

ACADÉMIE DE MONTPELLIER
UNIVERSITÉ MONTPELLIER II
--- SCIENCE ET TECHNIQUES DU LANGUEDOC ---

MÉMOIRE DE STAGE DE MASTER

SPÉCIALITÉ : Recherche en Informatique
Mention : Informatique, Mathématiques, Statistiques

Méthodes de clustering flou appliquées au
« filtrage parental » d'Internet

NICOLAS PASSALACQUA

Date de soutenance: 21 juin 2006
Effectué au LGI2P sous la direction de GERARD DRAY

Remerciements

Je tiens tout d'abord à exprimer toute ma gratitude à Monsieur GERARD DRAY, pour son encadrement, sa présence, ses conseils et tout le soutien qu'il m'a accordé durant ce stage.

Je tiens à remercier également Monsieur PASCAL PONCELET, responsable du groupe de recherche KDD (Knowledge Discovery for Decision making), pour m'avoir accueilli dans son équipe.

Je souhaite aussi remercier tout l'équipe du LGI2P pour leur bonne humeur permanente et l'atmosphère de travail agréable qui en découle.

Je remercie également Monsieur Roland Ducournau, Responsable de la Spécialité Recherche en Informatique du Master mention IMS à l'université Montpellier II, de m'avoir accueilli au sein du département Informatique.

Table des matières

Introduction.....	4
Chapitre 1. Problématique.....	5
Chapitre 2. One-Class.....	9
2.1sélection d'attributs (feature selection).....	11
2.1.1Définition.....	11
2.1.2problématique apporté par le One-class.....	12
2.2 Techniques de classification non supervisée.....	13
2.2.1Principes de base.....	13
2.2.2Mesures de similarité.....	15
2.2.3Quelques méthode de clustering.....	16
2.2.3.1La méthode des k-means.....	16
2.2.3.2Approche floue du clustering.....	18
Méthode du Fuzzy C-means.....	18
Méthode du "subtractive clustering".....	20
Chapitre 3. Notre Approche One-class.....	23
3.1Schéma Global.....	23
3.2la sélection d'attributs.....	24
3.2.1Processus de sélection d'attributs.....	24
3.2.2Le Stemming.....	26
3.2.3Les mots vides et stop word.....	26
3.2.4La couverture.....	28
3.2.5Pré-traitement des données.....	29
3.2.5.1Seuil des valeurs.....	29
3.2.5.2Normalisation.....	29
3.3Clustering.....	30
3.3.1Distance entre point.....	30
3.3.2Optimisation du clustering.....	31
3.3.3Optimisation de la graine pour le tirage aléatoire.....	31
3.3.4Optimisation du nombre de clusters.....	34
3.4Seuillage.....	36
3.4.1Rayon max.....	36
3.4.2Analyse par dimensions.....	38
3.5Expérimentation.....	40
3.5.1Implémentation.....	40
3.5.1.1Weka.....	40
3.5.1.2Eclipse.....	40
3.5.1.3Codage.....	41
3.5.2Résultats.....	41
3.5.2.1Évaluation sur l'ensemble des Iris.....	41
3.5.2.2Résultats sur corpus d'Adamentium.....	42
Chapitre 4. Conclusion et perspectives.....	47
Bibliographie.....	48
Annexe.....	50

Introduction

Ce rapport présente les travaux réalisés durant mon stage de Master Recherche 2 au sein du Centre de Recherche LGI2P de l'Ecole des Mines d'Alès.

La thématique générale de mon sujet était le filtrage d'Internet pour la protection des mineurs. Plus précisément, l'objectif était de proposer une approche originale de filtrage dynamique par analyse du contenu textuel des pages web.

Actuellement la majorité des méthodes de filtrage sont basées sur des listes d'URL à interdire ou sur une classification supervisée (i.e. Les classes à séparer sont déterminées à l'avance). Nous avons travaillé sur une approche différente, basée sur des méthodes de clustering à une seule classe. L'intérêt majeur de cette approche consiste à simplifier la tâche de constitution des corpus d'apprentissage.

Ce document est organisé en trois parties. Dans la première, la problématique abordée durant le stage est détaillée. La seconde partie introduit les notions fondamentales d'une approche de classification à une seule classe. Enfin dans la troisième partie, nous décrivons l'approche proposée ainsi que les expérimentations réalisées et les résultats obtenus.

Chapitre 1. Problématique

La protection des mineurs sur Internet est un sujet social, politique et économique de premier ordre. Les récentes décisions gouvernementales, rendant obligatoire et gratuit le contrôle parental pour les fournisseurs d'accès à Internet (FAI), en attestent.

La solution de protection par filtrage des pages Internet aux contenus choquants ou illicites est aujourd'hui la plus répandue. Deux types technologies différentes sont aujourd'hui utilisées : les technologies basées sur des listes d'url (listes noires : url à interdire et listes blanches url à autoriser) et les technologies basées sur une analyse dynamique du contenu des pages (textes, images, vidéos, ...). C'est dans cette dernière catégorie, limitée à l'analyse des contenus textuels, que se situe le travail présenté dans ce rapport.

Le filtrage de l'Internet par analyse dynamique des contenus textuels peut être considéré comme une problématique de classification de textes. On distingue classiquement deux types de classification : la classification supervisée et la classification non supervisée.

- La classification supervisée, consiste à analyser de nouvelles données et à les affecter, en fonction de leurs caractéristiques ou attributs, à telle ou telle classe prédéfinie. Les algorithmes reposent sur des arbres de décision, des réseaux de neurones, la règle de Bayes, les k plus proches voisins ...
- La classification non supervisée (clustering, segmentation) ressemble à celle de la classification, mais diffère dans le sens où il n'existe pas de classes prédéfinies : l'objectif est de grouper des enregistrements qui semblent similaires dans une même classe. La problématique est de trouver des groupes homogènes dans une population. On utilise souvent les techniques d'agrégation autour de centres mobiles ou de classification ascendante hiérarchique. La difficulté essentielle dans ce type de construction est la validation.

La majorité des solutions de filtrage par analyse dynamique des contenus existantes utilisent des méthodes de classification supervisées.

Technologiquement, on peut également distinguer deux grandes catégories de filtre. Les solutions de filtrage personnelles installées directement sur les postes ayant accès à Internet et les solutions installées sur les ordinateurs distribuant l'accès à Internet (serveurs des FAI par exemple). C'est dans le cadre de cette dernière solution dite "cœur de serveur" que se situe notre étude. Comme nous le verrons par la suite, ce choix technologique aura un impact très fort sur les solutions de filtrage proposées.

La problématique abordée dans ce rapport peut se résumer ainsi :

" Est-il possible de proposer une solution cœur de serveur de filtrage de l'Internet efficace par analyse dynamique du contenu textuel des pages ?"

La notion d'efficacité soulevée dans cette question peut se décliner suivant plusieurs critères :

- Critère de rapidité

La solution technologique "cœur de serveur" choisie impose que le temps de traitement d'une page Internet ne soit pas trop important. Cette contrainte impactera de manière importante les prétraitements effectués sur les textes (analyse morphologique, syntaxique, sémantique, ...) ainsi que les stratégies de classification mises en œuvre.

- Critère de performance de classification

Classiquement les résultats d'une méthode de classification se présentent sous la forme d'une matrice de confusion :

$$\begin{array}{cc}
 & i & \bar{i} \\
 \hat{i} & VP_i & FP_i \\
 \hat{\bar{i}} & FN_i & VN_i
 \end{array}$$

○ $i, \bar{i}, \hat{i}, \hat{\bar{i}}$ représentent respectivement, les textes de classe i désignés par experts, les textes d'autres classes que i désignés par expert, les texte d e classe i désignés par le classifieur et les textes d'autres classes que i désignés par le classifieur.

○ VP_i l'ensemble des Vrais Positifs, i.e. l'ensemble des textes de

classe i assignés dans la classe i .

- VN_i l'ensemble des Vrais Négatifs, i.e. l'ensemble des textes n'appartenant pas à la classe i assignés dans d'autres classes que i .
- FP_i l'ensemble des Faux Positifs, i.e. l'ensemble des textes n'appartenant pas à la classe i assignés dans la classe i .
- FN_i l'ensemble des Faux Négatifs, i.e. l'ensemble des textes de classe i assignés dans une autre catégorie que i .

Dans le cadre du filtrage d'Internet la valeur de VP_i doit être la plus élevée possible tout en veillant à ce que la valeur de FP_i soit la plus petite possible. Ceci afin de garantir un filtrage efficace des sites à risque tout en garantissant une navigation sans blocage sur les autres sites.

- Critère de facilité de mise en oeuvre

Les méthodes de classification de textes permettent d'identifier des modèles (classifieurs) par apprentissage à partir de corpus. La plupart de ces méthodes étant de type supervisée, il est nécessaire de constituer des corpus indexés (i.e. chaque texte est affecté à une classe connue à l'avance). La création de ces corpus représente une phase longue et fastidieuse. Dans le cadre d'une classification supervisée pour la génération de filtre Internet il est nécessaire de construire des corpus pour toutes les thématiques que l'on souhaite éliminer mais également pour les thématiques que l'on souhaite laisser passer (classe " divers ") et ce pour chaque langue que l'on souhaite filtrer.

Par exemple identifier un classifieur pour filtrer des thématiques : pornographie, pédophilie, racisme et violence nécessitera la création de quatre corpus sur ces thématiques ainsi qu'un corpus sur la classe " divers ". Or cette classe " divers " est difficilement qualifiable, il est en effet impensable de constituer un corpus qui aborderait toutes les thématiques traitées sur Internet. Il existera donc toujours un biais dans la constitution de ce corpus.

D'autre part les méthodes de classification de textes sont généralement basées sur un codage statistique de la fréquence des mots apparaissant dans les textes [1]. Pour chacune des thématiques à filtrer un vocabulaire spécifique à la thématique sera rajouté, la taille des vecteurs codant les textes devenant alors très important.

Notre contribution exposée dans ce rapport, propose une alternative nouvelle au filtrage d'Internet par classifieur à 1-classe (one-class classifier) [2].

Cette méthode sera exposée en détail dans la partie XXXX. Le principe général consiste à utiliser une seule classe pour identifier des modèles de classification non supervisée. L'intérêt principal étant qu'il n'est plus nécessaire de constituer des corpus de la classe " divers " et donc faciliter la constitution des corpus et de diminuer la taille des vecteurs codant les textes.

