluation du classifieur

Score f1

Évaluer notre classifieur consiste à mesurer sa justesse. L'approche de base est de sélectionner de manière aléatoire dans la base d'apprentissage un ensemble d'éléments qui vont servir de données de test. L'idée est d'entraîner la base d'apprentissage privée des données de test de mesurer le taux d'erreur comise par le modèle sur cet ensemble de test. De ce fait il est important pour nous de définir une mesure de l'erreur qui sera représentative de comment notre modèle se comporte.

Dans le cas d'une classification binaire, on choisi un label qui sera dit positif (l'autre label est alors qualifié de négatif). Et lorsqu'on fait une prédiction, quatre cas de figure sont envisageables :

- L'exemple est réellement positif et le modèle le prédit comme positif
- L'exemple est négatif et le modèle le prédit négatif
- L'exemple est positif et le modèle le prédit comme négatifs. Ces cas sont considérés, comme **faux négatifs**
- L'exemple est négatif et le modèle le prédit comme positif. Ces cas sont considérés comme **faux positifs**

Les deux derniers cas sont ceux où le modèle commet des erreurs. Par la suite nous définissons les metriques suivantes :

- **Le rappel** qui représente la proportion d'annonces détectés comme spam parmi l'ensemble des spams existants

— **La précision** qui donne la proportion de vrais spams parmi les annonces détectées comme spams par le modèle

À partir de celles ci nous pourrons calculer un score qui permet d'évaluer notre solution en tenant compte des faux positifs et des faux négatifs. Le score retenu est le **score f1**, généralement utilisé lors de la résolution de ce type de problème. Il se calcule tel qu'il suit :

$$f_1 = 2\frac{\text{précision} * \text{rappel}}{\text{précision} + \text{rappel}} \tag{3.1}$$

Pour une classification avec plus de deux classes :

1. Pour chacune des classes
   — on la choisit comme label positif. Tous les autres labels sont considérés comme négatif
   — on calcule le score f1 sur la base de la classe dite positif
2. Le score f1 final est calculé en faisant la moyenne des score f1 des labels. Cette valeur est le **macro-score**

Cette manière de faire peut également être appliquée à une classification binaire, chacune des classes sera à tour de rôle choisie comme positive.

Validation Croisée

Afin d'inclure toutes les entrées dans le processus de mesure du score. Nous faisons de la validation croisée cyclique (k-fold validation) comme décrite dans cet article [Refaeilzadeh 2009] mais modifié :

1. l'ensemble des données d'apprentissage est divisé en 6 parties égales
2. pour chacun des 6 ensembles
   (a) On le considère comme l'ensemble de test et les 5 autres comme l'ensemble d'apprentissage
   (b) Le modèle est entraîné sur l'ensemble d'apprentissage et son score f1 de cette étape est calculé
   (c) la précision et le rappel sont également calculé à cette étape
3. A la fin nous choisissons la répartition **données de test - données d'apprentissage** qui a obtenu le meilleur score. Le modèle entraîné avec ces données d'apprentissage sera notre classifieur final.
4. Néanmoins, le score f1 final est la moyenne des 6 scores f1 est calculé. De même pour la précision et le rappel

3.1.4 Prédiction

La prédiction consiste à utiliser le modèle entraîné afin d'affecter à toute nouvelle entrée, la classe qui lui correspond le mieux.

3.2 Mise en place d'un procédé de recherche de plus proche voisin sur une base de recherche dynamique

3.2.1 Constat

Pour effectuer une recherche rapide d'éléments dupliquées et similaires à une entrée, le principe est d'indexer l'ensemble B_s des N_s éléments sur laquelle la recherche de plus proche voisin doit se faire afin de rendre celle-ci efficace. Si la recherche est rapide, l'indexation cependant s'avère être couteuse en temps, de l'ordre de $O(N_s)$ or notre système doit être en mesure de se rendre compte qu'une annonce est dupliquée avec une autre nouvellement créée. Il faudrait donc qu'à chaque fois qu'une annonce est crée sur le site, celle-ci soit directement ajoutée à l'ensemble sur lequel la recherche est faite. Cependant on ne saurait faire une indexation à chaque fois qu'une annonce est créée car gourmand en ressource. La solution que nous proposons est l'utilisation de deux ensembles de recherche B_s et B_ts sur chacune desquelles sera appliquée une méthode spécifique de recherche.

3.2.2 Idée de base

Pour palier au problème que l'on rencontre avec la méthode d'indexation par **Forêt LSH** qui a du mal à fonctionner en incluant de nouvelles entrées, nous optons pour un système de recherche avec deux bases. La première base de recherche B_s sur laquelle l'indexation par **Forêt LSH** se fera à intervalle de temps régulier et relativement grand (du fait du temps de l'indexation). La deuxième base de recherche B_ts, plus petite que la première qu'on appellera base tampon où seront ajoutées instantanément les annonces nouvellement créées et sur laquelle la recherche se fera par la méthode dite "naïve". Pour chaque nouvelle entrée, la recherche de plus proche voisin se fera à la fois sur la base tampon B_ts et sur la base B_s, la réunion des deux résultats va constituer le résultat final.

3.2.3 Fonctionnement

Les deux ensembles de recherche sont construits à partir de la source de données qui contient l'ensemble des textes. Lorsqu'une recherche est effectuée, elle est faite à la fois dans la base de donnée tampon avec la méthode de recherche linéaire dite naïve et dans une base de données indexée par la méthode de forêt LSH. Une fois ces deux recherches effectuées, les résultats sont combinés pour obtenir le résultat final Figure 3.1.

De manière un peu plus spécifique, la Figure 3.2 présente comment est définie la recherche dans la base tampon, afin d'inclure de nouvelles entrées :

Avec cette manière de procéder, la base de recherche n'a plus besoin d'être initialisée à

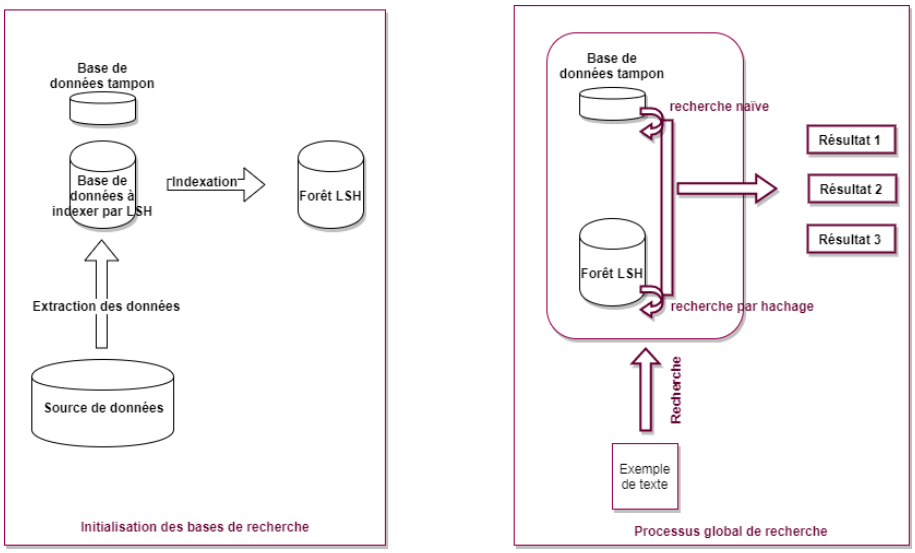

FIGURE 3.1 – Fonctionnement général du processus de recherche d'éléments similaires sur une base de données dynamique

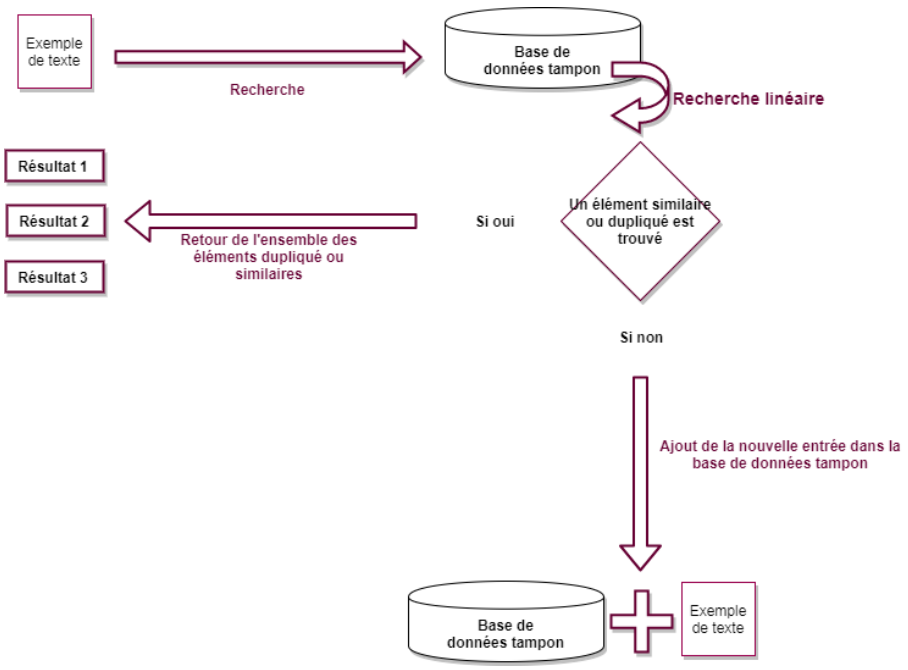

FIGURE 3.2 – Processus de recherche dans la base tampon

chaque fois qu'il y'a une nouvelle entrée. La taille de la base tampon étant fixe T, il faut fixer une période P après laquelle l'instance de forêt LSH sera rafraîchie en fonction de la

fréquence f avec laquelle les entrées sont ajoutées, on trouve :

$$P\frac{T}{f} \tag{3.2}$$

Car les annonces étant publiées à une fréquence f il faut que la période de rafraîchissement soit suffisament petite pour que la base tampon ne se remplisse pas. Or il faut $t = \frac{T}{f}$ secondes pour que la base tampon se remplissent. P doit donc être inférieur à t d'où la formule.

3.3 Modélisation du système

3.3.1 Concepts clés

Afin de mettre sur pied une plateforme intégrant les techniques présentées jusqu'ici, il nous a fallu d'abord définir un ensemble de concepts qui permettront une bonne modélisation du système et une bonne utilisation de celui-ci. En effet les étapes pour la détection de spams et pour la classification automatique étant définies, on doit pouvoir à partir de la plateforme les appliquer à la fois sur les sites kerawa et leportail. De même, l'implémentation de notre solution de recherche d'éléments dupliqués ou similaire ne doit pas être spécifique à un site mais applicaple à la fois pour kerawa et leportail. Chacune des plateformes étant indépendante l'une de l'autre et basées sur des technologies différentes.

Projet

Un projet, est l'entité principale de notre système. Toutes les opérations que ce soit la détection de spams, d'annonces similaires ou la catégorisation automatique se fait à l'intérieur d'un projet. C'est le contenant de tous les éléments nécessaires aux traitements à effectuer. Ainsi pour faire de la détection de spams ou de la catégorisation automatique, l'administrateur de la plateforme va :

- tout d'abord créer un projet
- ensuite télécharger les données nécessaires pour l'entraînement
- initialiser un entraînement à l'issue duquel sera produit un classifieur que nous associons au projet
- enfin pouvoir faire des prédictions à partir du classifieur obtenu

En ce qui concerne la recherche d'éléments dupliquées, l'administrateur :

- crée un projet
- lance l'initialisation de la base de recherche B_s depuis une base de données
- lance l'indexation de la base de recherche
- fait une recherche d'éléments dupliquées ou similaires pour une nouvelle entrée

Instance de base de donnée

Pour un projet de détection de similarité et d'annonces dupliquées, il nous faut nous connecter à une source de données qui constituera notre espace de recherhce B_s. Et puisque il nous est impossible de faire les opérations d'indexation depuis les bases de données de l'entreprise, nous définissons une entité dont le rôle sera d'extraire les données depuis une base de recherche. Une fois cette extraction faite, l'instance de base de données permettra aussi de faire des requêtes sur l'image de la base de données ainsi obtenue : récupérer l'élément qui se trouve à une position précise, récupérer la valeur

d'un attribut d'un élément particulier et bien d'autres.

Instance de vectorisation

C'est l'entité utilisée pour effectuer la vectorisation des textes. Elle obtenue lors de la phase d'initialisation de tout projet à partir de l'ensemble des mots présents dans les données texte du projet. A partir de celle-ci, toute nouvelle entrée sera transformée en vecteur.

Instance de prédiction

Il s'agit de l'entité utilisé pour effectuer les prédictions d'une part et la recherche d'autre part. Dans le cadre de la détection de spams, il s'agit du classifieur obtenu après la phase d'entraînement. Pour la recherche d'éléments similaires et dupliquées, c'est la structure de données obtenu après indexation des textes d'entrée.

API - Application Programming Interface

Une API (Application Programming Interface) est la définition d'un ensemble d'interfaces à partir desquelles une appplication communique avec d'autres qui utilisent ses services.

Les sites sur lesquels la plateforme devra être utilisée sont susceptibles d'être différentes et indépendante. Une solution qui devra s'intégrer dans l'un ou l'autre site n'est donc pas envisageable. De ce fait, les services de détection de spam, de classification et de recherche de similarité ou de dupliqué devront être des services distants qui sont appelés avec des données en entrée pour des résultats en sortie. D'où la notion d'API.

3.3.2 Analyse

Notre système interagit avec deux principaux acteurs : l'administrateur et le site ou plateforme extérieurs utilisant les services de MLA. L'administrateur devra pouvoir effectuer les opérations suivantes (Figure 3.3) :

- Créer un projet, c'est le préable pour toute autre opération. Le projet encapsule tous les entités qui entrent en jeu dans la résolution d'un problème spécifique par la plateforme
- Télécharger les données d'entraînement depuis des fichiers
- Etiqueter les données chargées en leur donnant des labels qui correspondent à la classe à laquelle ils appartiennent
- Lancer l'entraînement du modèle à partir des données étiquetées
- lancer le chargement des données depuis une base de données pour la recherche de similarité
- Initialiser l'extraction et l'indexation des données chargées

Le site ou plateforme d'extérieure devra pouvoir effectuer un ensemble d'opérations (Figure 3.4) :

- Prédire la classe d'une entrée qui sera le label spam ou non spam dans le cas de la détection de spams et la catégorie de l'entrée dans le cas de la catégorisation automatique

3.3.3 Architecture générale de la plateforme

Dans cette partie, nous présenterons comment la plateforme mise sur pied s'intègre dans l'infrastructure de l'entreprise. Ensuite, nous présenterons la plateforme du point de vue interne en explicitant les composants qui entrent en jeux dans son fonctionnement.

A. Intégration de la plateforme dans une infrastructure d'entreprise

Un site web comporte généralement trois principales parties : un client, un serveur d'application et un serveur de base de données.

Client Le client est la partie du site qui interagit avec les utilisateurs. Elle correspond à ce qui est affiché dans un navigateur web ou dans l'application mobile.

Serveur d'applications Il regroupe un ensemble d'applications contenant la logique métier du site. Ces applications interagissent entre elles afin d'offrir des services aux utilisateurs finaux.