Les travaux présentés dans ce rapport ont été réalisés en collaboration avec l'entreprise Adamentium dans le cadre d'un programme de recherche et développement réalisé en partenariat avec le Centre de Recherche LGI2P de l'Ecole des Mines d'Alès.

Adamentium est une jeune entreprise innovante spécialisée dans la protection des mineurs et la lutte contre la cybercriminalité. La société développe une nouvelle génération de logiciels de contrôle parental pour les Fournisseurs d'Accès Internet et pour les opérateurs de téléphonie mobile. Ses solutions de catégorisation de pages Web par analyse en temps réel permettent de bloquer les sites interdits aux mineurs, mais laissent passer les sites utiles à leur instruction.

Chapitre 2. One-Class

L'axe de recherche One-classe est relativement récent et la plupart des expérimentations menées à ce jour ont conduit à des résultats en deçà des espérances attendues.

Bien que cette solution permette d'obtenir des résultats relativement bon dans la catégorisation de la classe souhaitée, la catégorisation des outliers obtient souvent des résultats moins significatif, principalement en raison du choix de la fonction d'appartenance aux clusters générés.

La solution One-classe est en beaucoup de points semblable aux solutions de classification traditionnelles, à cela prêt qu'elle ne comporte qu'une classe, la classe recherchée.

Le problème dans la classification One-classe est de réaliser une description d'un jeu de données cible, et de détecter quels objets (nouveaux) ressemblent, ou non, à l'ensemble décrit. La différence avec la classification traditionnelle est que, dans l'optique One-classe, seulement une classe est présente. Tous les autres objets seront par définition considérés comme des outliers.

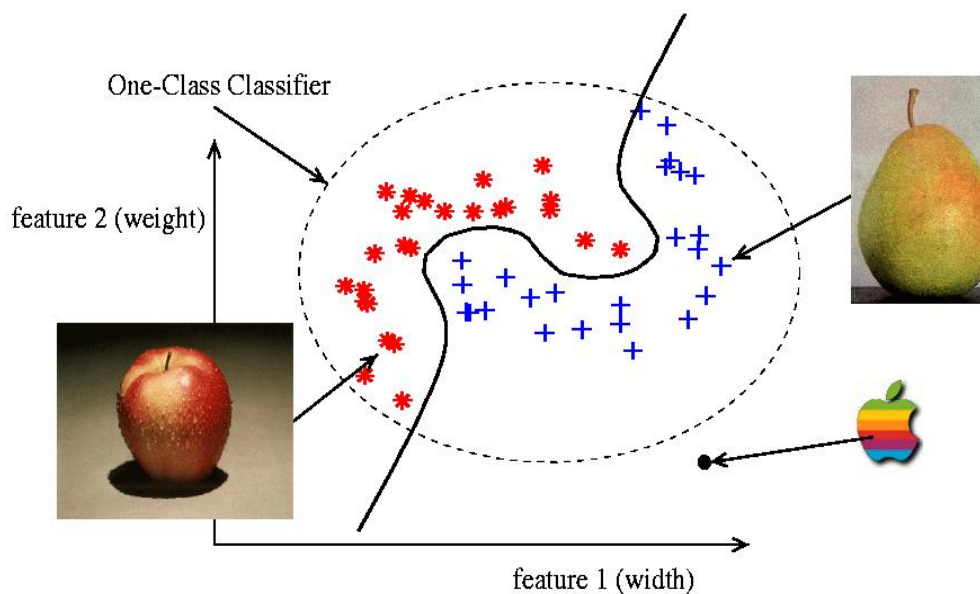


Fig 1. Une classification conventionnelle, et une One-classe, appliquée sur un exemple de jeu de données contenant la description, via deux caractéristiques, des pommes et des poires. La ligne continue représente la délimitation faite par le classifieur traditionnel, tandis que la ligne en pointillés représente le résultat d'une classification One-classe sur le même ensemble. On peut voir que la description faite par le classifieur One-classe peut identifier la fausse pomme comme un outlier tandis que le classifieur traditionnel la catégorise comme étant une poire.

Cette exemple illustre le fait qu'un objet qui n'apparaît pas dans le corpus¹ constitué, peut être totalement mal classé. Et du fait, tous les objets se rapprochant de celui ci risquent de subir le même traitement. Comme nous l'avons exposé précédemment, la classe représentant le « divers », n'est pas identifiable car elle représente un ensemble beaucoup trop vaste.

D'où l'idée de ce séparer de la classe du divers afin de se concentrer uniquement sur la classe catégorisable.

Le choix de cette classification nécessite un traitement particulier composé d'un pré-traitement des données adaptés ainsi qu'un algorithme de classification pouvant faire abstraction de la notion de classe tel que le clustering.

Il n'existe pas d'algorithme à proprement parlé One-classe. Le One-classe apparaît comme une solution nécessitant un cheminement particulier, que tout un chacun se doit d'adapter à sa problématique. On peut identifier trois grandes étapes nécessaires à la génération d'un modèle One-classe. La sélection d'attributs qui doit conduire à réduire l'espace (nombre de mots). Le clustering qui va générer un modèle représentant la classe étudiée. Et pour finir la fonction d'appartenance, qui permettra de déterminer si un nouvel objet (site Web dans notre cas) appartient ou non à la classe recherchée.

¹ Un corpus est un ensemble de documents, artistiques ou non (textes, images, vidéos, etc.), regroupés dans une optique précise. On peut utiliser des corpus dans plusieurs domaines : études littéraires, linguistiques, scientifiques, etc.
Dans ce cas il correspond à la description des pommes et des poires.

2.1 sélection d'attributs (feature selection)

2.1.1 Définition

Ce que l'on appelle « Feature Selection », comprenez sélection d'attributs, est une étape cruciale dans la réalisation d'une classification efficace. On peut considérer que c'est cette étape qui va définir si le modèle de classification final aura la capacité d'obtenir de bons résultats ou pas.

La sélection d'attributs a pour but de supprimer l'information redondante ou non pertinente qui n'apporterait aucun bénéfice à la classification des données et rendrait celle-ci plus difficile.

Ce pré-traitement des données (réduction de corpus) permet d'améliorer la précision des données, d'opérer plus rapidement et de façon plus efficace, d'obtenir des résultats plus compacts, et d'interpréter plus facilement les résultats.

La plus part des algorithmes de sélection d'attributs (features Selection) se basent sur l'appartenance à des classes (sous ensembles caractéristiques) afin de déterminer les attributs pertinents pour la classification.

On peut citer pour les plus connus:

- Information Gain Attribute Ranking
- Relief
- Principal components
- CFS (Correlation-based Features Selection)
- Wrapper Subset Evaluation
-

Ces algorithmes ne fonctionnent pas tous de la même façon. On peut catégoriser deux grandes classes d'algorithmes: les filtres, et les wrapper.

Les « Wrappers » évaluent les attributs en fonction de l'algorithme d'apprentissage utilisé par la suite.

Les « Filtres » eux utilisent des caractéristiques générales et opèrent donc de façon totalement indépendante.

On peut aussi introduire une autre taxonomie sur les algorithmes:

- Ceux qui évaluent les attributs individuellement
- Ceux qui utilisent les groupes d'attributs

Les articles [3][4][5] proposent des benchmark intéressants pour des algorithmes de sélections d'attributs. Certains de ces algorithmes n'opèrent que sur des valeurs d'attributs discrètes². Pour appliquer ces techniques aux attributs à valeur continus³, on utilise la discrétisation. Cette discrétisation peut s'effectuer de différente façon comme le propose Fayyad et Irani[4].

2.1.2 problématique apporté par le One-class

L'hypothèse d'une classification One classe engendre des difficultés non négligeables. La sélection des attributs se trouve grandement complexifiée par l'utilisation de corpus One classe.

La plus part de ces algorithmes s'appuient sur la présence de diverses classes afin de retenir les attributs pertinents pour chacune d'entre elles. Or par définition le One-class n'en comporte qu'une.

L'une des règles les plus appliquée, mais également la plus efficace utilisée par ces algorithmes consiste à repérer les mots qui ne sont pas relevant pour une classe. Généralement ce traitement est réalisé via l'utilisation de l'algorithme de TF/IDF. Ceci consiste à éliminer tous les mots qui n'ont pas de rapport directe avec une classe.

Dans une optique multiClasse on peut opposer les différentes classes afin de déduire les attributs pertinents représentant chacune des catégories, tout en éliminant ceux totalement inutiles. Pour ce faire les algorithmes analysent tout simplement le nombre de fois où les mots apparaissent dans chaque classe, et comparent ces résultats entre eux.

Si un mot se révèle comme apparaissent un grand nombre de fois dans une classe, on pourrait penser qu'il peut être considéré comme un mot caractérisant cette classe. Or si de la même manière ce mot apparaît fréquemment dans d'autres classes on peut alors penser que ce mot n'est plus du tout relevant. Il ne permet dès lors plus de déterminer une classe particulière. Parmi ces mots on peut considérer tous les mots d'usage courant: déterminant, pronom ...

2 Valeur discrète: Qui a un espace de valeur limité. (ex classe{Pornographique, Adulte, Adolescence, Tout public})

3 Valeur continus: Peut prendre n'importe quel valeur.

Prenons un exemple:

Dans le cas de la pornographie, on s'aperçoit que le mot sexe va revenir dans la plus part des sites et, du fait, doit être considéré comme un mot (attribut) représentatif de cette catégorie et donc conservé pour la classification.

Cependant en appliquant le même raisonnement on va trouver le même résultat pour le mot « le » (déterminant de la langue française).

Or on conviendra que celui ci n'est absolument pas représentatif pour la catégorie pornographie.

L'utilisation de plusieurs classes (catégories), permettrai de repérer que ce mot « le », apparaît également dans les autres catégories (le divers), ce qui nous laisserai penser que ce mot n'est pas distinctif d'une catégorie particulière, mais plutôt un mot d'usage courant de la langue étudiée.

L'utilisation d'un corpus composé d'une seul classe, ne nous permet pas de repérer ce problème. Il a donc fallut trouver des solutions nous permettant d'éliminer ces mots « d'usage courant ». Nous avons pour cela étudié diverses possibilités pour palier à ce problème.

Un autre problème, cette fois caractéristique des corpus constitués de pages Web, est introduit par Wai-chiu Wong et Ada Wai-chee Fu [7] et M. Beigbeder et A. Mercier[8]. Leurs travaux révèlent une particularité des pages Web gênant l'utilisation d'algorithmes de sélection d'attributs traditionnels.