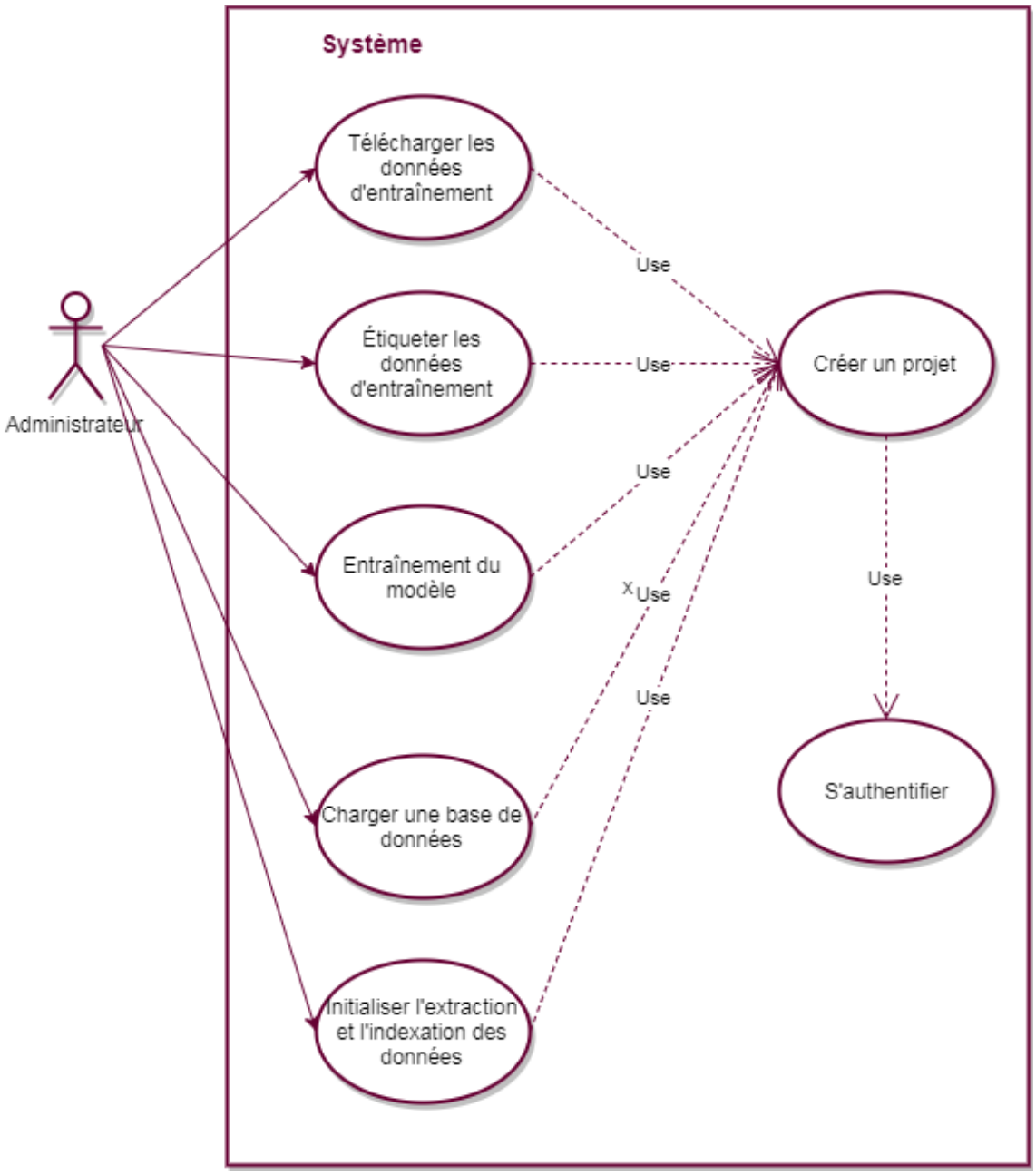

FIGURE 3.3 – Diagramme cas d'utilisation de l'acteur administrateur

Serveur de base de donnée C'est l'instance de stockage de toutes les données qui sont manipulées sur le site.

Intégration de la plateforme MLA La plateforme MLA s'intègre interagit avec deux principales composantes du site (Figure 3.5) : le serveur d'application et la base de

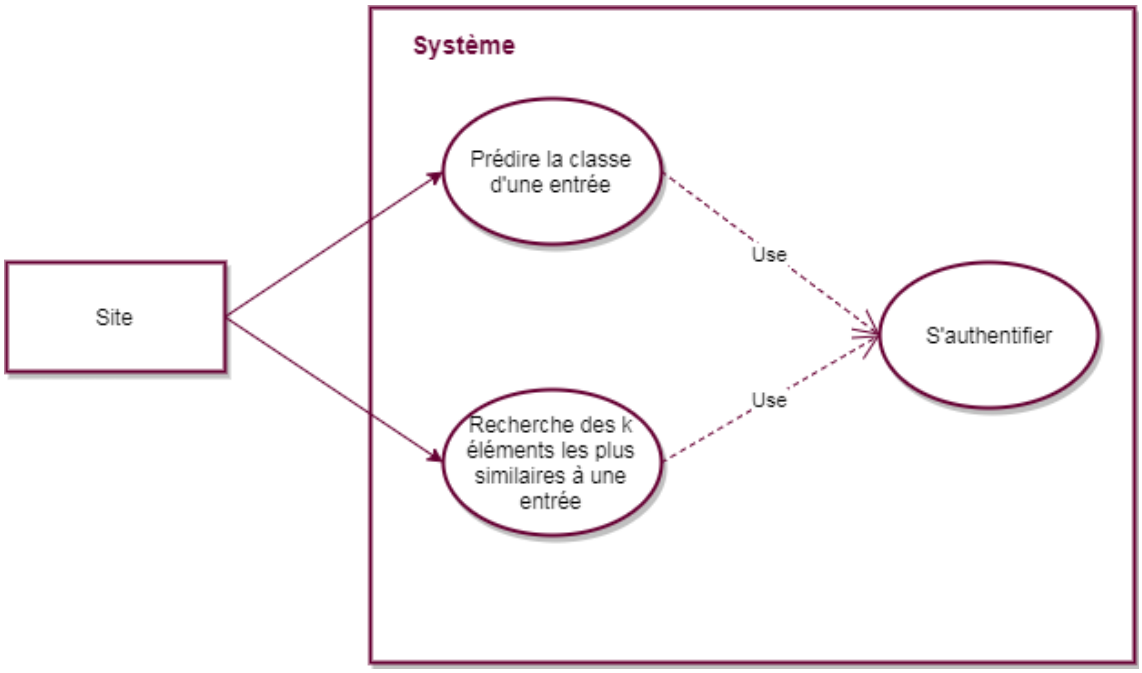

FIGURE 3.4 – Diagramme cas d'utilisation de l'acteur "site"

données.

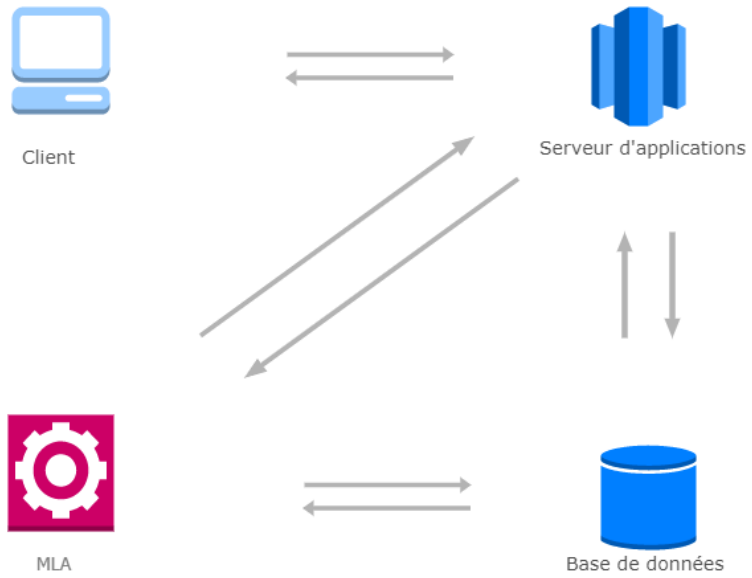

FIGURE 3.5 – Intégration de la plateforme MLA

— D'une part, les nouvelles annonces sont créées depuis le client et transite par le serveur d'application qui fait des traitements. Ainsi, la nouvelle annonce créées est envoyé à MLA qui s'occupe de faire les traitements qui sont attendus de lui et

renvoie le résultat
— D'autre part, la plateforme MLA interagit avec la base de données pour soit y extraire des données, soit pour faire de la prédiction sur des données déjà présentent en base de données via un cron qui parcours ladite base envoie chaque élément afin que la plateforme lui retourne un résultat

B. Architecture interne de la plateforme MLA

Dans sa composition interne, la plateforme est structurée en plusieurs modules (Figure 3.6) :

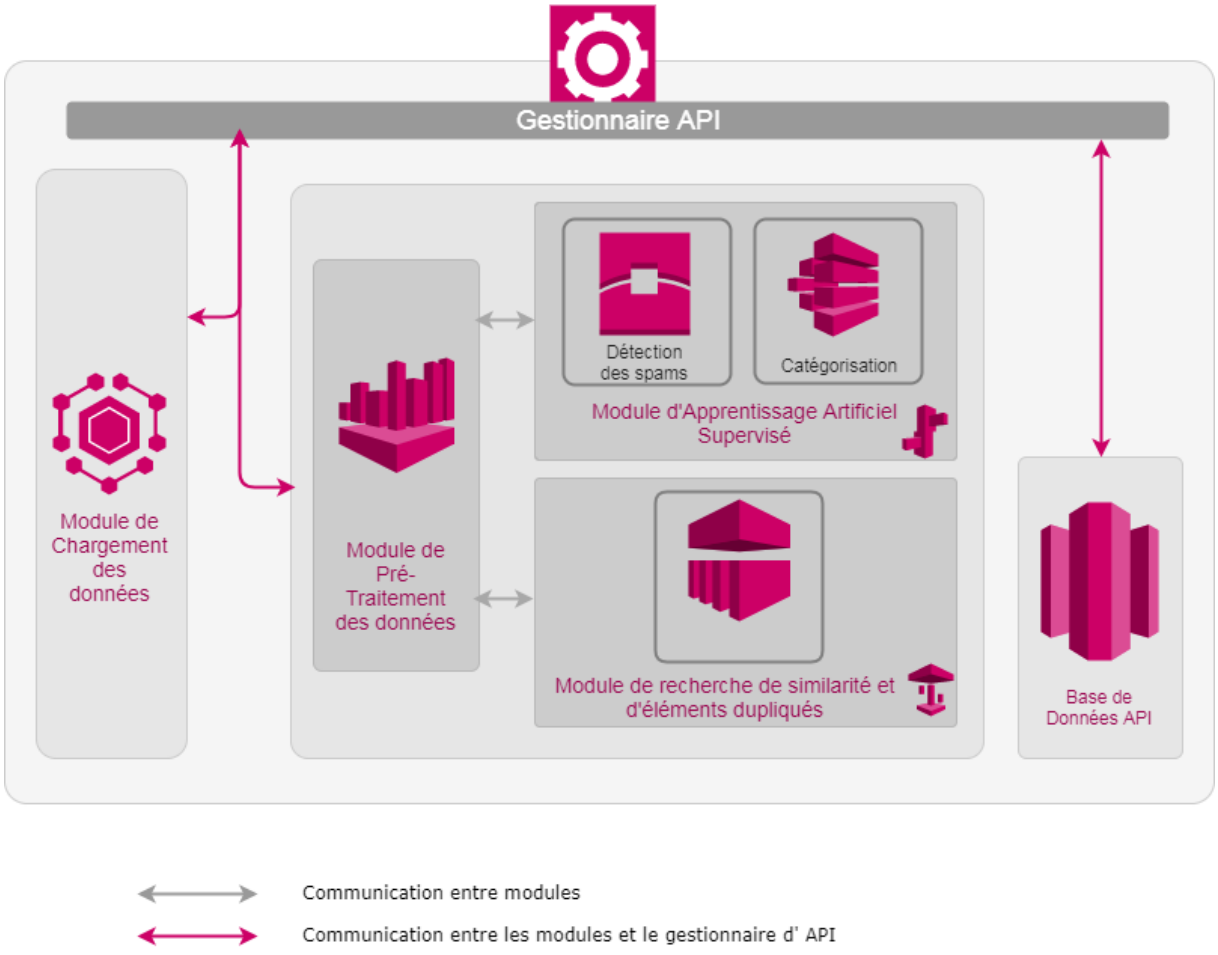

FIGURE 3.6 – Structure interne de la plateforme MLA

— **Le gestionnaire d'API** c'est à ce niveau que sont orchestrées toutes les opérations entre les autres modules. Il est l'entité d'interaction directe avec l'utilisateur. C'est à ce niveau que sont créés et configurés les projets, les instances de bases de donnée, instances de vectorisation et les instances de prédiction.

— **Chargement des données** (Figure 3.7) : dans ce module sont réunies toutes les opérations d'extraction des données depuis des bases de données ou des fichiers. Ces données sont stockées sous forme de fichier et manipuler grâce au gestionnaire de fichier implémenté dans ce module.

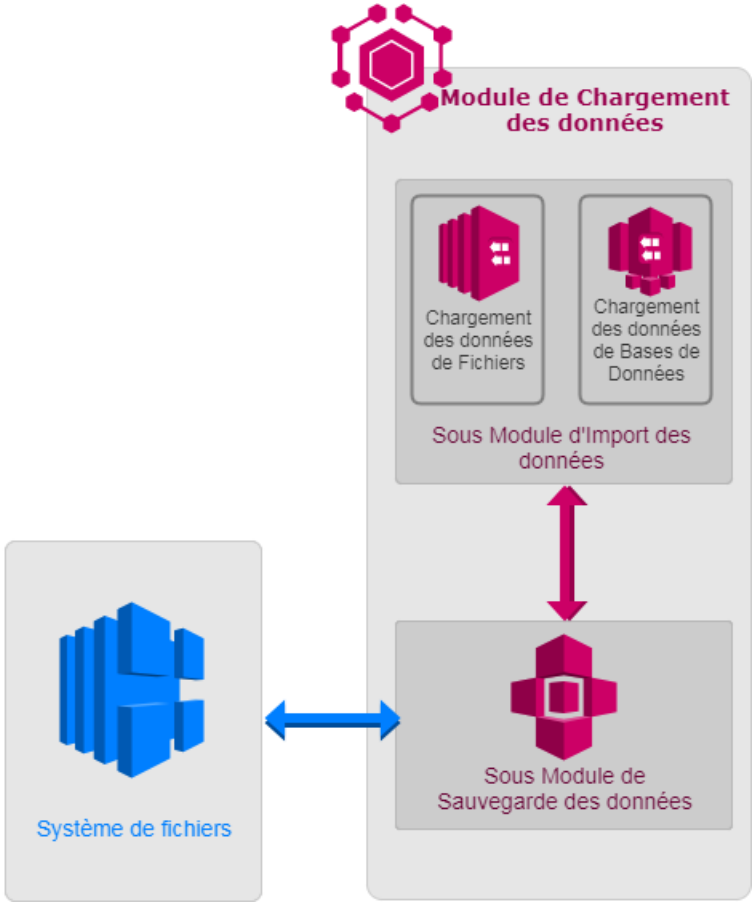

FIGURE 3.7 – Module de chargement de données

— **Module de pré-traitement des données** : Ce module sert principalement à la vectorisation des données textes issues du chargement des données.
— **Module d'apprentissage artificiel supervisé** (Figure 3.8) : utile dans la détection de spam et la catégorisation automatique, il est basé sur les étapes de la classification supervisée définie (section 3.1)
— **Module de recherche de similarité et recherche d'éléments dupliqués** : il comporte essentiellement l'indexation des données qui produit une structure de données où la recherche se fait de manière rapide.
— **Base de données de l'API** unité de stockage de toutes les informations nécessaire au fonctionnement de la plateforme.

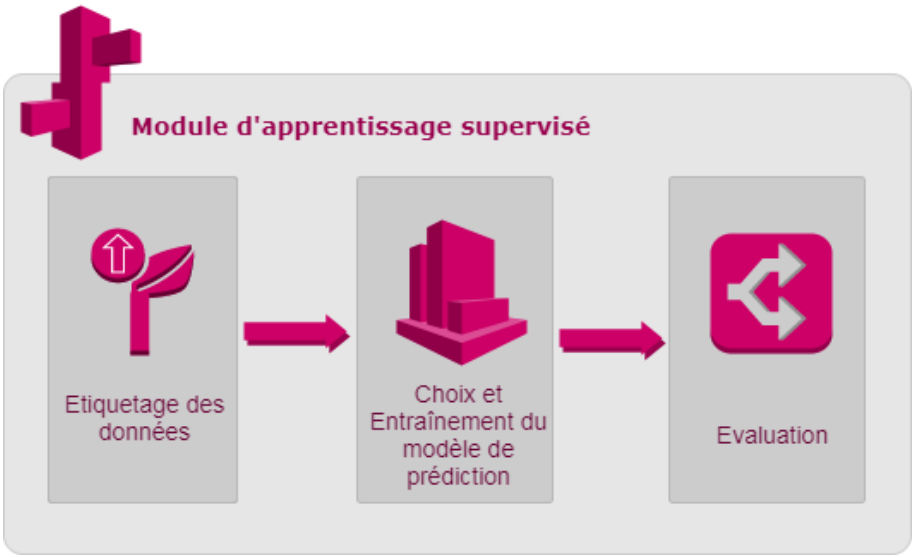

FIGURE 3.8 – Module d'apprentissage artificiel supervisé

3.3.4 Conception

Dans cette partie nous présentons comment chacune des opérations effectuée par les acteurs de la plateforme se déroule de manière séquentielle.

Authentification (Figure 3.9) : Le processus est le même pour tous les acteurs.

Créer un projet (Figure 3.10) : L'administrateur crée un projet en lui donnant un nom et une description où il spécifie le contexte et l'objectif du projet créé.

Télécharger les données d'apprentissage (Figure 3.11) Une fois le projet créé, les données d'apprentissage doivent être chargées et associées au projet. Cette tâche est spécifique aux problèmes de classification supervisé.

Etiqueter les données d'apprentissage (Figure 3.12) Les données chargées pour l'apprentissage doivent être étiquetées : l'administrateur spécifie à quelles classes appartiennent les données chargées. Ces données étiquetées sont stockées dans le système de fichier.

Entraînement du modèle (Figure 3.13) La phase d'entraînement est initialisée par l'administrateur et un classifieur est généré et stocké dans le système de fichier.

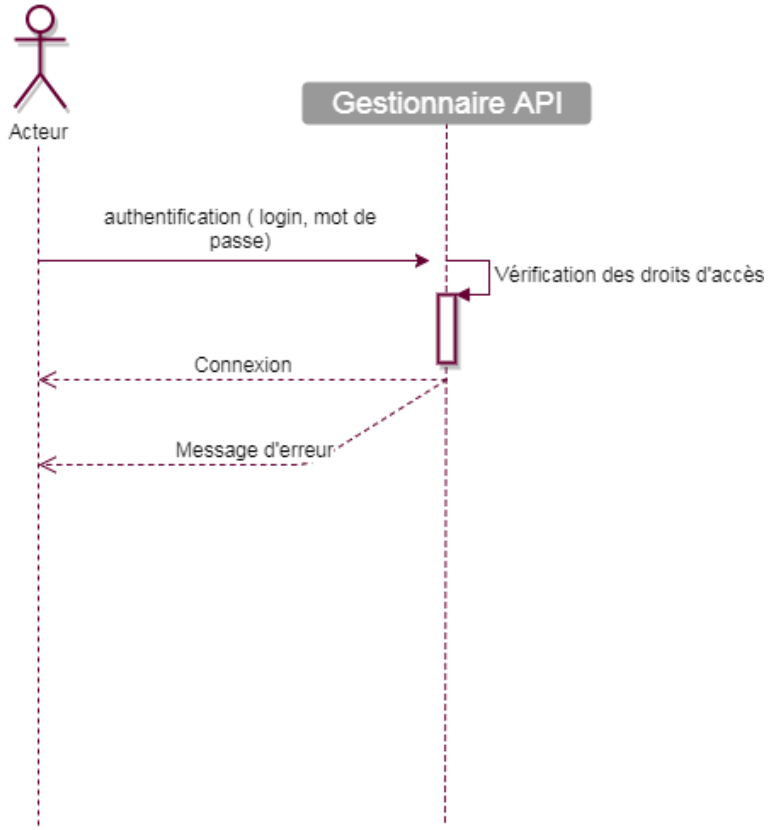

FIGURE 3.9 – Diagramme de séquence du cas d'utilisation S'authentifier

Chargement d'une base de données (Figure 3.14) La recherche de similiraté se fait sur les données d'une base de données. De ce fait la première étape de ce processus sera d'extraire ces données depuis la base pour une indexation ultérieure. L'indexation ne pouvant se faire directement sur la base de données.

Indexation des données (Figure 3.15) L'administrateur initialise la construction de la structure de données Forêt LSH qui est sauvegardé sous forme de fichier et sera utilisée plutard lorsqu'il faudra faire une recherche d'éléments similaires

Prédire la classe d'une entrée (Figure 3.16) Les séquences présentées jusque là sont celles qui précèdent la prédiction. Celle-ci consiste à donner pour une nouvelle entrée le label (la classe) qui lui correspond.

Recherche des k éléments les plus similaires à une entrée (Figure 3.17) Le site fournit à la plateforme MLA une entrée et la plateforme lui retourne une liste d'éléments qui lui sont similaires ou identiques.

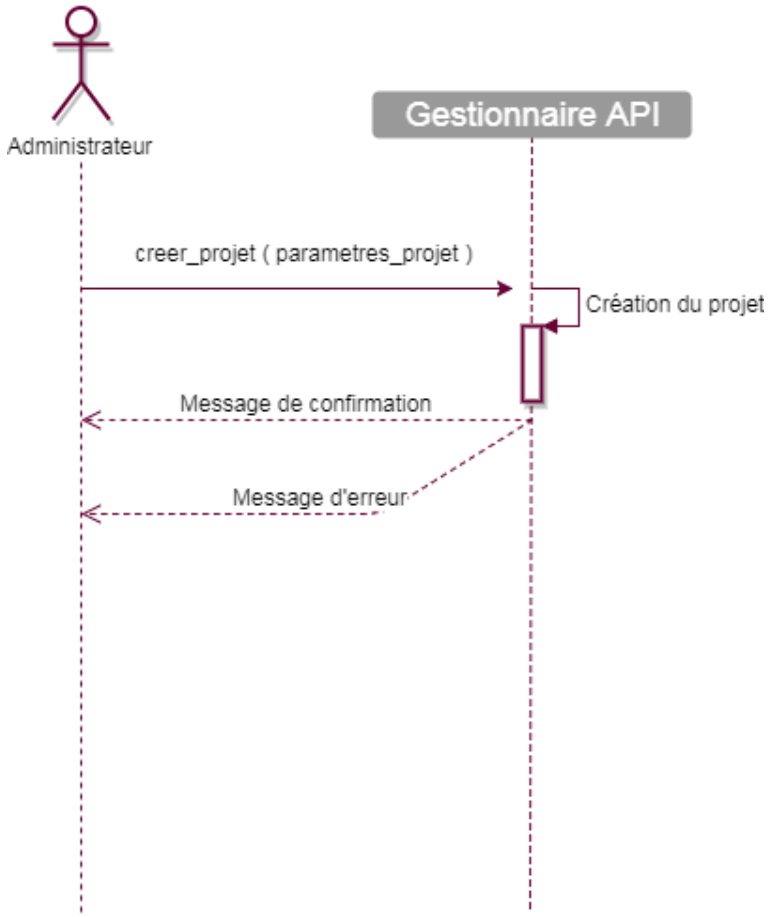

FIGURE 3.10 – Diagramme de séquence du cas d'utilisation Créer un projet

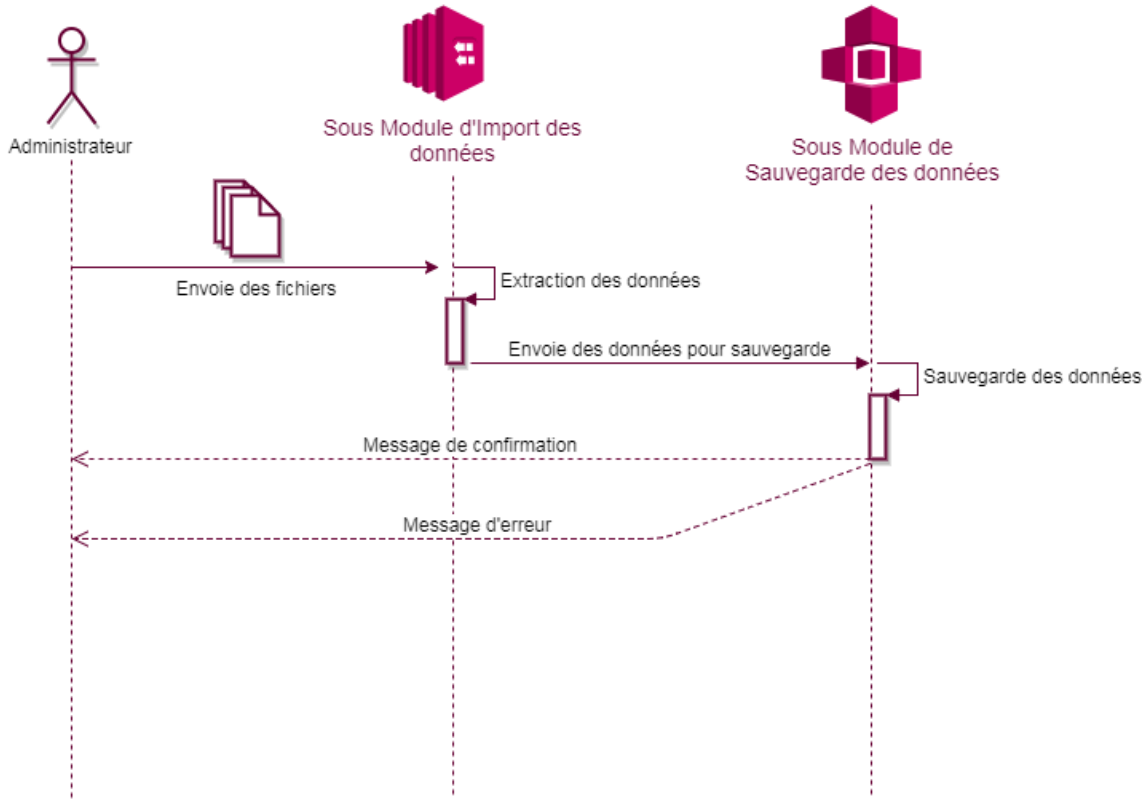

FIGURE 3.11 – Diagramme de séquence du cas d'utilisation Télécharger les données d'apprentissage

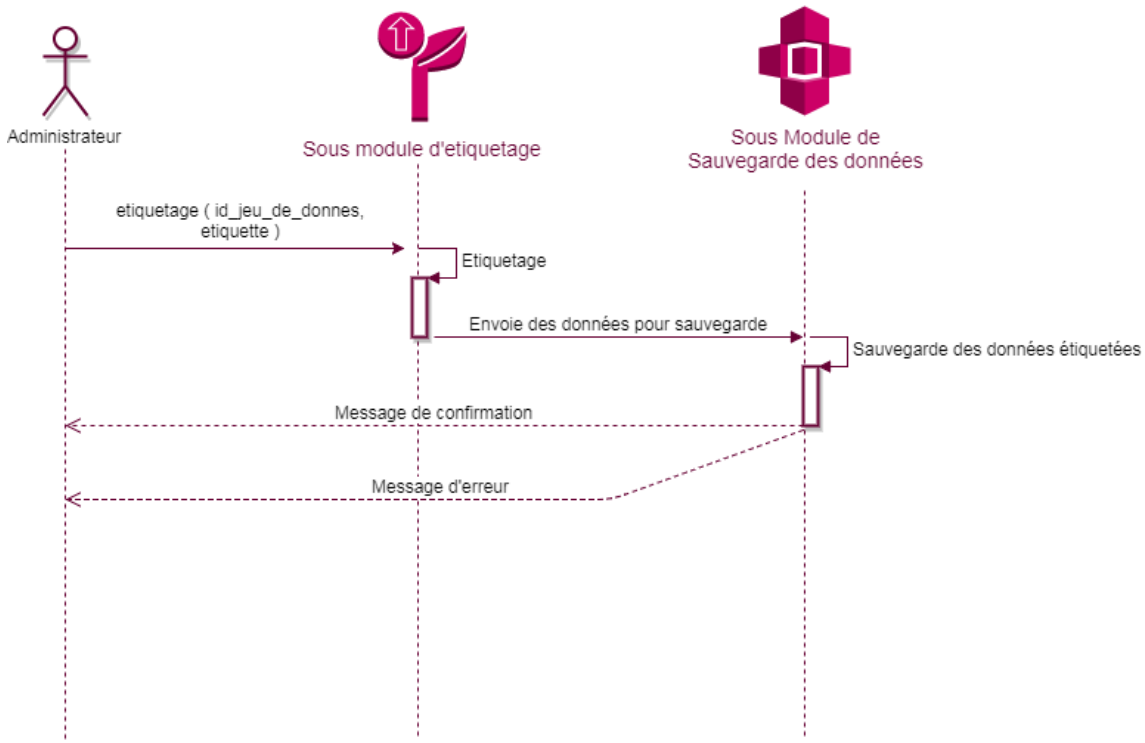

FIGURE 3.12 – Diagramme de séquence du cas d'utilisation Etiqueter les données d'apprentissage