Ils exposent le fait que les pages Web contiennent généralement un petit nombre de mots et la plupart de ces mots n'apparaissent qu'une ou deux fois, tandis qu'a contrario certains termes spécialement employés à but de recensement apparaissent un nombre de fois très importantes.

2.2 Techniques de classification non supervisée

2.2.1 Principes de base

La classification non supervisée (*clustering en anglais*) consiste à segmenter un ensemble de vecteurs non labellisés (la classe n'est pas fournie) en groupes (*clusters*) qui possèdent les propriétés suivantes :

- *Homogénéité* dans les groupes, i.e. les données appartenant à un même cluster doivent être les plus similaires possibles.
- *Hétérogénéité* entre groupe, i.e. les données appartenant à différents clusters doivent être les plus dissemblables possibles.

Les notions de similarité / "dissimilarité" seront abordées dans le paragraphe suivant.

Un exemple illustratif d'une classification non supervisée est donné sur la figure suivante. Les vecteurs d'entrée sont décrits sur la figure (a) et les clusters désirés sur la figure (b).

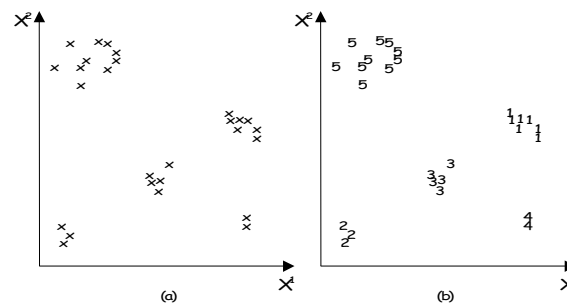


Fig 2.Principes du Clustering

Il existe plusieurs familles d'algorithmes de classification non supervisée : les algorithmes conduisant directement à des partitions comme les méthodes d'agrégation autour de centres mobiles; les algorithmes ascendants (ou encore agglomératifs) qui procèdent à la construction des classes par agglomérations successives des objets deux à deux, et qui fournissent une hiérarchie de partitions des objets; enfin les algorithmes descendants (ou encore divisifs) qui procèdent par dichotomies successives de l'ensemble des objets, et qui peuvent encore fournir une hiérarchie de partitions. On se limitera ici aux deux premières techniques de classification non supervisée:

- les groupements peuvent se faire par recherche directe d'une partition, en affectant les éléments à des centres provisoires de classes, puis en recentrant ces classes, et en affectant de façon itérative ces éléments. Il s'agit des techniques d'agrégation autour de centres mobiles, apparentées à la méthode des "nuées dynamiques", ou méthode "k-means", qui sont particulièrement intéressantes dans le cas des grands tableaux
- les groupements peuvent se faire par agglomération progressive des

éléments deux à deux. C'est le cas de la classification ascendante hiérarchique qui est présentée ici suivant plusieurs critères d'agrégation. Nous envisagerons d'une part la technique "du saut minimal" (single linkage) équivalente, d'un certain point de vue, à la recherche de l'arbre de longueur minimale, et d'autre part la technique d'agrégation "selon la variance", intéressante par la compatibilité de ses résultats avec certaines analyses factorielles.

Ces techniques présentent des avantages différents et peuvent être utilisées conjointement. Il est ainsi possible d'envisager une stratégie de classification basée sur un algorithme mixte, particulièrement adapté au partitionnement d'ensembles de données comprenant des milliers d'individus à classer.

2.2.2 Mesures de similarité

Toutes les techniques de clustering que nous allons étudier font référence à la notion de mesure de similarité entre deux vecteurs. La diversité des types et des échelles des dimensions des vecteurs donne une importance fondamentale à la mesure de similarité utilisée.

Les mesures de similarité les plus utilisées sont les mesures de distance. Nous allons plus particulièrement nous intéresser aux mesures de distance pour des champs continus. Pour ce type de données, la distance euclidienne reste la plus utilisée.

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^d (\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^2 \right)^{1/2} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$$

avec $\mathbf{x}_i = (\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,d})$ et $\mathbf{x}_j = (\mathbf{x}_{j,1}, \dots, \mathbf{x}_{j,d})$ deux vecteurs de dimension d .

Cette distance est un cas particulier ($p=2$) de la métrique de Minkowski.

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^d (\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^p \right)^{1/p} = \|\mathbf{x}_i - \mathbf{x}_j\|_p$$

Ces mesures posent un problème lorsque les échelles des données ne sont pas homogènes. En effet les grandes échelles sont favorisées dans le calcul de la

distance.

Pour palier le problème, les données peuvent être normalisées (part rapport à la variance ou à une plage fixée).

2.2.3 Quelques méthode de clustering

2.2.3.1 La méthode des k-means

Bien qu'elle ne fasse appel qu'à un formalisme limité et que son efficacité soit dans une large mesure attestée par les seuls résultats expérimentaux, la méthode des k-means est probablement la technique de partitionnement la mieux adaptée actuellement aux vastes recueils de données ainsi que la plus utilisée pour ce type d'application.

Description de la méthode :

- On considère l'espace de n points de dimension p suivant :

$$X : \begin{bmatrix} x_1^1 & \dots & x_1^j & \dots & x_1^p \\ \dots & \dots & \dots & \dots & \dots \\ x_i^1 & \dots & x_i^j & \dots & x_i^p \\ \dots & \dots & \dots & \dots & \dots \\ x_n^1 & \dots & x_n^j & \dots & x_n^p \end{bmatrix}$$

- On suppose que les n points peuvent être groupés en c clusters $c < n$
- Les clusters sont décrits par leurs centres :

$$V_k = [v_k^1, v_k^2, \dots, v_k^j, \dots, v_k^p], \quad 1 \leq k \leq c$$

- On note $d(i,k)$ la distance entre le point x_i et le centre V_k
- Le point X_i est affecté au cluster dont le centre est le plus proche (au sens de d)
- On note m_k la moyenne des vecteurs dans le cluster k

Algorithme :

Initialiser la position des centres :

$$V_k = [v_k^1, v_k^2, \dots, v_k^j, \dots, v_k^p], 1 \leq k \leq c$$

Calculer les m_k Jusqu'à ce qu'il n'y ait plus de changement sur les m_k FaireChaque point X_i est affecté au cluster le plus procheCalculer les nouveaux m_k

Fin Jusqu'à

La figure suivante illustre le déroulement de l'algorithme pour :

$$P=2$$

$$c=4$$

$$V_k = [0, 0], 1 \leq k \leq 4$$

Les cercles rouges représentent les positions successives des centres des 4 clusters.

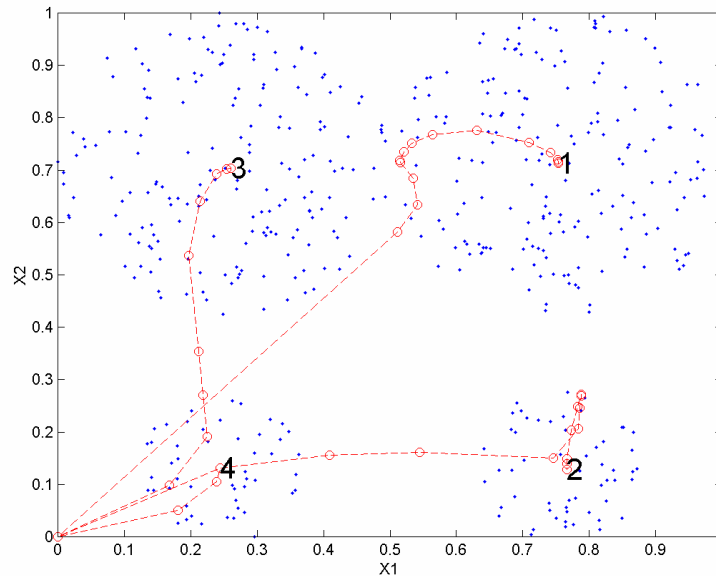


Fig 3. Exemple d'application du kmeans

2.2.3.2 Approche floue du clustering

Les sous-ensembles flous permettent une représentation simple des incertitudes et imprécisions liées aux informations et aux connaissances. Son principal avantage est d'introduire le concept d'appartenance graduelle à un ensemble alors qu'en logique ensembliste classique cette appartenance est binaire (appartient ou n'appartient pas à un ensemble).

Méthode du Fuzzy C-means

En 1981, Besdek a proposé une version floue du K-means : l'algorithme du Fuzzy C-means (FCM) [9].

Description de la méthode :

- On considère l'espace de n points de dimension p suivant :

$$X = \begin{bmatrix} x_1^1 & \dots & x_1^j & \dots & x_1^p \\ \dots & \dots & \dots & \dots & \dots \\ x_i^1 & \dots & x_i^j & \dots & x_i^p \\ \dots & \dots & \dots & \dots & \dots \\ x_n^1 & \dots & x_n^j & \dots & x_n^p \end{bmatrix}$$

- On suppose que les n points peuvent être groupés en c clusters $c < n$
- Les clusters sont décrits par leurs centres :

$$V_i = [v_i^1, v_i^2, \dots, v_i^j, \dots, v_i^p], \quad 1 \leq i \leq c$$

- On considère la matrice de proximité suivante :

$$U = \begin{bmatrix} u_{11} & \dots & u_{1k} & \dots & u_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ u_{i1} & \dots & u_{ik} & \dots & u_{in} \\ \dots & \dots & \dots & \dots & \dots \\ u_{c1} & \dots & u_{ck} & \dots & u_{cn} \end{bmatrix}, \text{ avec } k = 1, \dots, n \text{ et } i = 1, \dots, c$$

u_{ik} représente le degré d'appartenance du point X_k au centre V_i

$$u_{ik} = \left[\sum_{j=1}^c \left(\frac{d_{ik}}{d_{jk}} \right)^{\frac{2}{m-1}} \right]^{-1}, \text{ } d_{ik} \text{ représente la distance entre } V_i \text{ et } X_k$$

$$u_{ik} = \left[\sum_{j=1}^c \left(\frac{\|X_k - V_i\|}{\|X_k - V_j\|} \right)^{\frac{2}{m-1}} \right]^{-1}, \text{ si on choisit la distance euclidienne}$$

Algorithme :

1) Initialiser la position des centres :

$$V_i = [v_i^1, v_i^2, \dots, v_i^j, \dots, v_i^p], \quad 1 \leq i \leq c$$

$l=0$, initialiser la matrice : $U^{(l)}$

2) Calculer la position des centres

$$V_i = \frac{\sum_{k=1}^n (u_{ik})^m x_k}{\sum_{k=1}^n (u_{ik})^m}, \quad 1 \leq i \leq c$$

3) Calculer la matrice : $U^{(l+1)}$

4) Si $\|U^{(l+1)} - U^{(l)}\| < \epsilon$

Arrêter l'algorithme

Sinon

$l = l + 1$ et retourner en 2)

La figure suivante illustre l'application de cette méthode sur un nuage de points en deux dimensions (X, Y). A l'état initial les trois centres sont positionnés au centre du nuage. Puis après quelques itérations, ils convergent vers trois positions d'équilibre qui sont les barycentres des trois clusters constituant le nuage de points.