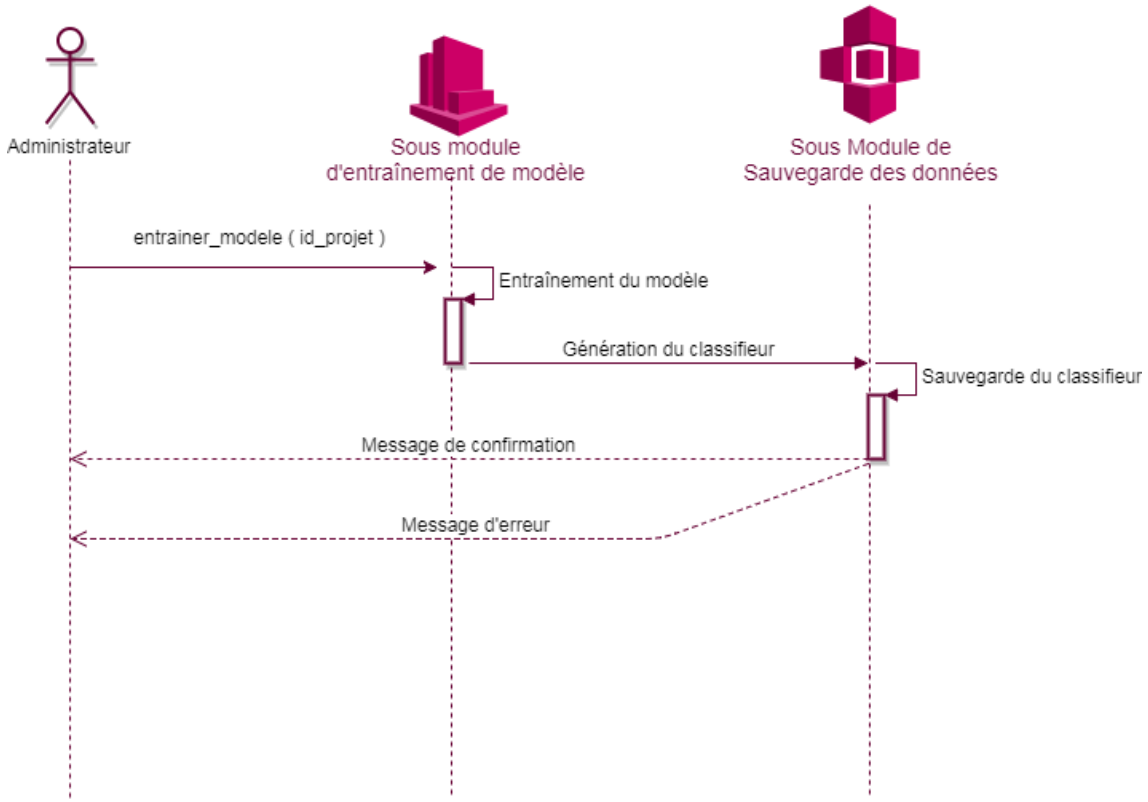

FIGURE 3.13 – Diagramme de séquence du cas d'utilisation Entraîner le modèle d'apprentissage

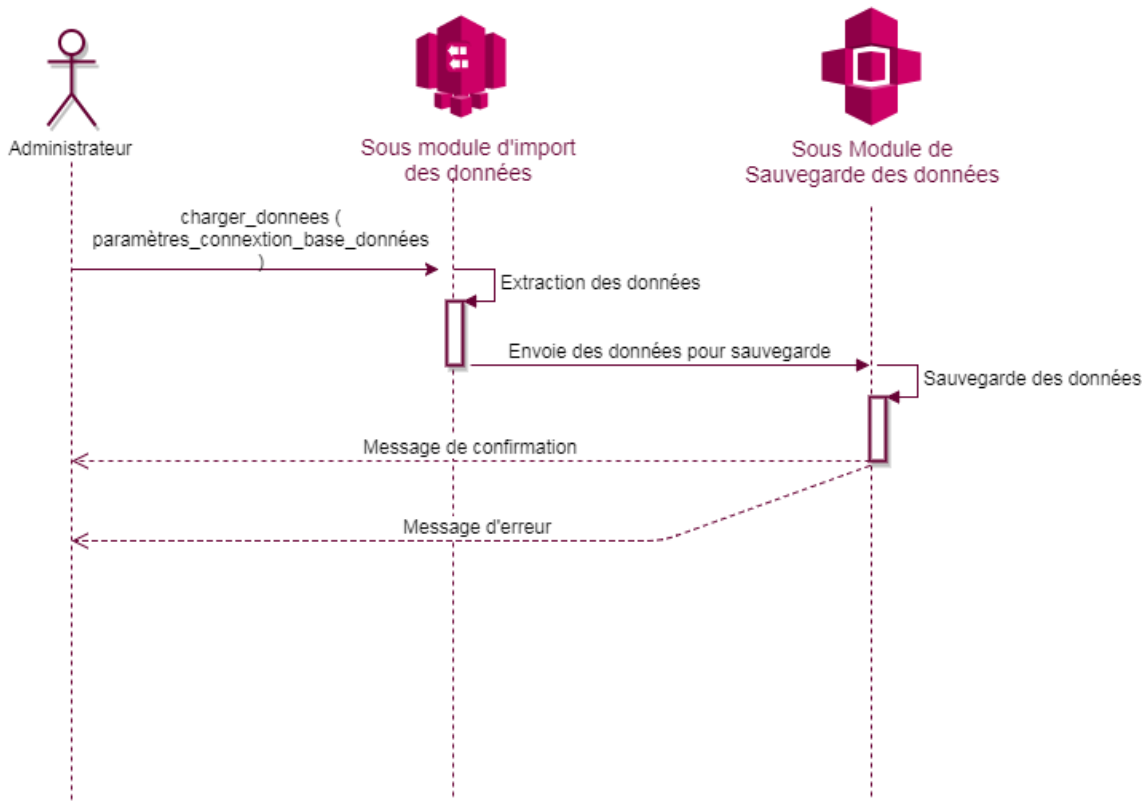

FIGURE 3.14 – Diagramme de séquence du cas d'utilisation Charger la base de données

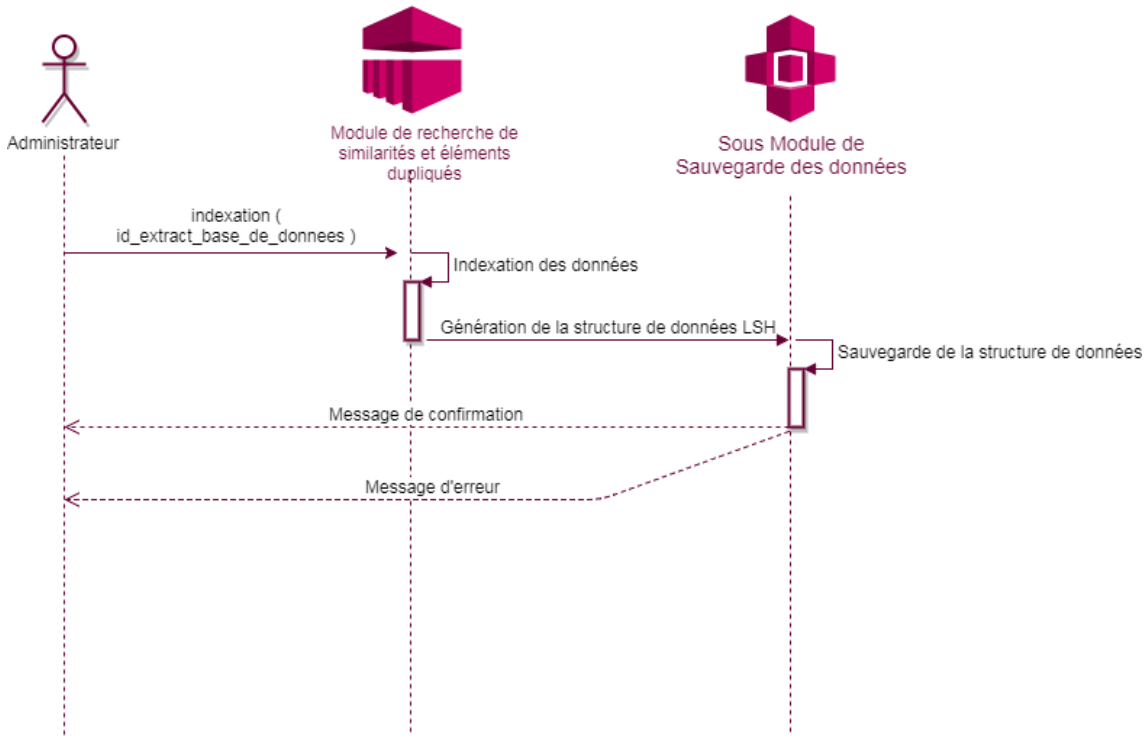

FIGURE 3.15 – Diagramme de séquence du cas d'utilisation Indexation des données

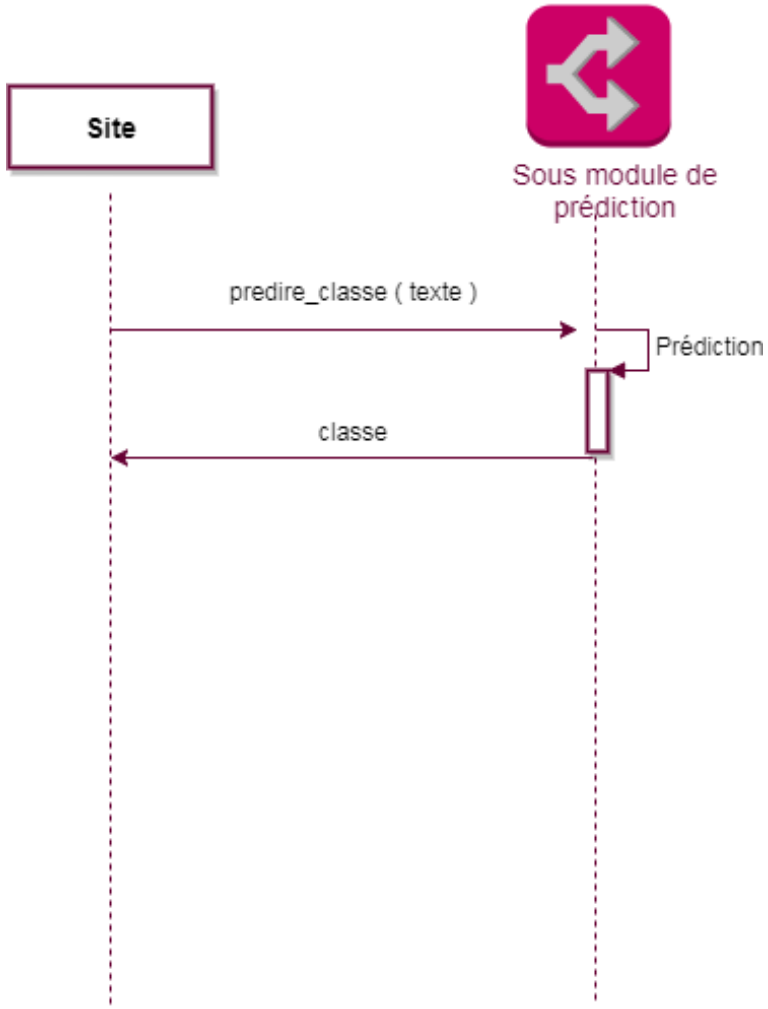

FIGURE 3.16 – Diagramme de séquence du cas d'utilisation Prédire la classe d'une entrée

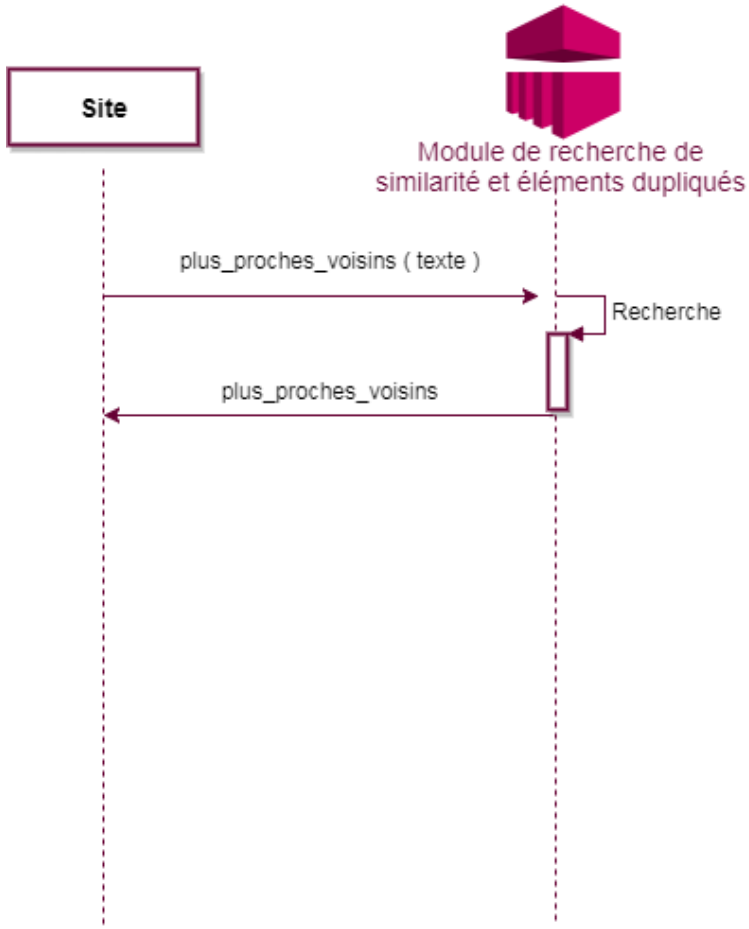

FIGURE 3.17 – Diagramme de séquence du cas d'utilisation Recherche des k éléments les plus similaires à une entrée

Chapitre 4

Implémentation

Ce chapitre est consacré à l'implémentation de la solution. Il présente les outils en termes de caractéristiques et rôle dans l'implémentation.

4.1 Présentation des outils

4.1.1 Les outils généraux

Ce sont les outils utilisés de manière générale dans le développement de logiciel, sans pour autant être spécifique aux problèmes de machine learning ou d'analyse de données.

Le SGBD MySQL

MySQL est un système de gestion de base de données relationnelles distribué sous licences GPL et propriétaire. Il est donc utilisable gratuitement.MySQL est compatible avec plusieurs systèmes d'exploitation et peut être combiné à d'autres langages de programmation.

Le Gestionnaire de version Git

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds. Sa principale fonction est de gérer l'évolution du contenu du code d'une application sous forme d'arborescence. Il fonctionne par indexation des fichiers manipulés d'après une somme de contrôle qui permet de savoir si un fichier a été modifié ou pas. C'est l'un des outil de versioning les plus utilisés notament par les plateformes de développement en équipe tel que github.com et bitbucket.com

Python

Python est un langage de script, interprété dont la philosophie insiste sur la lisibilité du code. Il est doté d'un typage dynamique et d'une gestion automatique de la mémoire tout en supportant plusieurs paradigmes de programmation : l'orienté objet, la programmation impérative, programmation fonctionnelle et procédurale. Disponible pour plusieurs de systèmes d'exploitation.

L'EDI PyCharm

PyCharm est un environnement de développement intégré utilisé pour développer en python. Il offre l'analyse de code, un débogueur graphique, la gestion des tests unitaires, l'intégration de logiciel de gestion de versions. Il est développé par l'entreprise tchèque JetBrains. Pycharm est multi-plateforme et fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, réalisé sous licence propriétaire, et en édition communautaire gratuite.

Le framework Django

Django est un cadre de développement web écrit en Python et qui fonctionne selon le principe MVC. Il a pour but de rendre le développement web 2.0 simple et rapide. Il comprend

- Un serveur web léger pour développer et tester ses applications sans pour autant les déployer
- Un système élaboré de traitement des formulaires
- Un cadre de cache pouvant utiliser différentes méthodes
- Un ORM (Object-Relational Mapping) qui permet la manipulation des bases de données à partir d'objet.

Redis

Redis est un système de gestion de base de données clef-valeur scalable, très hautes performances, écrit avec le langage de programmation C ANSI. Il a la particularité de fonctionner en mémoire vive et permet ainsi d'effectuer des requêtes rapides. Il supporte plusieurs types de structures de données tel que les chaînes de caractères, les listes et tableaux associatifs.

4.1.2 Les outils spécifiques à l'analyse de données et apprentissage artificiel

Scikit-learn

cikit-learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs2 notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria3 et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support.

4.2 Présentation de la solution

Dans cette partie nous présentons notre solution sous plusieurs angles de vue.

4.2.1 Architecture fonctionnelle

Dans cette architecture (Figure 4.1), l'administrateur se connecte à l'interface d'administration et c'est le gestionnaire d'API qui s'occupe de faire les vérifications nécessaire sur la légitimité de l'utilisateur. Ce module gère également l'interaction entre l'utilisateur et le reste du système. Il s'occupe du rendu des vues et de l'ochestration des différents composants du système pour effectuer les traitements voulus par l'utilisateur. Cette architecture correspond à la décomposition de la plateforme en module telle que présentée en section 3.3.3. Chaque module encapsulant un ensemble de fonctionnalité.

4.2.2 Architecture Technologique

L'architecture technologique (Figure 4.2) sur laquelle tourne la plateforme MLA comprendre trois principales composantes.

- **Un serveur web** dans lequel sont déployés les composantes web de notre système. Il s'occupe du rendu des pages aux clients.
- **Un serveur d'application** où sont déployés les composants applicatifs de la plateforme. Il comprend tout ce qui a trait aux opérations de machine learning et d'analyse des données.
- **Base de données**. Ici sont stockées toutes les données nécessaires au fonctionnement de la plateforme.
- **Un serveur de base de données en mémoire vive** : cet élément abrite la base de donnée tampon utilisée dans le processus de recherche d'éléments similaires que

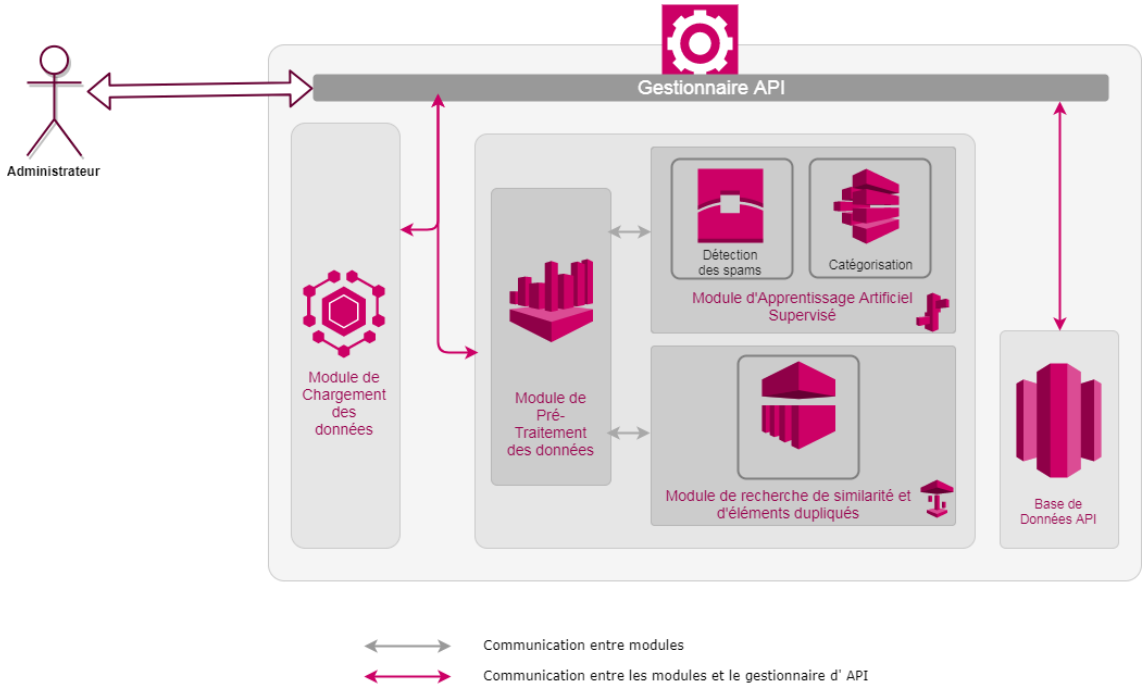

FIGURE 4.1 – Architecture fonctionnelle

nous avons défini

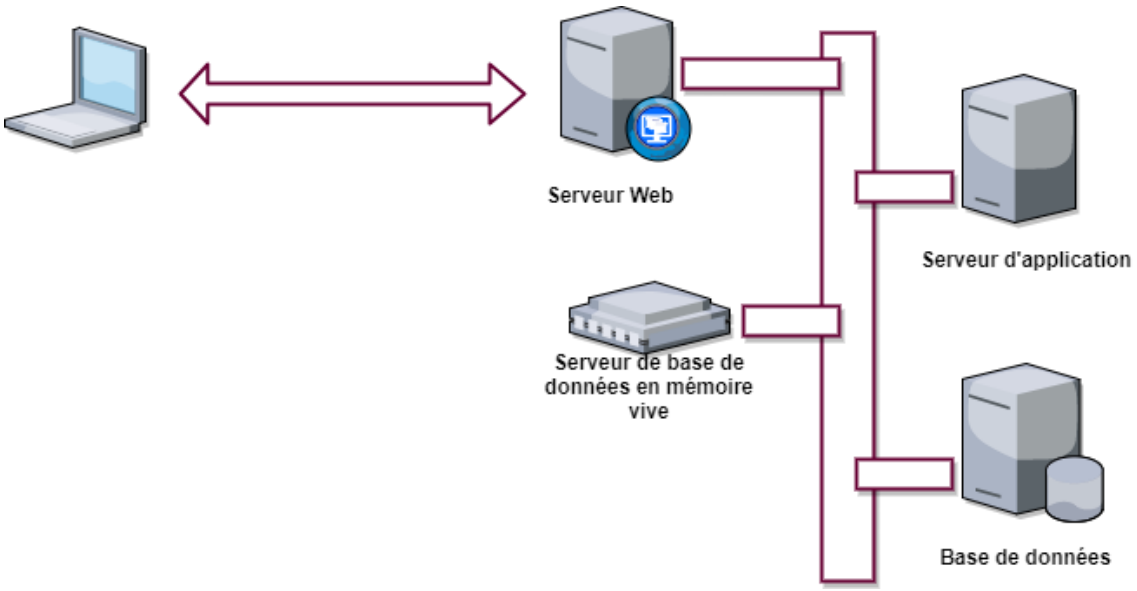

FIGURE 4.2 – Architecture technologique

4.2.3 Architecture applicative

Dans cette architecture (Figure 4.3) nous présentons les différentes applications qui entrent en jeux dans le fonctionnement de la plateforme.

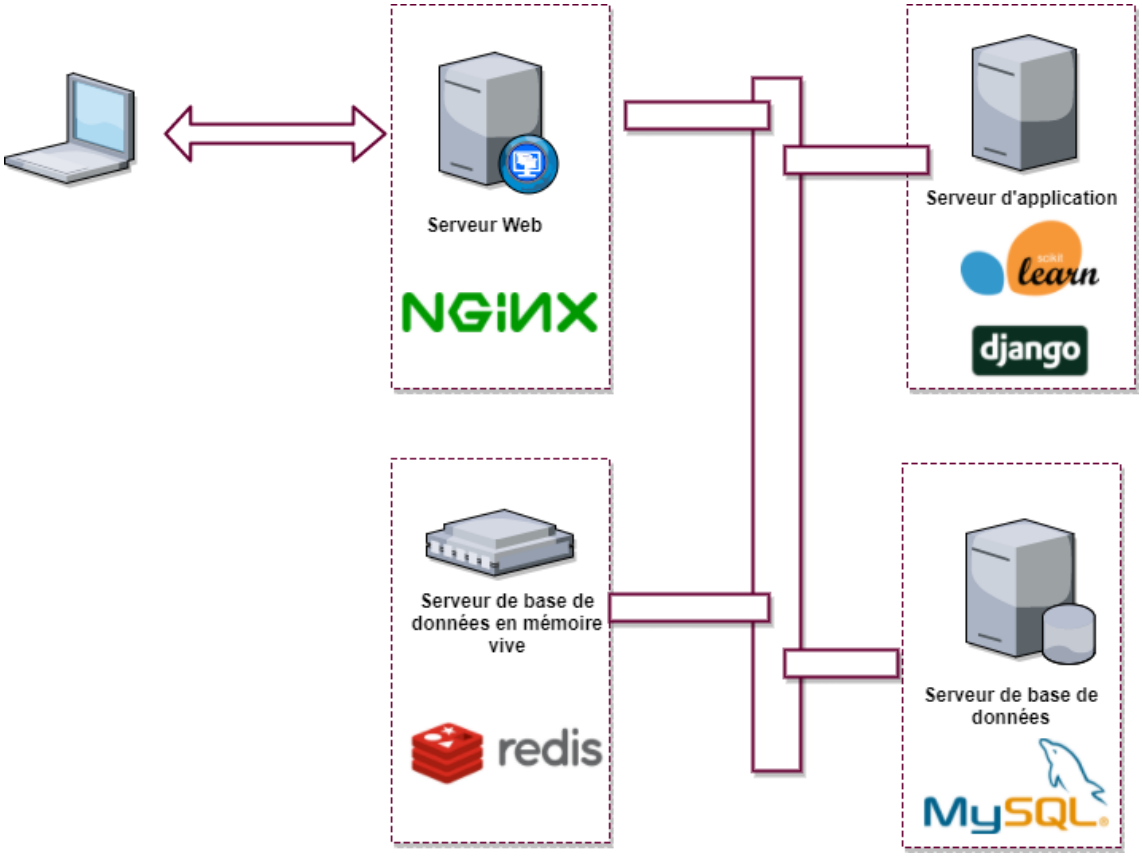

FIGURE 4.3 – Architecture applicative

Chapitre 5

Résultats

Dans cette partie, nous présenterons tout d'abord l'application dans son ensemble, ensuite il sera question de faire un point sur les évaluations de chacune des fonctionnalités implémentées au sein de l'application.

5.1 Application

Ici nous présentons l'application dans son fonctionnement. Tout commence avec l'authentification (Figure 5.1) où l'utilisateur renseigne son nom d'utilisateur et son mot de de passe.

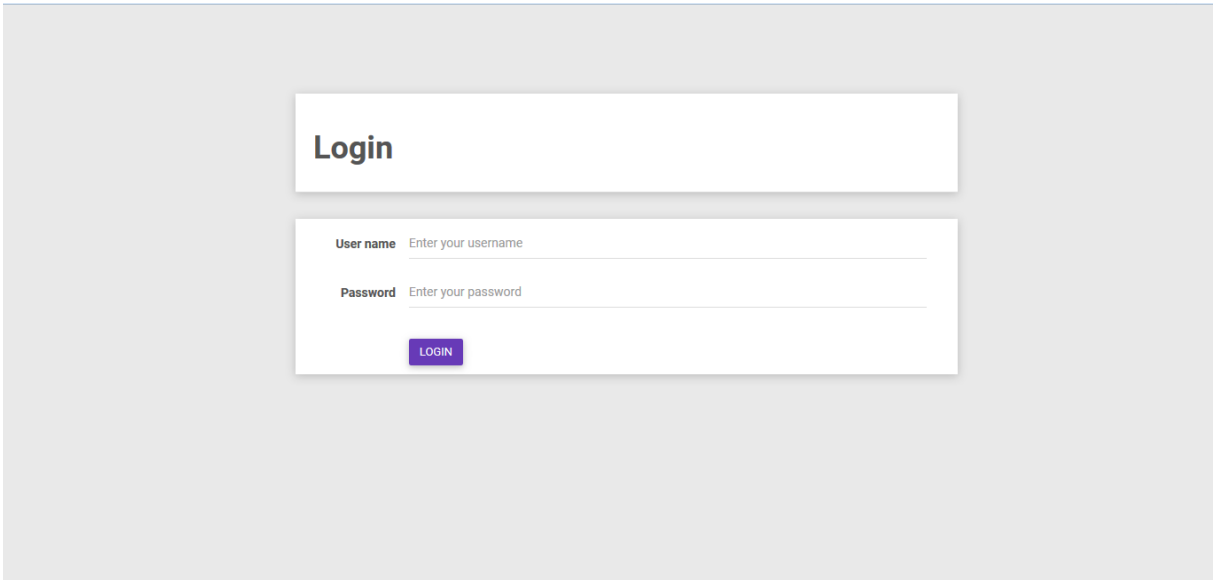

FIGURE 5.1 – Authentification

Une fois l'utilisateur connecté, il est redirigé vers une page d'accueil où lui sont présentés les projets qu'il a eu à créer.

5.1.1 Détection de Spams

Au départ, l'utilisateur n'a encore aucun projet (Figure 5.2).

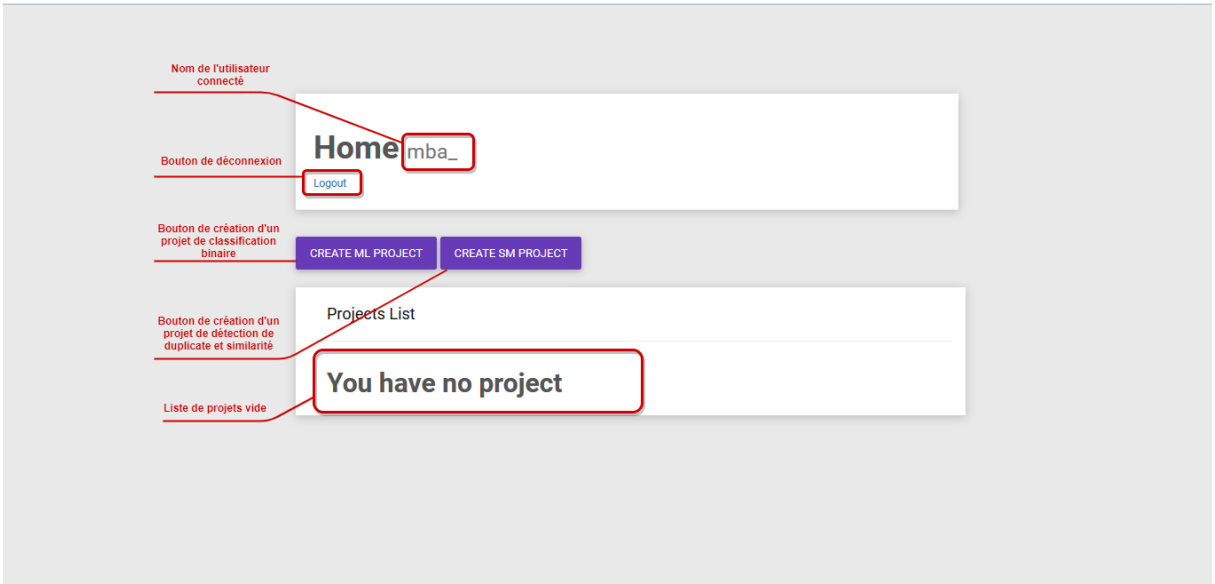

FIGURE 5.2 – Page d'accueil sans projet créé au préalable

L'utilisateur crée un projet en spécifiant son nom et une description qui permettra d'avoir un peu plus de détail sur le projet (Figure 5.3).

Le projet créé apparaît désormais sur la page d'accueil (Figure 5.4). L'utilisateur peut cliquer sur le projet pour en voir les détails et l'initialiser.

Un projet nouvellement créé n'a encore aucun jeu de données (Figure 5.5), il faut donc en télécharger sous forme de fichier CSV (Comma-separated Values) en spécifiant un nom au fichier (Figure 5.6)

Le jeu de données télécharger est étiqueté, on lui donne un "tag" (Figure 5.7). Plusieurs fichiers peuvent être télécharger et à chacun on affecte une étiquette qui correspond à la classe à laquelle le jeu de données correspondant appartient.