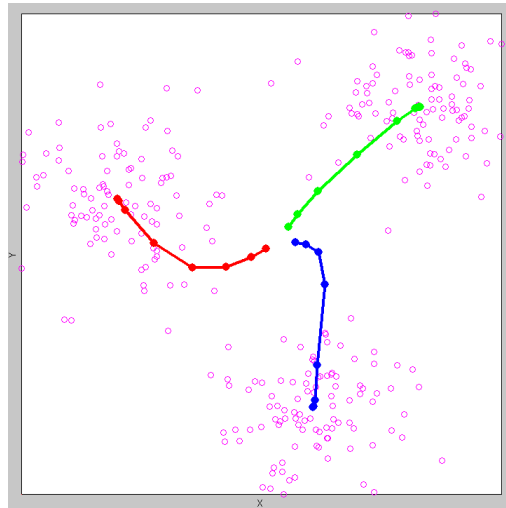


Fig 4.Exemple d'application du Fuzzy c-means

Méthode du "subtractive clustering"

L'inconvénient majeur de l'algorithme du FCM est que l'on doit connaître a priori le nombre et la position des centres des clusters. La méthode du "subtractive clustering" [10] permet d'estimer et de positionner automatiquement les clusters initiaux. C'est une méthode itérative basée sur une pondération des points à traiter. Plus un point aura de voisins proches, plus son potentiel sera élevé. A partir du calcul initial des différents potentiels, l'algorithme permet de sélectionner les meilleurs représentants du nuage de points et de définir ainsi les clusters principaux.

La figure suivante illustre le principe de formation des clusters pour un espace trois dimensions (Y1, Y2, Z). La figure a décrit le nuage de points initial. Les figures suivantes illustrent l'évolution des potentiels des points, le choix des centres et l'affectation des fonctions d'appartenance aux centres.

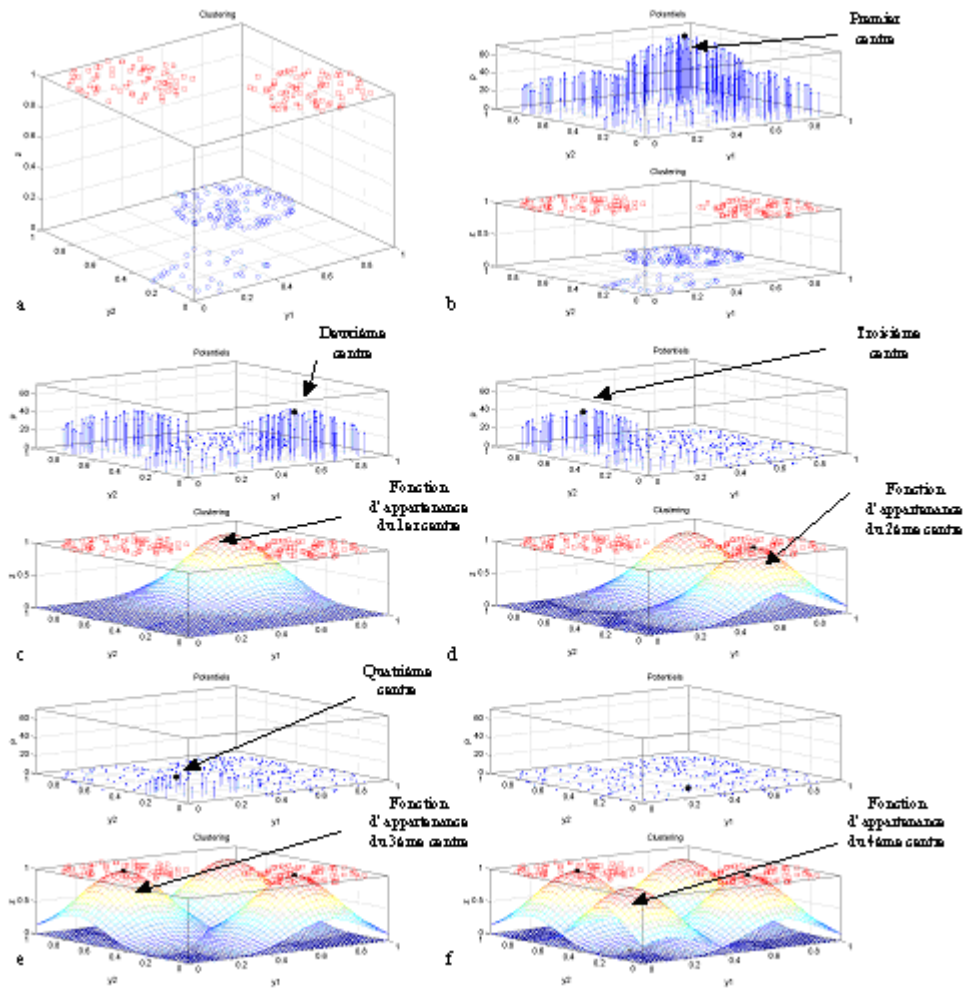


Fig 5. Visualisation d'une classification de type subtractive clustering

On considère l'espace de n points de dimension p suivant :

$$X : \begin{bmatrix} x_1^1 & \dots & x_1^j & \dots & x_1^p \\ \dots & \dots & \dots & \dots & \dots \\ x_i^1 & \dots & x_i^j & \dots & x_i^p \\ \dots & \dots & \dots & \dots & \dots \\ x_n^1 & \dots & x_n^j & \dots & x_n^p \end{bmatrix}$$

 Algorithme :

1) $k=0$

$$P_i = \sum_{j=1}^n e^{-\alpha \|x_i - x_j\|^2}, \alpha = \frac{4}{r_a^2}$$

2) $k=k+1$ Choisir le point avec le P_i le plus grand

Ce point devient un centre :

Coordonnées : x_k^ , et de potentiel : P_k^** 3) Modifier les P_i :

$$P_i = P_i - P_k^* e^{-\beta \|x_i - x_k^*\|^2}, \beta = \frac{4}{r_b^2}$$

4) Répéter 2 et 3 jusqu'à :

$$P_k^* < \varepsilon P_1^*$$

A chaque cluster i identifié par la méthode précédente est associé une fonction d'appartenance de la forme :

$$\mu_i(X) = e^{-\alpha \|X - X_i^*\|^2}$$

Chapitre 3. Notre Approche One-class

3.1 Schéma Global

Pour réaliser cette classification One-class nous avons suivi un processus typique de classification consistant à effectuer une sélection des attributs afin de réduire l'espace, suivi d'un algorithme de Clustering de type Kmeans. Nous obtenons ainsi un modèle qui nous permet d'évaluer les nouveaux sites Web.

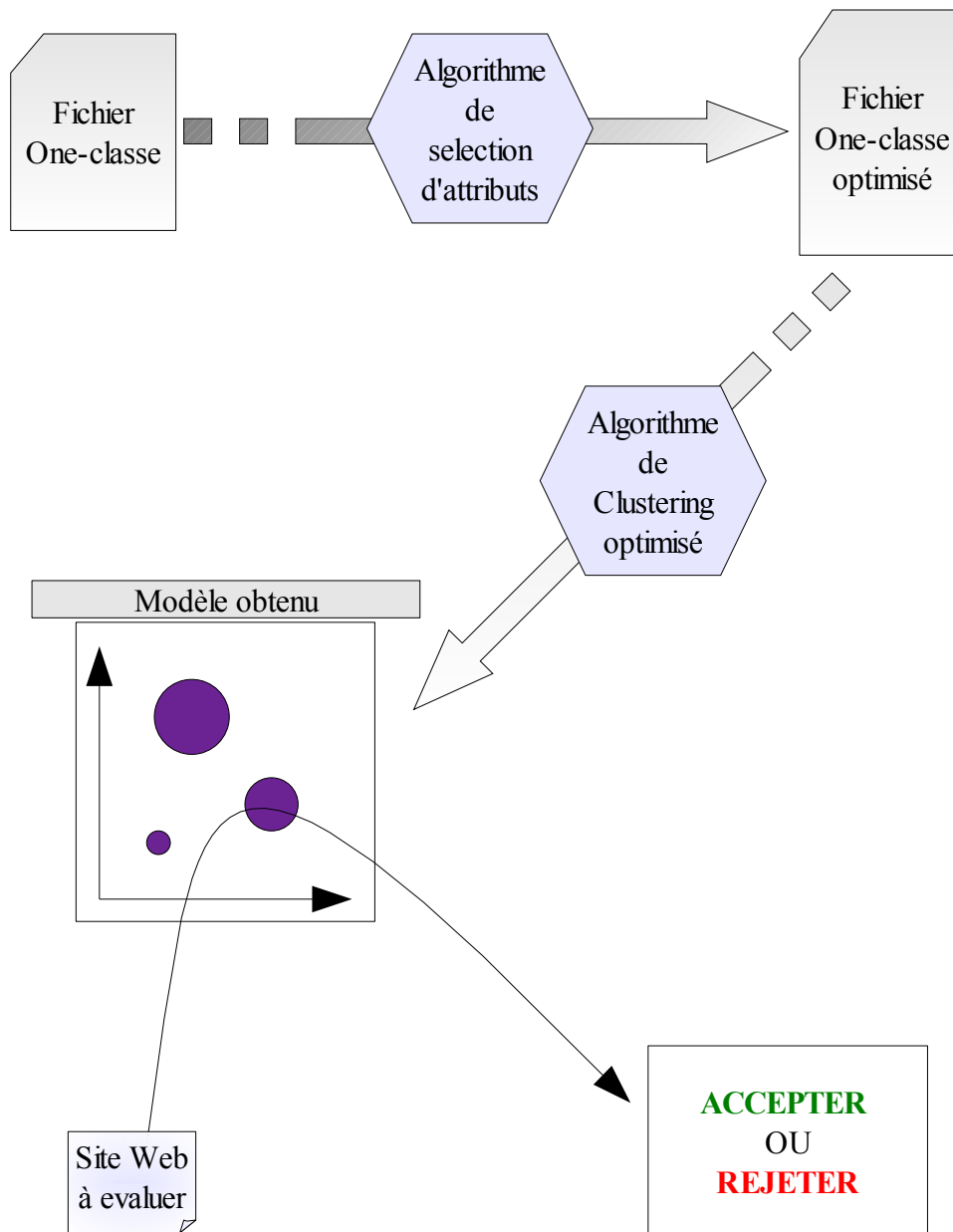


Fig 5. schéma de notre processus de classification

Chaque étape de ce processus a été adapté afin de convenir à une classification de type One-class.

3.2 la sélection d'attributs

Afin de réduire au maximum la taille de l'ensemble des termes du corpus, nous avons mis en place une succession de traitements organisés d'une façon bien particulière. Ce processus ayant pour but de sélectionner les termes n'ayant un rapport qu'avec la classe étudiée.

Rappelons que la réussite d'un clustering, et donc l'efficacité du classifieur, obtenu en sortie dépend principalement de cette aptitude à réduire l'espace en ne conservant que les termes les plus pertinents pour le domaine étudié.

3.2.1 Processus de sélection d'attributs

Notre processus intègre différents traitements permettant de déjouer les problèmes engendrés par la solution One-Classe.

Nous appliquons tout d'abord un algorithme de Stemming permettant de regrouper les mots ayant la même racine. Ensuite nous retirons à notre ensemble tous les mots apparaissant dans une liste de mots dits fréquents.

Une fois ces deux opérations effectuées nous nous retrouvons avec un ensemble de termes sensés appartenir uniquement au domaine représenté par la classe.

Nous appliquons donc un algorithme de couverture qui va nous permettre de sélectionner les termes les plus importants qui suffisent à couvrir notre espace.

Une fois cette opération effectuée nous nous retrouvons avec notre ensemble final de termes.

Nous procédons ensuite à une modification des valeurs des termes afin d'optimiser l'espace final.

Pour cela nous supprimons l'ensemble des sites éventuellement non couverts (selon taux de couverture utilisé durant l'étape de même nom) par l'ensemble des termes finaux.

Nous effectuons ensuite un seuillage des valeurs, afin de concentrer l'espace.

Et pour finir une normalisation de l'ensemble des valeurs des attributs

pour obtenir un espace uniforme entre [0,1].

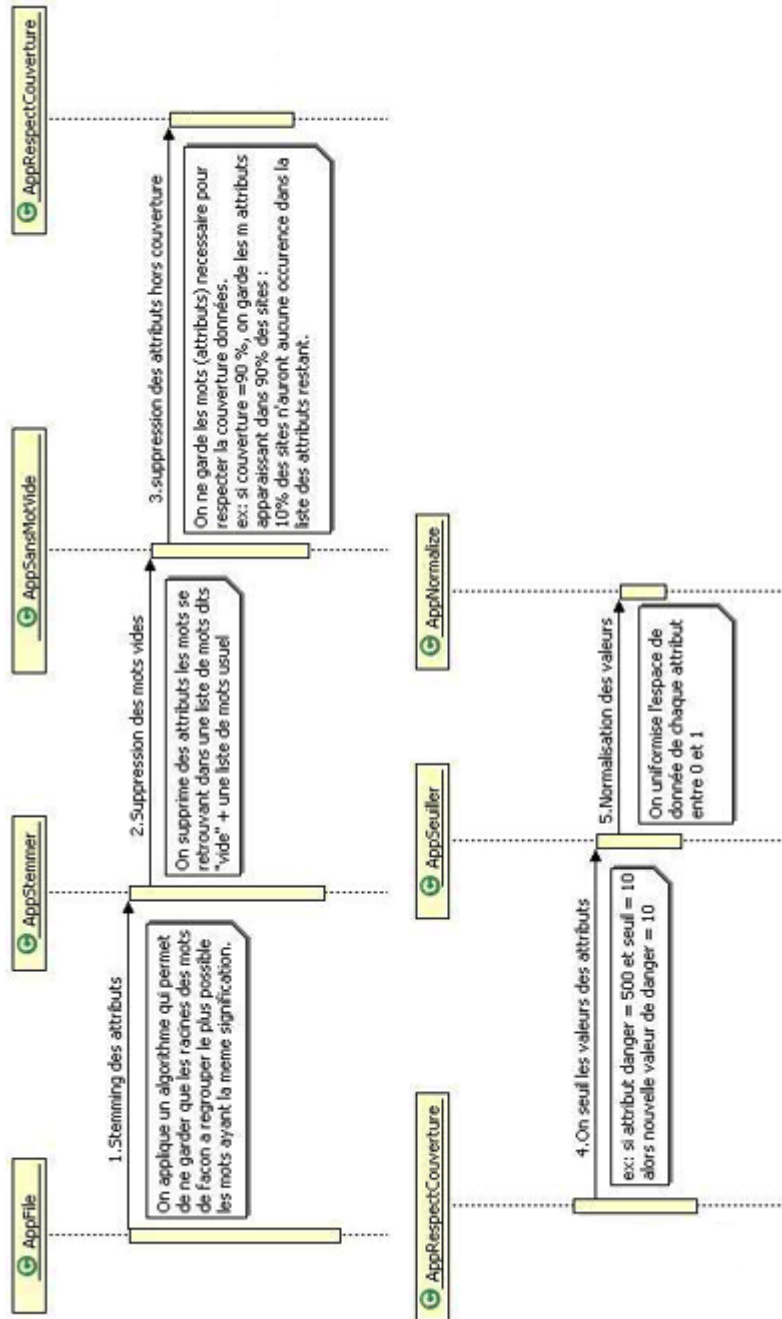


Fig 6. diagrammes de séquence du processus de sélection d'attribut

3.2.2 Le Stemming

Comme nous venons de l'expliquer un des enjeux principaux est de réduire l'espace. Pour ce faire, il existe de nombreux algorithmes allant du très simple au plus complexe permettant d'effectuer des traitements sur le texte (propre à un langage) afin d'optimiser au maximum le corpus. Comme nous l'avons exposé au début du rapport, une des contraintes commerciales liées à notre étude est le temps de calcul nécessaire à la classification d'un site.

Ce temps de réponse nous impose le choix d'un algorithme très peu coûteux aussi bien en temps qu'en puissance de calcul.

Conscient de ces contraintes nous nous sommes tournés vers le Stemming comme premier traitement visant à réduire l'espace.

Le Stemming est un algorithme très simple s'appuyant sur la construction de la langue étudiée, et visant à réduire les mots à leurs racines la plus simple. On trouve un bon nombre de ces algorithmes sur internet allant du plus simple au plus abouti. Ces algorithmes se trouvent aisément pour les langues les plus manipulées.

Notre choix c'est porté sur l'algorithme présent dans la solution de recherche de texte Lucene [11], développé par la fondation Apache.

Exemple de résultat d'un stemming sur la langue anglaise:

managing, manage, managed, manager, managers = manage

3.2.3 Les mots vides et stop word

les mots vides représentent les mots très fréquemment utilisés dans le langage courant.

Les stop word eux représentent les mots utilisés pour la construction d'une phrase.

Dans l'avenir nous ne ferons plus la distinction entre ces deux ensembles, les considérant comme une seule et unique ressource que nous nommerons « mot Vide ».

Comme nous l'avons exposé précédemment l'utilisation d'une seule classe pose des problèmes quand à l'identification des mots d'usage courant du langage.

Dans une optique multi-classes l'algorithme TF-IDF (cf annexe), permet de mettre en évidence les termes qui apparaissent fréquemment au sein d'une

classe, mais qui n'apparaissent presque pas dans l'ensemble des autres classes.

Cet algorithme est particulièrement efficace pour faire une distinction entre les termes pertinents permettant de catégoriser une classe et les termes qui se révèlent comme communs pour l'ensemble des sites.

Ces mots sont, en effet, considérés comme non intéressants puisque le but est d'obtenir le vocabulaire propre à chaque classe. Or un mot qui a un faible TF-IDF, correspond à un mot que l'on retrouve dans l'ensemble des sites et donc des classes.

Cet algorithme est particulièrement efficace puisqu'il permet très rapidement d'éliminer un grand nombre de termes non intéressants pour la classification. Ces termes correspondant pour la plupart aux termes fréquemment utilisés dans une langue pour construire une conversation.

Dans une optique One-classe on ne peut pas appliquer le principe de l'algorithme TF-IDF puisque la classe et le document représentent le même ensemble. Si l'on prend l'exemple de la catégorie pornographie, le mot "sexe" apparaît dans pratiquement tous les sites. Ce qui implique qu'il apparaît dans toute la classe mais également dans tout le document puisque dans notre cas le document n'est composé que de la classe. Il est donc intéressant de le conserver.

Pour solutionner ce problème, nous avons opté pour une solution utilisée principalement dans la recherche de document.

On trouve sur internet, pour les langues les plus parlées, une liste des mots dits « vide ou fréquent » par langue. Cette liste représente un à plusieurs milliers de mots selon les langues concernées, et est composée en générale d'articles, de déterminants, de verbes de liaison, de verbes/mots/adjectifs très courant, de chiffres, ect ...

La encore, nous avons opté pour les ressources présentes dans l'application Lucene.

Cette solution nous permet de palier au problème évoqué par le choix one classe qui reste facilement traitable en solution multi-classe.

3.2.4 La couverture

La couverture permet de réduire de façon très importante le nombre d'attributs nécessaires à la catégorisation de la classe. Le principe est de ne retenir que les n attributs permettant de couvrir l'ensemble des sites Web d'apprentissage.

Nous avons choisi de mettre au point une fonction nous permettant de définir un seuil est de calculer en fonction les attributs nécessaires à la couverture de ce seuil.

Pour schématiser si l'on souhaite avoir une couverture égale à 90 %, on ne garde que les m attributs apparaissant dans 90% des sites. Au final nous aurons donc 10% des sites qui n'auront aucune occurrence dans la liste des attributs restants.