Une fois les jeux de données téléchargés et étiquetés (Figure 5.8), on peut initialiser l'entraînement du modèle. Une instance de prédiction est alors créée et associée au projet (Figure 5.9 à. L'instance de prédiction servira à la prédiction de nouvelles entrées lors de l'apppel à l'API de MLA. A noter que tout appel à l'API se fait sur un projet dont on a les droits (c'est à dire qu'on a créé).

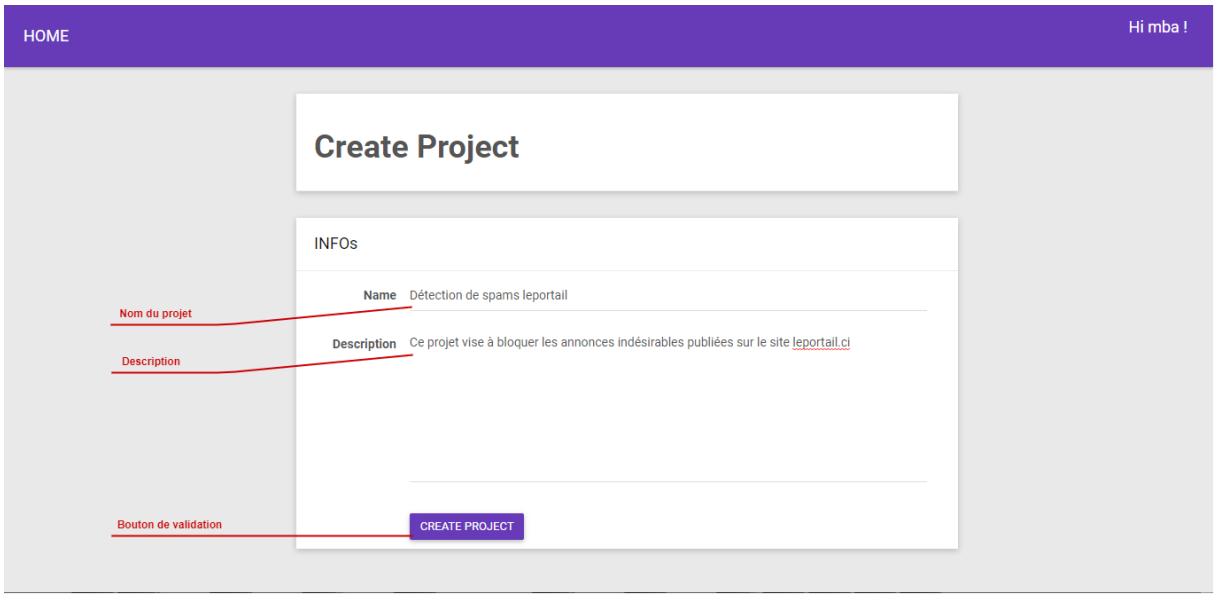

FIGURE 5.3 – Création d'un projet

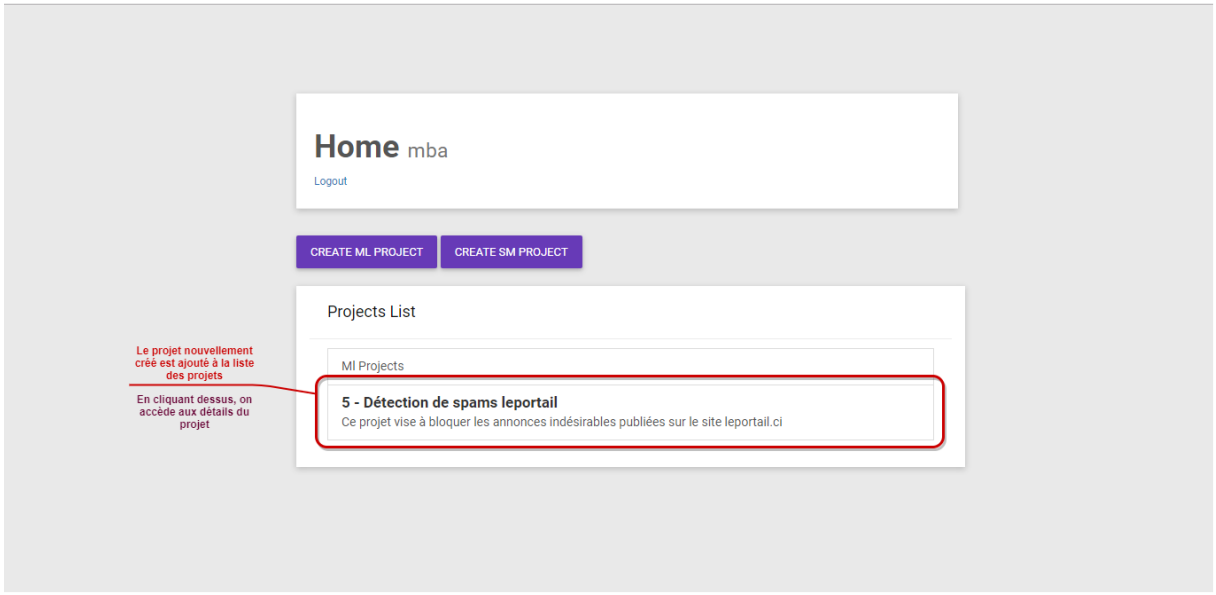

FIGURE 5.4 – Page d'accueil avec projet créé au préalable

5.1.2 Détection de similarité

Une fois authentifié, l'administrateur peut créer un projet de type détection de similarité où il renseigne d'abord le nom et une description au projet Figure 5.10, ensuite il renseigne les paramètres nécessaire à l'initialisation de la base de recherche et de la base tampon Figure 5.11

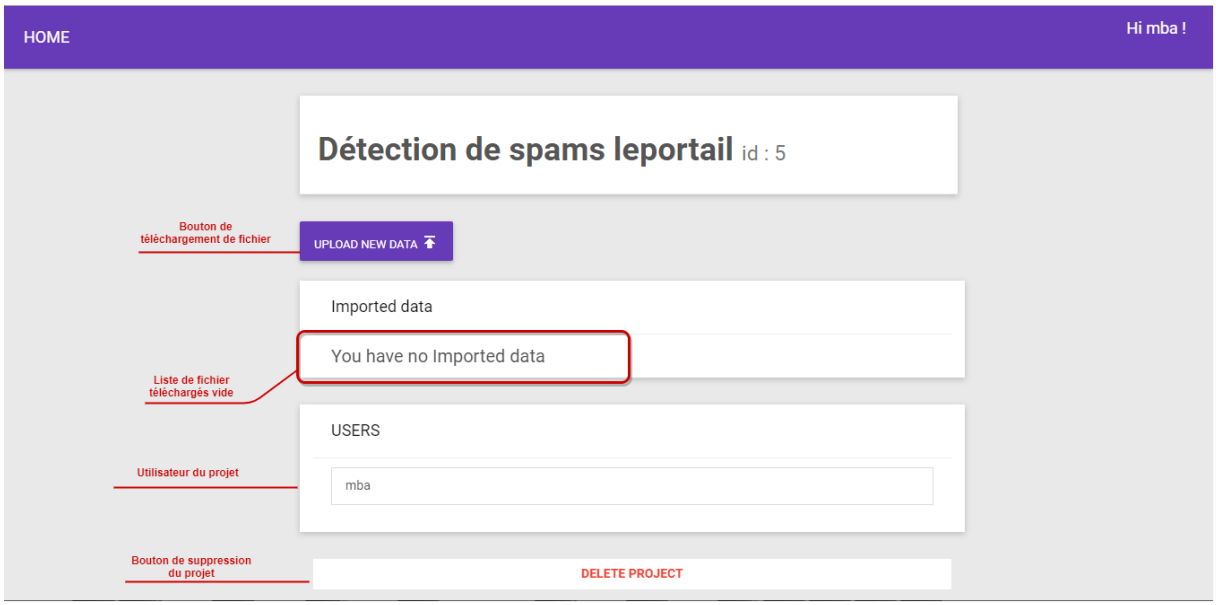

FIGURE 5.5 – Projet sans jeu de données

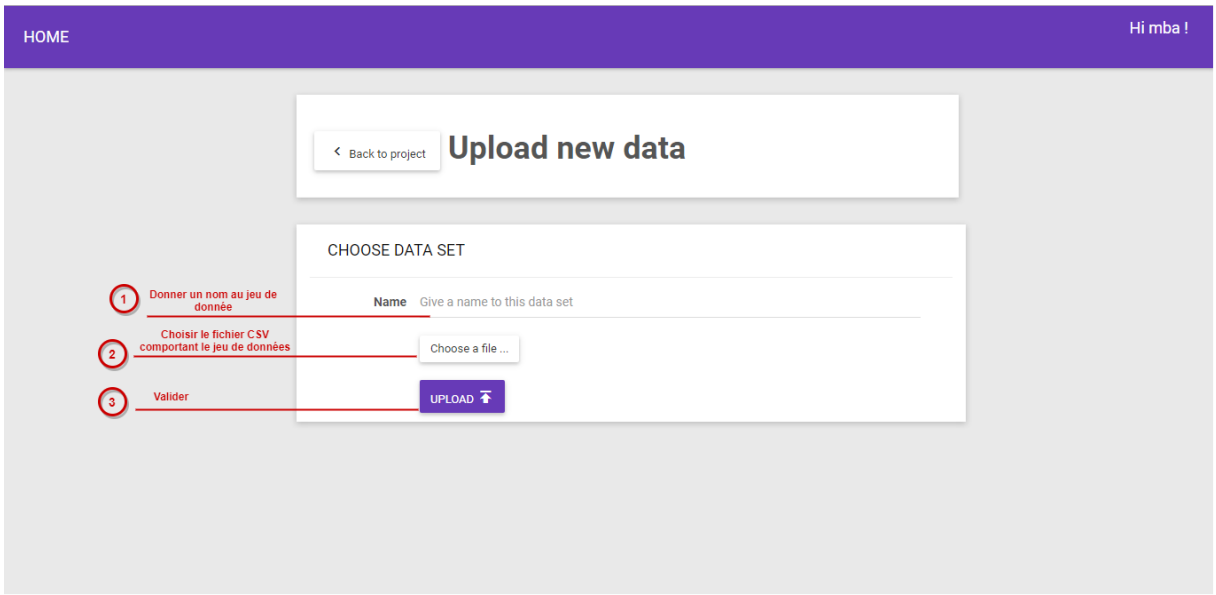

FIGURE 5.6 – Télécharger un fichier

Le projet nouvellement créé apparaît dans la liste des projets **??** et il peut être initialisé Figure 5.12

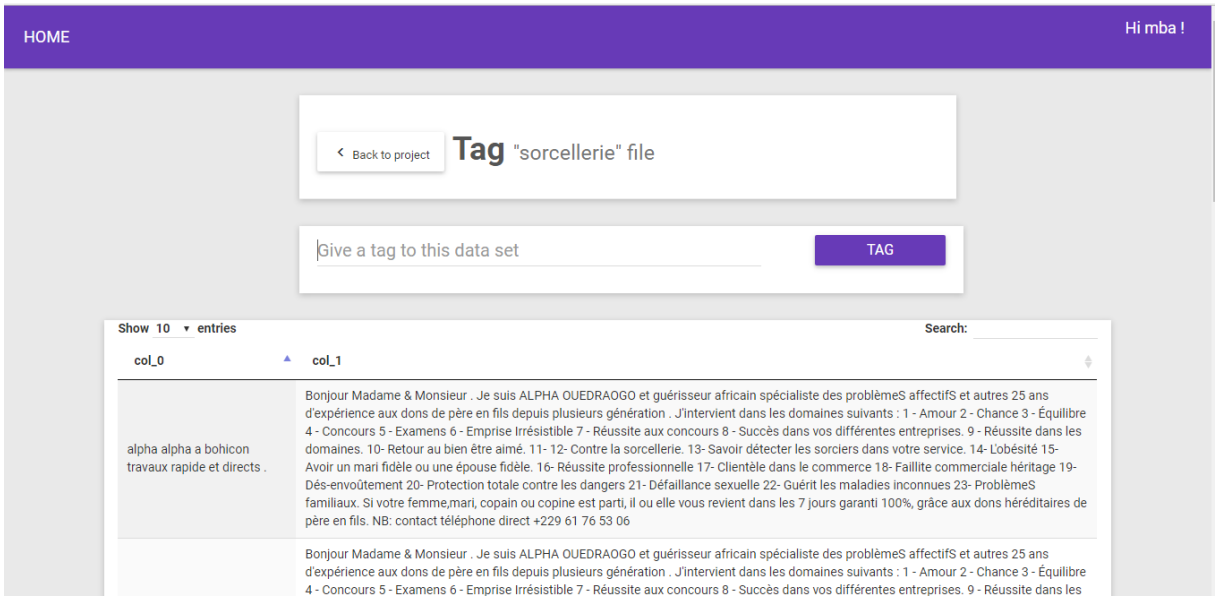

FIGURE 5.7 – Etiqueter un jeu de données

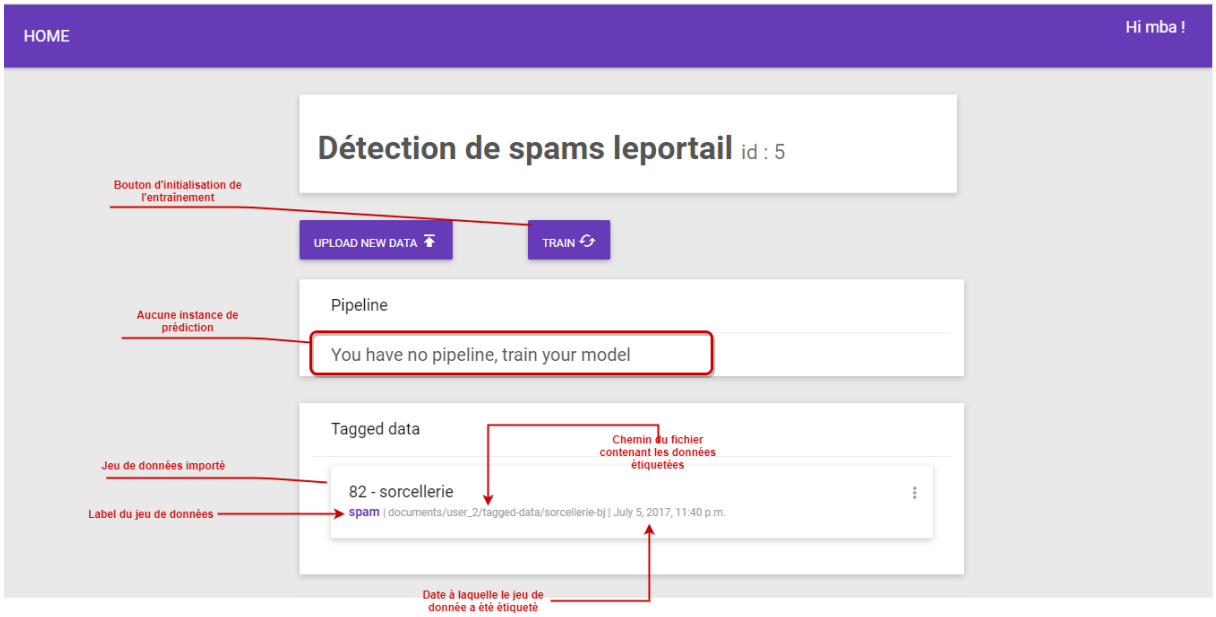

FIGURE 5.8 – Projet contenant des jeu de données étiquetés mais pas encore de modèle entraîné