Algorithme:

Data: matrice, seuil

return: listeAttRetenu

```
for( i=0 ; i< matrice.listeAtt.size ; i++ )
    listCouvertureAtt(i)= df( matrice.att(i) );
listBestAtt = inversOrder( listCouvertureAtt );
j=0;
while ( couv < seuil )
    couv += df( matrice.att( listBestAtt(j) ) )
    matrice.listeSite -= listeSiteWhereAtt ( listBestAtt(j) ) > 0 ;
    listeAttRetenu.add( listeBestAtt(j) );
    j++;
return listeAttRetenu;
```

Au terme de cette opération, nous obtenons un certains nombre de sites non couverts par le seuil défini à la base, qui se retrouvent avec aucune occurrence pour l'ensemble des attributs retenus. De ce fait, nous ne pouvons plus considérer ces sites comme relevant pour la génération du modèle de classification. Nous supprimons donc ces sites de notre corpus.

Nous verrons dans les perspectives comment nous pouvons néanmoins trouver une utilité à ces sites.

3.2.5 Pré-traitement des données

Maintenant que nous nous sommes assurés d'avoir gardé seulement les attributs les plus représentatifs de notre classe, nous allons traiter les valeurs de ces attributs.

Comme nous l'avons présenté précédemment les sites Web ont la particularité de ne pas comporter beaucoup de terme. De plus on sait que ces termes se retrouvent en très faible occurrences à l'exception des techniques de répétition de mots utilisées afin d'augmenter l'attention de l'utilisateur ou encore le référencement par certains moteurs de recherche.

Le traitement de site Web ne peut de ce fait pas s'aborder comme le traitement de texte classique.

Dans le but de rendre les données le plus exploitable possible par l'algorithme de clustering, il est important d'essayer de rendre l'espace des valeurs le plus homogène possible. Pour cela nous avons effectué différents traitements tendant à rendre cohérentes les valeurs par site et par attribut.

3.2.5.1 Seuil des valeurs

En analysant les valeurs des attributs nous avons constaté que dans la majorité des cas seuls certains sites avaient de fortes valeurs, tandis que la majorité des valeurs se concentraient dans un espace très faible. Nous avons donc un écart type très faible situé aux alentours des premières valeurs.

On peut expliquer ceci par l'utilisation de répétitions de mots cachés permettant un meilleur référencement dans certains moteur de recherche.

Laisser l'ensemble tel quel est préjudiciable pour un bon clustering.

Pour palier ce problème d'ensembles démesurés, nous avons introduit une notion de seuil ayant pour objectif de concentrer les valeurs sur un ensemble beaucoup plus restreint.

3.2.5.2 Normalisation

Pour finir nous nous sommes intéressés aux différentes valeurs non pas pour un même site, mais aux différentes valeur au sein des attributs. Nous avons pour cela développé un algorithme de normalisation tendant à

repositionner l'ensemble des valeurs de chaque attribut dans un ensemble [0;1].

Ceci nous permettra d'avoir un ensemble où toutes les dimensions seront égales.

3.3 Clustering

Pour mener à bien notre expérimentation nous avons choisi d'utiliser dans un premier temps le clustering de type SimpleKmeans présent sous le logiciel Weka. Ce clustering représente le double avantage d'être éprouvé comme un clustering fiable est rapide.

Ceci étant ce clustering comporte une zone de flou quand à la détermination du nombre de clusters nécessaires ainsi que la « graine »⁴ à utiliser.

Afin d'optimiser notre clustering nous avons joué sur deux paramètres en les faisant varier jusqu'à obtenir un résultat que nous qualifierons d'optimal pour notre ensemble.

Afin de déterminer notre optimal nous utilisons une notion de distance entre point, qui nous permet en quelque sorte d'évaluer la densité des clusters.

3.3.1 Distance entre point

On définit la distance entre deux points comme la longueur de la droite les reliant. Nous avons choisi la distance euclidienne pour représenter cette distance.

Elle s'applique facilement à des espaces multidimensionnels puisqu'elle correspond à la racine carrée des sommes des distances pour chaque dimension des deux points.

Soit X et Y deux points de l'espace:

$$\sqrt{\sum_{i=1}^{nbDimension} (X_i - Y_i)^2}$$

Nous utiliserons par la suite cette distance pour évaluer notre espace contenant les clusters.

4 La graine correspond à un tirage aléatoire pour déterminer la position initiale des clusters.

3.3.2 Optimisation du clustering

Pour nous permettre d'évaluer notre clustering nous avons considéré les éléments suivant au sein d'un cluster: le centre du cluster et son rayon.

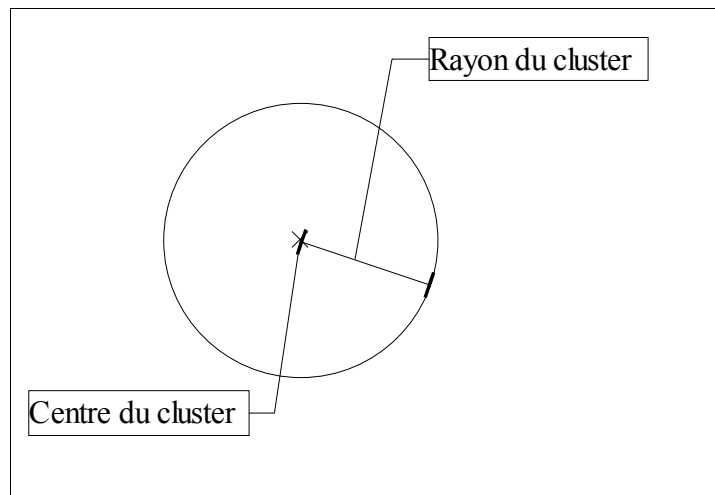


Fig 7. Représentation d'un cluster.

Le rayon du cluster correspond à la distance la plus importante entre une des instances composant le cluster et son centre, comprenez l'instance la plus éloigné du centre du clusters.

Le centre du cluster quand à lui est représenté par les coordonnées dans l'espace, du centre du cluster.

3.3.3 Optimisation de la graine pour le tirage aléatoire

Le clustering présent sous Weka comporte une notion de graine. Cette graine correspond à un point de départ aléatoire dans l'espace pour calculer les différents clusters.

Selon la graine de départ choisi les points représentant les clusters vont être placés aléatoirement sur l'espace est c'est à partir de cet emplacement que les clusters vont évoluer jusqu'à obtenir un état de stabilité qui conduira à la fin du clustering.

Nous avons constaté par des expérimentations simples que la variation de cette Seed générant des clustering différents, et ce de manière relativement

conséquente. Notre intérêt est d'obtenir des clusters les plus compacts possibles, c'est à dire des clusters de petite taille qui concentrent un maximum de points (site Web).

En effet plus ces clusters sont importants en taille, plus l'espace qu'ils occupent sur notre ensemble est important. Ce qui conduit à avoir un grand nombre de faux positifs. Chose que logiquement nous cherchons à éviter au maximum.

Analysons ce problème sur un exemple concret.

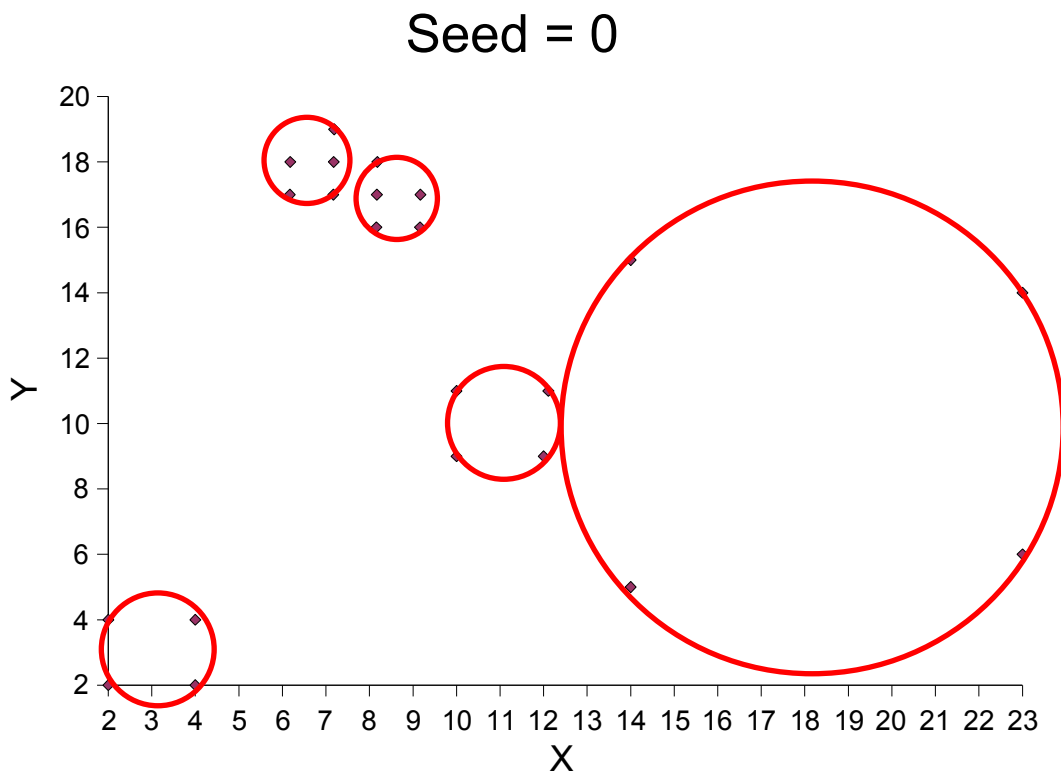


Fig 8. clustering effectué sur un ensemble de test pour une Seed = 0

On observe sur ce diagramme que le cluster de droite est beaucoup trop important par rapport aux 4 points qu'il comporte. Il est donc primordial de décomposer ce cluster qui n'est pas représentatif de notre espace.

Afin d'optimiser ce paramètre, nous avons mis au point une solution s'appuyant sur le simpleKmeans. Cette méthode, effectuée le clustering plusieurs fois en faisant varier la Seed entre 0 et 10.

Au final elle retourne le clustering et la Seed utilisée qui d'après nos calculs représente le clustering optimal.

Cette notion d'optimal est calculé en s'appuyant sur les distances euclidiennes entre points.

Nous additionnons simplement toutes les distances euclidiennes entre chaque point et le centroïde du cluster auquel ils ont été affectés par le simpleKmeans. Ce qui équivaut en quelque sorte à la somme des densités des différents clusters.

Au final plus la somme obtenue est faible, plus les clusters calculés sont denses.