5.2 Evaluations

L'objectif à était multiple. Il était question de :

— De réduire la charge de travail dû à la modération des annonces en bloquant le

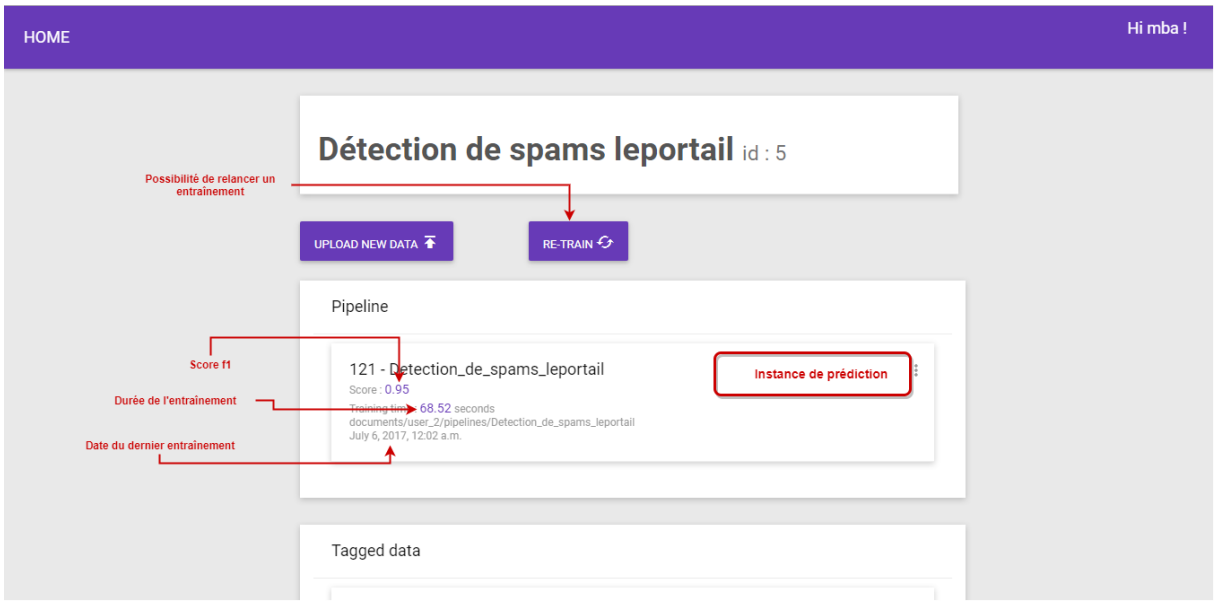

FIGURE 5.9 – Projet ayant une instance de prédiction

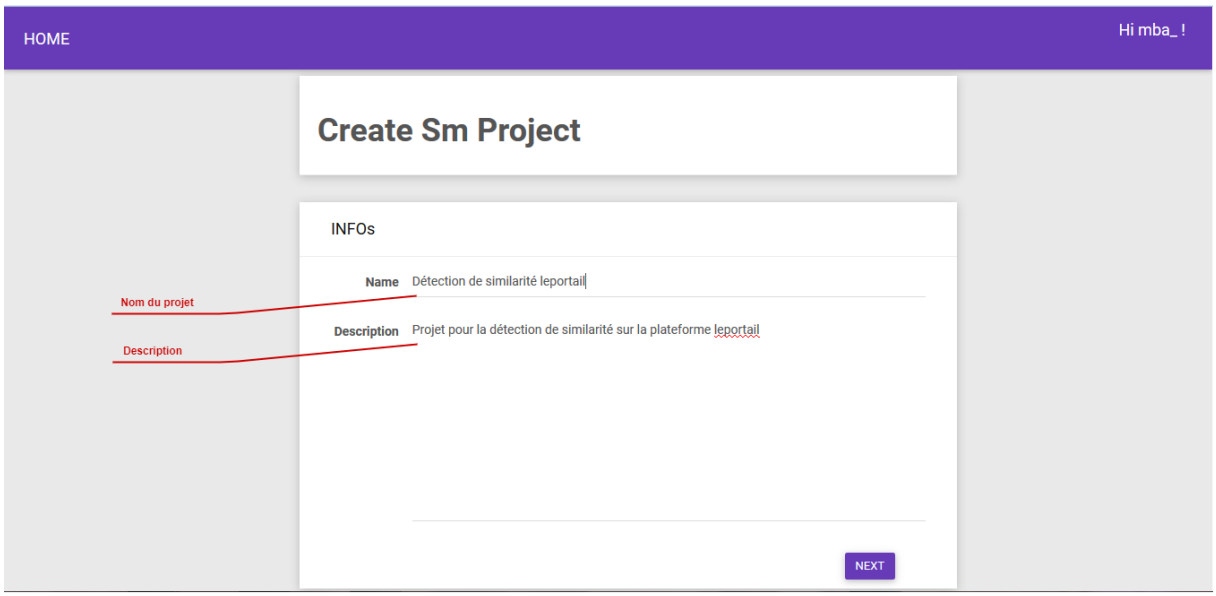

FIGURE 5.10 – Créer un projet de détection de similarité - Etape 1

minimum d'annonces possible

— De réduire au maximum possible le nombre de Spams publiées sur les sites. C'est-à-dire les annonces ayant trait à l'occultisme, les annonces adultes et les arnaques du type voyage à l'étranger et offre d'emploi dans des organismes internationaux

— De réduire au maximum le nombre d'annonces dupliquées qui sont publiées sur les sites.

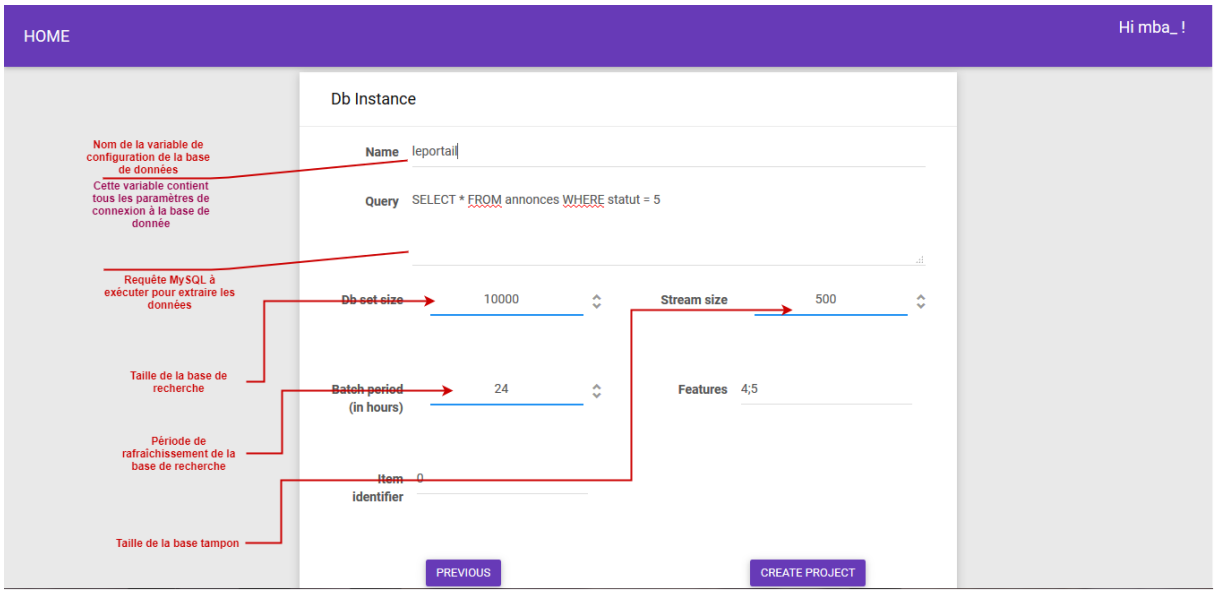

FIGURE 5.11 – Créer un projet de détection de similarité - Etape 2

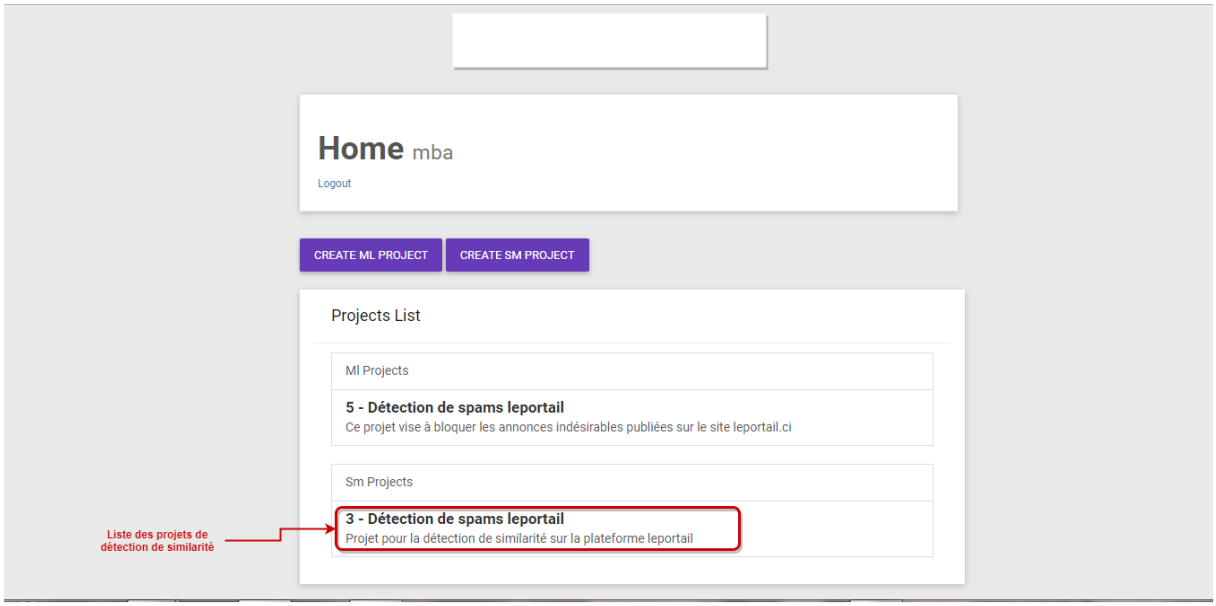

FIGURE 5.12 – Liste des projets avec projets de détection de similarité

De ce fait pour évaluer notre solution nous avons mesuré cinq métriques :

— Le rappel qui est la proportion de spams sur le site qui sont déclarés spam par MLA
— La précision qui est la proportion d'annonces légitimes qui sont bloquées par MLA
— Le score f1
— le pourcentage d'annonces dupliquées qui sont détectées
— Le nombre d'annonces renvoyées par jour en modération, c'est-à-dire les annonces

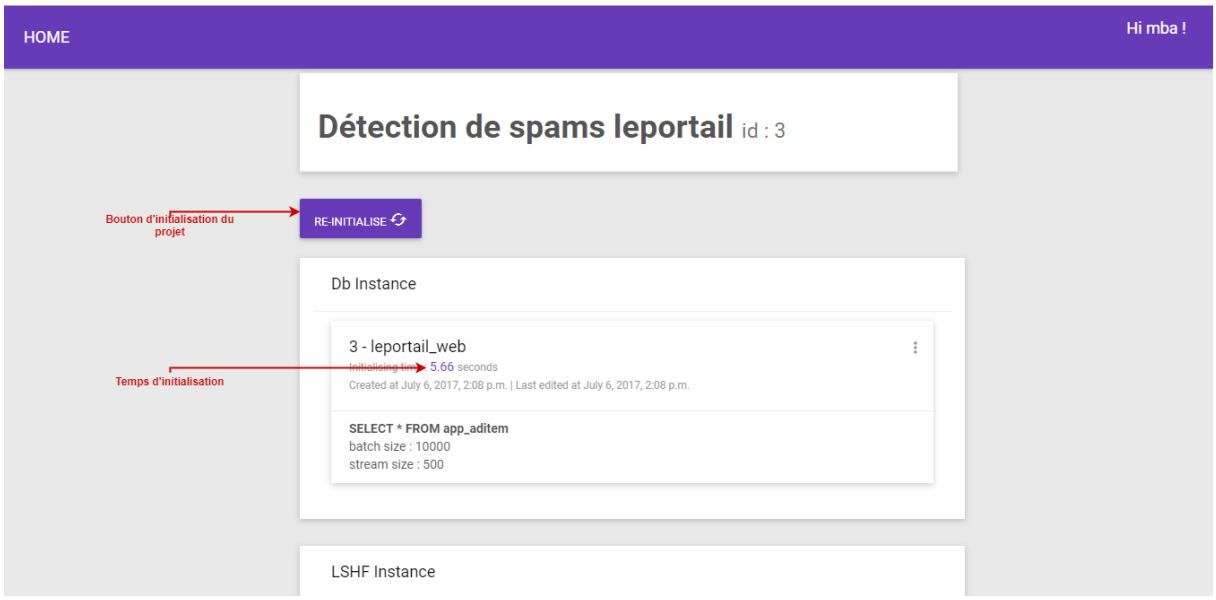

FIGURE 5.13 – Projet de détection de duplicate initialisé

déclarées comme spam par MLA

5.2.1 Detection de spams

Contexte

Pour sa première version, le détecteur de spam a été déployé en aval du processus de création d'annonce :

1. les utilisateurs créent leurs annonces comme ils l'ont toujours fait
2. Un cron passe à intervalle de temps régulier et pour toute nouvelle annonce qui a été créé depuis son dernier passage, il fait un appel à l'API de MLA pour savoir si l'annonce est un spam ou non. Il lui envoie le titre et la description de l'annonce et l'API de MLA lui retourne le statut (Spam ou Non Spam) avec la probabilité $p = 1 \breve{}$probabilité de se tromper
3. Pour toutes les annonces qui sont spams, celles-ci sont bloqués et n'apparaissent plus sur le site
4. Seuls les annonces bloquées sont désormais transférées dans l'interface de modération afin de débloquées les annonces sur lesquelles le détecteur se serait trompé

Il est donc important que toutes les annonces spams soient bloquées, au risque de bloquer des annonces qui ne le sont pas. Car des annonces légitimes, même si elles sont bloquées par le détecteur, seront réactivées plus tard manuellement.

Chaque annonce inspectée par MLA est marquée afin d'être inspectée une seule fois.

Jeu de données

La constitution du jeu de données a consisté à réunir un ensemble d'exemples de spams d'une part et de non spams d'autres part. Pour ce faire, nous avons d'abord défini la notion de spam. Qu'est-ce qu'une annonce spam ? Un spam est une annonce indésirable des plateformes Wariba (Kerawa et Leportail). Ce sont :

— des annonces qui ont trait à la sorcellerie, l'occultisme et toute autre pratique mystique
— des annonces de type érotique, ayant trait à la sexualité, aux rencontres adultes
— Les arnaques, qui ici sont des propositions d'emploi fausses généralement attribuées à des organismes internationaux ainsi que des offres de voyage pour l'étranger et d'autres exemples que nous présenterons avec plus de détails par la suite

Une fois la notion de spam définie, nous avons constitué une collection de données qui répond aux critères suscités. Voici un tableau qui récapitule pour chaque type d'annonces indésirables le nombre d'elements qui ont pu être réuni ainsi que des exemples illustratifs :

Type	Nombre d'exemples
Sorcellerie	813
Erotique	5331
Arnaques	649

TABLE 5.1 – Extrait du jeu de données d'annonces spam

Au total nous avons 6793 annonces de type **spam**.