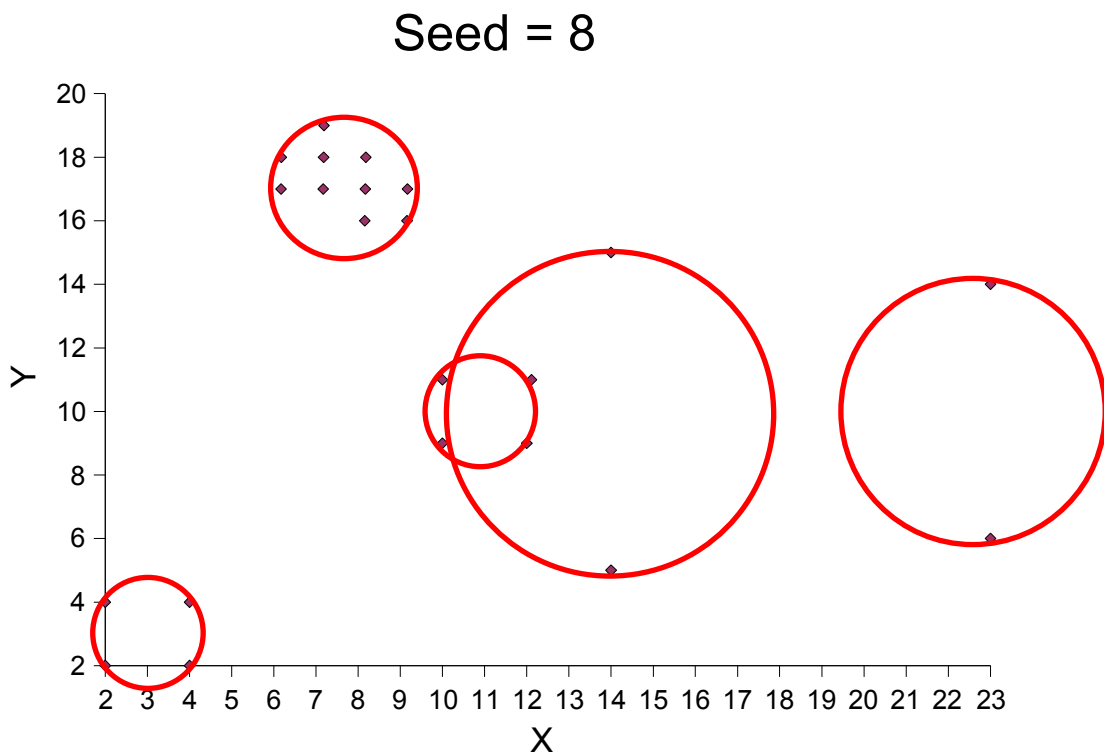


Fig 9. Même algorithme de clustering avec une Seed différentes égale à 8

Ce clustering apparaît d'ors et déjà comme un peu plus abouti et cohérent. Cependant on peut noter d'une part que les deux clusters de droite restent encore disproportionnés par rapport aux données qu'ils comportent, et d'autre part, qu'un des clusters englobe quasiment l'intégralité d'un des clusters qu'il chevauche.

Pour palier à ce problème nous avons introduit la notion d'optimisation du nombre de clusters.

Algorithme de la solution:

Fonction: runClustering(): effectue le clustering.

 getCenter(): retourne les coordonnées de centre du cluster auquel appartient l'instance.

 getCoordonne(): retourne les coordonnes de l'instance.

Data: listeInstance;

Return: meilleurSeed;

```

For seed in {0,10}
    runClustering(seed);
    for instance of listeInstance
        coordonneInstance = getCoordonne(instance);
        coordonneCentre = getCenter(instance);
        distance += distance (coordonneCentre, coordonneInstance)
    if( seed=0 )
        meilleurDistance = distance;
        meilleurSeed = 0;
    if( meilleurDistance > distance )
        meilleurDistance = distance;
        meilleurSeed = seed;
return meilleurSeed;

```

3.3.4 Optimisation du nombre de clusters

L'algorithme du simple kmeans, ne sélectionne pas automatiquement le nombre de cluster qui lui parait judicieux d'avoir. Au démarrage de l'algorithme on définit un nombre de clusters que l'on souhaite avoir et en fonction de cela, l'algorithme repartit les clusters de la façon qui lui semble la plus judicieuse en jouant sur la taille des clusters.

Ainsi comme nous avons pu le voir dans les exemples précédents on peut se retrouver avec des situations où un cluster très gros ne comporte que 2 points, tandis qu'un autre très petit va en comporter énormément.

Ce résultat engendre deux problèmes principaux dans la logique one classe.

Premièrement, un cluster qui occupe 1/3 de l'espace alors qu'il ne contient que 2 sites, est fortement préjudiciable pour notre classification future. Il va générer un nombre de faux positifs très important en raison de l'espace qu'il

couvre et qui ne représente pas notre classe.

Le deuxième problème est que de tels clusters, englobent dans certains cas, des clusters plus petits, et potentiellement plus denses donc beaucoup plus représentatifs.

Ceci confirme que le cluster important occupe beaucoup trop d'espace, et rend tous les clusters englobés totalement inutiles. Ce qui au final réduit la précision du modèle en le privant d'un certain nombre de clusters.

Notre approche pour solutionner ce problème a été de calculer le nombre de clusters englobés suite à la génération du modèle, et de re-effectuer le clustering en ajoutant autant de clusters que le nombre trouvé. De cette façon au bout d'un certain temps l'algorithme s'équilibre jusqu'à obtenir des clusters totalement indépendants.

Considérant deux clusters A et B, C_A C_B leurs centres respectif, et R_A , R_B leurs rayons.

Soit la distance entre A et B:

$$D_{AB} = \text{Distance} (C_A / C_B)$$

On considère que si:

$$R_A > D_{AB}$$

ou

$$R_B > D_{AB}$$

Alors les clusters sont superposés.

Algorithme:

Data: cluster -> liste des clusters

Return : nbClusterToAdd

```
nbClusterToAdd = 0;
Forall cluster
    A= cluster en cours;
    forall cluster except cluster A
        B = cluster en cours
        if ( $R_A > D_{AB}$  ou  $R_B > D_{AB}$ )
            nbClusterToAdd ++;
return nbClusterToAdd;
```

Au final on effectue le clustering avec le nombre de cluster précédemment utilisé auquel on ajoute le nombre de clusters superposés que l'on vient de trouver. L'algorithme se termine quand il n'y a plus de cluster superposés.

3.4 Seuillage

Le seuillage correspond à la façon dont nous allons nous y prendre pour categoriser si un nouveau site est considéré comme appartenant à la classe que nous évaluons ou pas. Nous avons mis au point deux méthodes différentes chacune ayant ses avantages et ses inconvénients.

3.4.1 Rayon max

La première méthode très simple rentre totalement dans l'optique d'une solution facilement incrémentale. Elle consiste simplement à caractériser un cluster généré par notre algorithme par son centre et ce que nous appelons rayon max (fig 5.).

Dans cette optique le cluster est considéré comme une énorme sphère multidimensionnelle déterminée par le rayon max.

Pour évaluer par la suite si un nouveau site est à rejeter ou à laisser passer au travers du filtre, il nous suffit de placer ce site sur notre espace, et regarder si il rentre ou non dans un cluster.

A la suite du clustering nous possédons les coordonnées de chacun des centres ainsi que la taille de leurs rayons max.

Il nous suffit donc de calculer la distance entre le point que l'on souhaite évaluer et les différents centres des clusters. Dès que cette distance est inférieure

au rayon max du cluster évalué, on categorise ce site comme appartenant à la classe à filtrer.

Algorithme

Fonction getCoordonne(): retourne les coordonnées d'une instance

Data: instance (site a évaluer), listeCluster (resultant du modèle de clustering)

Return : boolean (true -> correspond à la classe à filtrer, false -> ok)

```
match=false;
For cluster of listeCluster
    A= Cluster en cours;
    Ra = rayon max du cluster;
    Ca= getCoordonne(A);
    Ci= getCoordonne(instance);
    Dia= distance ( Ca, Ci );
    if ( Dia <= Ra)
        match = true;
        break;
return match;
```

Cette solution à plusieurs avantages non négligeables.

Premièrement, les données nécessaires à l'évaluation des sites sont simplement les coordonnées des différents centres des clusters ainsi que la taille de leurs rayons max associés. Le faible volume de données manipulées permet d'avoir des traitements rapides. Nous avons exposé précédemment la possibilité de multiplier le nombre de domaines à filtrer et d'ajouter ces domaines progressivement, sous forme de modules, à l'application déployée sur le serveur. Dans le cas de volumes de données importantes, l'ajout de ces nouveaux modules aurait pu apporter des contraintes de lourdeur. Dans cette solution les données nécessaires à la gestion d'un domaine étant très faible l'ajout en cascade de plusieurs modules ne risque pas d'altérer le temps de traitement.

Deuxièmement la simplicité de l'algorithme mis en jeu pour analyser les sites entrant, le rend extrêmement rapide, ce qui renforce encore les contraintes de temps de traitement que nous nous étions fixés durant la problématique du sujet abordé.

3.4.2 Analyse par dimensions

Le clustering de type simpleKmeans appartenant à la solution logiciel Weka, que nous présenterons un peu plus loin, intègre par défaut la possibilité d'obtenir les différentes déviation standard pour chaque attribut des clusters obtenus (dimension de l'espace de travail). Cette donnée est intéressante dans notre cas puisqu'elle nous permettrait de moduler la forme de nos clusters. Nous pourrions dès lors obtenir une forme multidimensionnelle qui comporterait une surface beaucoup moins importante que celle obtenue dans l'utilisation unique du rayon max.