En ce qui concerne les annonces de type **non spam** nous avons réuni un ensemble d'annonces légitimes représentatif de l'ensemble des catégories du site. Voici un extrait du nombre d'éléments par catégorie :

Catégories	Nombre d'exemples
Téléphones	734
Recherche d'emploie	1860
Jeux consoles	800

TABLE 5.2 – Extrait du jeu de données d'annonces non spam

Au total notre base d'apprentissage initiale comporte 5411 exemples de non spams.

Le jeu de données, dans chaque cas, se doit d'avoir des exemples variés pour chacune des catégories à prédire afin d'être représentatif celles ci. De ce fait, pour minimiser les éventuelles erreurs que nous pourrions commettre en essayant de construire d'un seul

trait l'ensemble des données d'apprentissage, nous avons décidé de procéder par iteration. L'objectif est d'aller sur la base d'un jeu de donnés initial que nous ajusterons au fur et à mesure des apprentissages afin d'en améliorer le score.

Entraînement du modèle et évaluation

Les données sont injectées dans le modèle dont l'entraînement produit le classifieur. Pendant cette phase, l'évaluation du classifieur est aussi faîte par validation croisée (voir sous-section 3.1.3).

Un premier entraînement a été fait puis le classifieur a été utilisé en production. Son observation pendant près d'1 mois nous a permis de noter qu'il avait du mal à classer comme non spam les annonces ayant trait aux téléphones, aux demandes d'emploi de particulier, à la restauration de plats africains et aussi à l'immobilier.

Du fait de ce constat, nous avons injecté dans le modèle des annonces des types suscités en leur donnant l'etiquette de non spam. Les résultats (macro-score) obtenus sont les suivants :

Rappel	**Précision**	**f1 score**
0.952	0.949	0.951

TABLE 5.3 – Scores obtenus pour le détecteur de spams

Avec ce score 100% des annonces adulte et occultes sont bloquées. Néanmoins, le classifieur se comporte moins bien avec les offres d'emploi potentiellement fausses car celles-ci dans leur forme ne sont pas très différentes des offres d'emploi légitimes.

5.2.2 Detection d'annonces dupliquées

Contexte

La fonctionnalité de détection d'annonces dupliquées a été mise en production et fonctionne directement à la création d'annonce. Lorsqu'un utilisateur crée une annonce :

1. On check via l'api de MLA si une annonce existante lui est identique
2. Si oui un message d'erreur est envoyé à l'utilisateur lui spécifiant de modifier le contenu de son annonce

Base de recherche

L'algorithme de forêt LSH, malgré qu'il soit conçu pour de large base de recherche présente cependant des limites. Kerawa présente plus de 300 000 annonces publiées sur sont site

et leportail en a près de 200 000. Déjà avec 100 000 entrées dans la base de recherche, l'algorithme commence déjà à être peu performant avec un temps de réponse de l'ordre de 10 secondes pour des annonces similaires retournées peu pertinentes. Après des discussions avec l'entreprise, il en ressort que pour un site de petites annonces, il ny'a une forte nécessité de rechercher des annonces dupliquées parmi toutes annonces publiées. En effet, il y'a comme une notion d'expiration des annonces qui fait que des annonces vieilles ne représente plus aucun intérêt pour les utilisateurs. De ce fait une base de recherche plus petite pourrait tout aussi bien résoudre le problème.

Alors pour la première version de l'application,nous sommes partit sur la base qu'une annonce de plus de 20 jours était suffisament vieille pour ne pas faire parti de la base de recherche. Avec un rythme pouvant aller jusqu'à 500 annonces par jour et par site, la taille de la base de recherche choisie a été de **10 000** annonces pour chaque site.

Evaluation

Le détecteur de duplicate a d'abord été déployé sur leportail après que des tests en interne ai été effectué. Il s'agissait de :

1. créer une annonce avec un certain contenu
2. essayer de créer une autre annonces avec le même titre et la même description
3. L'annonce devrait être signaler comme dupliquée et dans ce cas, apporter de manière progressive des modifications à l'annonce jusqu'à ce que celle-ci ne soit plus bloquée

A partir de ces tests nous avons fait la remarque que l'application détectait 100% des annonces identiques (ayant exactement le même contenu). En ce qui concerne les annonces identiques à quelques mots ou expression près, il fallait fixer un seuil de distance en dessous duquel deux annonces sont déclarés comme identiques. Nous l'avons fixé au départ à 0.07 mais après observation de du fonctionnement pendant une journée le seuil de **0.2** a été retenu. Ainsi deux annonces ayant une distance inférieure à 0.2 sont déclarées comme identiques.

Evaluation

Un remarque importante à noter est que les auteurs d'annonces ayant trait aux offres d'emploi fictive avait tendance à publier plusieurs fois la même annonce. Avec le système mis en place, celles-ci se voient désormais bloquées et leurs auteurs ne prennent pas la peine de modifier. Du coup elles ne sont plus publiées sur le site. Ce module offre donc une solution partielle immédiate au problème d'offres d'emploi fictives publiées sur le site.

5.2.3 Charge de travail dû à la modération

Un autre aspect important de notre travail était justement la diminution de la charge de travail dû à la modération des annonces. D'une part le module de détection de spams devait réduire la nécessité de parcourir un gros volume d'annonces afin d'identifier les spams. Pour évaluer cela, voici un graphique Figure 5.14 qui montre l'évolution conjointe du nombre d'annonces publiées par jour, le nombre de non spams et le nombre de spams détectés : données.

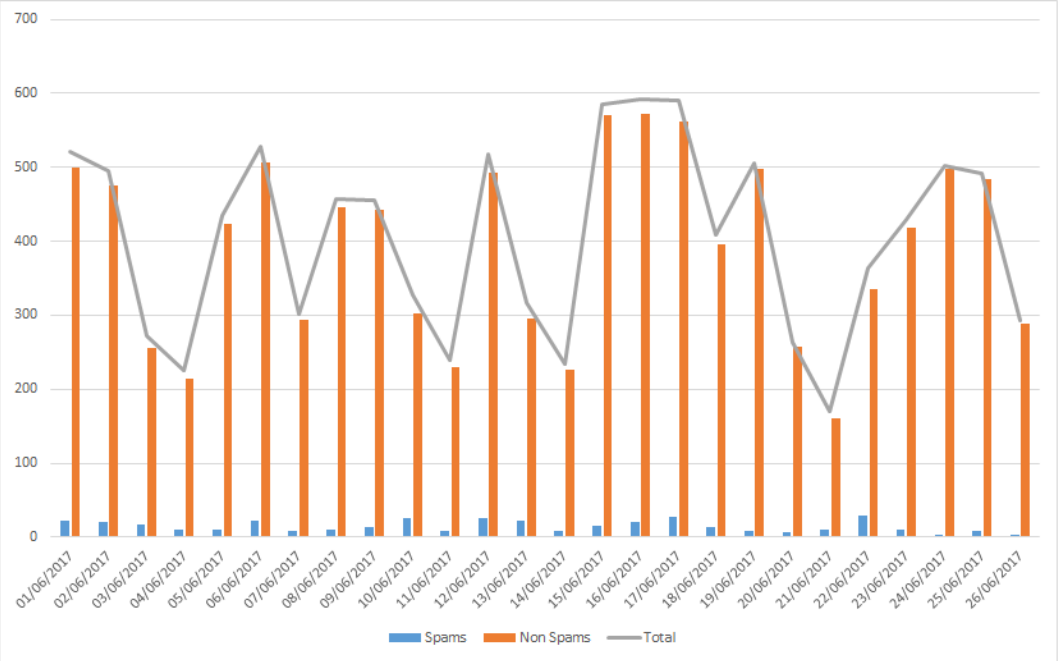

FIGURE 5.14 – Nombre d'annonces Spams et Non Spams par jour sur 20 Jours

On remarque que le nombre d'annonces modérées par jour a drastiquement chuté. Il est passé d'une moyenne de 413,85 annonces par jour à 15,95 par jour. Ce qui signifie une diminution de près de 97%.

Conclusion Générale

La problème abordé dans ce mémoire était celui de mettre sur pied un outil pour la gestion intelligente et automatique de la qualité des données produites par des utilisateurs sur des sites de petites annonces. L'outil mis sur pied devait être capable de : détecter automatiquement les annonces publiées qui ont un contenu indésirable (vis à vis de la politique du site), de permettre une recherche rapide d'annonces similaires ou dupliquées à un annonce donnée et de faire de la classification automatique des annonces dans la catégorie du site qui lui correspond le mieux. Nous avons dans un premier temps analysé les trois problématiques afin de circonscrire le domaine de recherche des solutions envisageable pour leurs résolution. Nous en sommes sorti avec deux axes principaux : **la classification supervisée** et **la recherche de plus proche voisin**. Dans le premier cas, nous avons présenté les techniques existantes pour la résolution de ce type de problème avant d'en choisir une selon la recommendation faite par le projet scikit-learn. Néanmoins cette recommendation présentait des limites en ce qui concerne la catégorisation automatique des annonces, lui aussi étant un problème de classification supervisée. L'approche prise a alors été de s'appuyer sur la résolution du problème de recherche de similarité et utiliser ses résultats pour implémenter une catégorisation automatique. De plus, nous avons défini une démarche formelle de la classification supervisée sur laquelle nous nous sommes appuyés afin de mettre sur pied notre solution. Pour le problème de recherche de similarité en lui même les solutions existantes présentaient des limites car avaient toutes du mal à produire un résultat convenable pour la recherche de plus proches voisins dans une base de recherche dynamique de grande taille avec des éléments de très grande dimension. Nous avons proposé une nouvelle approche qui combine les techniques d'indexation par forêt LSH et la recherche linéaire afin de proposer une solution à ce problème. Cette solution utilise deux bases de recherche : une première de petite taille sur laquelle est appliquée la recherche linéaire et une deuxième base de grande taille où l'indexation par forêt LSH est appliquée.

La plateforme obtenue, dénomée MLA (Machine Learning Application), est en production et intègre déjà les fonctionnalités de détection de spams et de recherche de similarité. La fonctionnalité de catégorisation quant à elle a déjà été conçue et son implémentation est en cours. La mise en production de notre solution a contribué à réduire de 100% les annonces spams de type occulte et adulte publiées sur le site, de plus on note une diminution de

90% de la charge de modération manuelle des annonces publiées qui est désormais presque inexistante. La fonctionnalité de recherche de similarité permet à présent de détecter 100% des annonces dupliquées lorsque celles-ci sont créées.

La plateforme peut encore être améliorée d'abord en terminant la fonctionnalité de catégorisation automatique. De plus des optimisations peuvent encore être faîtes quant à la précision du détecteur de spam. Mais pour aller plus loin, un aspect important à prendre en compte lorsqu'on veut résoudre un problème lié à la qualité des données produites par des utilisateurs c'est la crédibilité de ceux-ci. En effet chaque utilisateur a un comportement qui peut être étudié afin de déterminer sa prédisposition à produire du contenu de mauvaise qualité. Cet aspect, celui de la **qualification des utilisateurs** constitue le principal axe de perspective afin d'améliorer les résultats produits par notre plateforme.

Bibliographie

[Bentley 1975] Jon Louis Bentley. *Multidimensional binary search trees used for associative searching.* 1975.

[Buitinck 2013] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt et Gaël Varoquaux. *API design for machine learning software : experiences from the scikit-learn project.* In ECML PKDD Workshop : Languages for Data Mining and Machine Learning, pages 108–122, 2013.

[Chih-Wei Hsu] Chih-Jen Lin Chih-Wei Hsu. *A Comparison of Methods for Multi-class Support Vector Machines.* `https://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.pdf`. 03-06-2017.

[Clémençon 2011] Stéphan Clémençon. *Classification supervisée et SVM - Pesto Web-Mining.* 2011.

[Garg] Nikhil Garg. *What is the LSH Forest algorithm ?* `https://www.quora.com/What-is-the-LSH-Forest-algorithm`. 03-06-2017.

[Gionis 1999] Rajeev Motwani Gionis P. Indyk. *Similarity Search in High Dimensions via Hashing.* Proceedings of the 25th Very Large Database (VLDB) Conference, 1999.

[J. 1977] Stone C. J. *Consistent nonparametric regression.* Annals of Statistics, no. 5, pages 595—-620, 1977.

[Jaccard 1901] Paul Jaccard. *Bulletin de la Société Vaudoise des Sciences Naturelles.* pages 241,272, 1901.

[Jones 1972] Spärck Jones. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval.* Journal of Documentation, pages 11–21, 1972.

[Kosub] Sven Kosub. *A note on the triangle inequality for the Jaccard distance.*

[Kotsiantis 2007] S. B. Kotsiantis. *Supervised Machine Learning : A Review of Classification Techniques.* 2007.

[Liefooghe 2013] Arnaud Liefooghe. *Analyse de comportements avec Twitter, Classification supervisée - Université de Lille 1.* 2012-2013.

[Luhn 1957] Hans Peter Luhn. *A Statistical Approach to Mechanized Encoding and Searching of Literary Information.* IBM Journal of research and development, 1957.

[Mayank Bawa] Prasanna Ganesan Mayank Bawa Tyson Condie. *LSH Forest : Self-Tuning Indexes for Similarity Search.*

[Mehryar Mohri 2012] Ameet Talwalkar Mehryar Mohri Afshin Rostamizadeh. *Foundations of Machine Learning.* The MIT Pres, 2012.

[OQLF] Office québécois de la langue française OQLF. *Combien de mots ?* `https://www.oqlf.gouv.qc.ca/actualites/capsules_hebdo/phistoire_nombre_20030123.html`. 03-06-2017.

[Piotr Indyk 1998] Rajeev Motwan Piotr Indyk. *Approximate Nearest Neighbors : Towards Removing the Curse of Dimensionality.* Proceedings of 30th Symposium on Theory of Computing, 1998.

[R. 1957] Bellman R. *Dynamic Programming.* Princeton University Press, 1957.

[Rajaraman 2011] A. Rajaraman et J. D Ullman. *"Data Mining"*. In Mining of Massive Datasets, pages 1–17, 2011.

[Refaeilzadeh 2009] Payam Refaeilzadeh, Lei Tang et Huan Liu. Cross-validation, pages 532–538. Springer US, Boston, MA, 2009.

[Richard O. Duda 2001] David G. Stork Richard O. Duda Peter E. Hart. *Pattern Classification.* Wiley-interscience, 2001.

[Sariel Har-Peled 1998] Rajeev Motwani Sariel Har-Peled Piotr Indyk. *Approximate Nearest Neighbor : Towards Removing the Curse of Dimensionality.* 1998.

[scikit learn] scikit learn. *scikit-learn, Machine Learning in Python.* `http://scikit-learn.org/stable/`. 21-04-2017.

[T. 2003] Pavlenko T. *On feature selection, curse of dimensionality and error probability in discriminant analysis.* Journal of Statistical Planning and Inference, no. 115, pages 565–584, 2003.

[Tretyakov 2004] Konstantin Tretyakov. *Machine Learning Techniques in Spam Filtering - Institute of Computer Science, University of Tartu.* 2004.

[Vladimir Vapnik 1998] Corinna Cortes Vladimir Vapnik. *Support-Vector Network.* 1998.

[Wael H. Gomaa 2013] Aly A. Fahmy Wael H. Gomaa. *A Survey of Text Similarity Approaches.* In International Journal of Computer Applications, 2013.