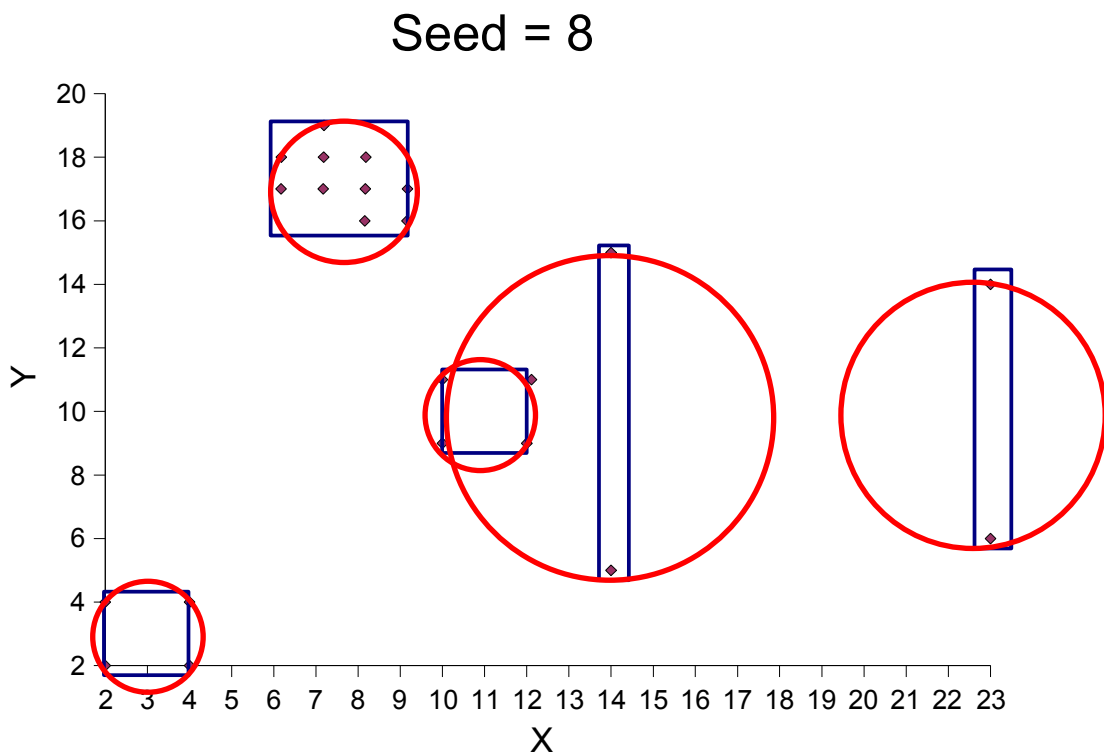


Fig 10. diagramme représentant en rouge le résultat de la zone couverte par un cluster évaluant le rayon max et en bleu le résultat de la zone couverte par un cluster utilisant le principe de l'écart type par dimensions pour évaluer l'appartenance.

On voit très nettement que selon les clusters ce procédé apparaît plus qu'intéressant et évite de considérer toute une zone de l'ensemble qui n'a pas forcément raison de l'être.

Algorithme

Data: listClusterFonction: coordonneForDimension(): retourne la coordonnée de l'instance pour la dimension indiquée;

deviationStandart(): retourne la déviation standard pour la dimension du cluster indiqué.

Return: match

```

for all cluster of listeCluster
    A = cluster;
    match = true;
    for all dimensions of A
        Da= dimension en cours;
        Ca= coordonneForDimension(A);
        DevA= deviationStandart( Da);
        Ci= coordonneForDimension(Instance);
        if ( (Ca - 2*DevA) >= Ci >= (Ca+ 2*DevA) )
            match = false;
            break;
    if ( match == true )
        break;
return match;

```

L'algorithme présenté ci-dessus fait l'hypothèse, en entrée, que chaque site présenté à un cluster lui appartient. Il vérifie cette hypothèse, ensuite, pour chaque dimension du cluster. Si il s'avère que le site ne correspond pas avec une dimension du cluster, il décline du fait le site du cluster et passe au cluster suivant. Des qu'il a pu vérifier qu'un cluster contient le site, l'algorithme se termine.

Cette solution bien qu'un peu plus lourde nous permet de travailler sur des clusters qui ont maintenant une forme multidimensionnelle qui représente directement les différentes valeurs des instances selon les dimensions.

Nous n'avons plus maintenant de forme sphérique, mais une forme optimisée en terme de superficie couverte. Du fait l'espace couvert par chaque cluster dans l'espace est diminué, ce qui implique une diminution des faux positifs.

3.5 Expérimentation

3.5.1 Implémentation

3.5.1.1 Weka

Weka est un ensemble de classes et d'algorithmes en Java implémentant les principaux algorithmes de data mining.

Il est disponible gratuitement à l'adresse [/www.cs.waikato.ac.nz/ml/weka](http://www.cs.waikato.ac.nz/ml/weka), dans des versions pour Unix et Windows.

Weka peut s'utiliser de plusieurs façons :

- Par l'intermédiaire d'une interface utilisateur.
- Par ligne de commande.
- Par l'utilisation des classes fournies à l'intérieur de programmes Java : toutes l'api étant très bien documentées.

Nous avons retenu la dernière solution qui nous a permis de manipuler et d'ajouter des fonctionnalités supplémentaire à l'application afin de convenir à nos besoins.

Weka utilise des fichiers de type « .arff » qui représentent les fichiers de corpus. Ces fichiers sont composés d'un entête simple décrivant le fichier, suivit de la liste des attributs et leur type et pour finir des différentes valeurs.

Ces fichiers représentent ni plus ni moins qu'une matrice correspondant au corpus.

3.5.1.2 Eclipse

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.

I.B.M. est à l'origine du développement d'Eclipse qui est d'ailleurs toujours le coeur de son outil Websphere Studio Workbench (WSW), lui même à la base de la famille des derniers outils de développement en Java d'I.B.M. Tout le code d'Eclipse a été donné à la communauté par I.B.M afin de poursuivre son développement.

L'ensemble des développements effectués durant le stage ont été effectués à l'aide de cet outil.

3.5.1.3 Codage

Un grand nombre de classes ont été créées afin de réaliser les nombreuses manipulations et tests des fichiers nécessaires.

Pour ce faire nous avons implémenté différents packages nous permettant d'effectuer les différents traitements présentés dans ce rapport.

On peut citer un package dédié aux pré-traitements des données. En effet le corpus en notre possession pour effectuer les tests, n'était pas un corpus one classe. De ce fait nous avons dû effectuer plusieurs pré-traitements nous permettant de recréer un corpus de type One-classe à partir d'un de type multi-classe.

Nous avons ensuite développé un package permettant la réalisation de la sélection d'attributs, un autre concernant la réalisation du clustering, plus différents petits packages contenant des classes plus particulières comme pour la manipulation des fichiers Arff, la normalisation des valeurs, l'algorithme de clustering flou... Nous avons implémenté ce dernier, mais nous n'avons pas pu encore totalement l'exploiter à ce jour.

Toutes les classes produites ont été soigneusement commentées afin d'être facilement exploitable par des tierces personnes qui reprendraient le projet par la suite(cf annexe).

3.5.2 Résultats

3.5.2.1 Évaluation sur l'ensemble des Iris

Afin de valider notre approche, nous avons réalisé des expérimentations sur l'ensemble des iris.

Le corpus des iris est un corpus bien connu dans le monde du data-mining, puisqu'il est très approprié pour tester les algorithmes de par le fait que ces classes sont très distinctes les unes des autres. On le retrouve fréquemment dans les publications pour valider des tests ou réaliser des comparatifs de performances.

D'après nos tests, l'utilisation d'une solution de simpleKmeans

traditionnelle (nombre de cluster choisi de façon aléatoire) les résultats obtenus sont loin d'être parfait. En comparaison notre algorithme de bestClustering obtient pour chaque essai 100% de bonne classification.

Nous avons réalisé nos expérimentations de façon à sélectionner successivement une des classes du corpus Iris et à générer un corpus ne comportant que cette classe. Cette manipulation ayant pour but de simuler le one classe.

Nous évaluons par la suite le modèle obtenu sur le corpus Iris complet.

Pour chacune des classes isolée en One-classe nous avons obtenu 100% de bonne classification (vrai positif et vrai négatif).

Nous n'avons pas ici utilisé notre algorithme de sélection d'attributs puisque le corpus des iris est très léger et ne comporte que quatre attributs.

3.5.2.2 Résultats sur corpus d'Adamentium

Une fois notre procédure validée sur le corpus des iris, nous avons lancé une batterie de tests sur le corpus mis à disposition par la société Adamentium.

Nous avons dans le cadre des tests, utilisé les deux méthodes de classification des nouveaux sites, afin de pouvoir les comparer.

Dans le même temps, nous avons fait varier les paramètres de couverture, et de seuil de selection d'attributs.

Au départ du processus Adamentium nous a fourni un corpus sur la langue anglaise. Ce corpus est composé de deux fichiers: un pour la génération du modèle(le fichier d'apprentissage) et un pour l'évaluation du modèle(le fichier test). Ces fichiers sont d'une taille relativement conséquente avant traitement puisqu'il font à eux deux près d'un 1Go de données brut avec 62437 attributs (mots).

Comme précisé précédemment notre premier traitement a été de passer le corpus permettant de créer le modèle, en corpus One-class. Après ce traitement le fichier est réduit de façon considérable, et ne comporte "plus" que 8974 attributs.

Nous avons ensuite appliqué notre processus de sélection d'attributs sur ce fichier. Au final nous obtenons un ensemble inférieur à 100 attributs pour une couverture a 99%.

En analysant ce fichier nous constatons que les termes considérés comme

les plus représentatifs pour le domaine étudié (dans notre cas la pornographie) sont pertinents (cf Annexe).

Ces résultats nous laissent présumer que la première étape de notre proposition est correctement exécuté, et peut être considéré comme une méthode fiable de sélection d'attributs dans une optique de classification One-class.

Au terme de cette première manipulation nous obtenons un jeu de plusieurs fichiers de corpus différents selon le seuil, et la couverture utilisée durant la sélection d'attributs.

Une fois ces corpus créés, avec un nombre d'attributs réduit au minimum, et surtout aux termes les plus pertinents pour la classe étudiée, nous procédons au clustering.

Une fois le modèle obtenu nous les évaluons avec le corpus de test. En résulte un ensemble de résultats que nous mettons en comparaison afin d'en déduire des tendances.

Rayon Max:

Pornographie -> taux de bonne détection d'un site de catégorie pornographie

Divers -> taux de bonne détection d'un site de catégorie divers

Seuil = 1

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	89,92	86,72	87,48
Divers	33,98	31,02	5,08

Seuil = 4

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	78,44	75,89	95,01
Divers	42,17	34,9	5,43

Seuil = 6

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	76,37	79,28	93,69
Divers	42,94	33,43	4,99

Seuil = 10

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	73,92	80,04	95,86
Divers	43,12	32,78	3,88

Fig 11. résultats pour une évaluation basée sur le rayon max du cluster

Ce premier jeu de résultats nous laisse paraître que les meilleurs résultats se retrouvent pour une couverture faible aux alentours de 90%. On s'aperçoit que plus on ajoute de la couverture et plus le taux de bonne détection de la catégorie « divers » s'amointrit.

Évaluation par dimensions

Seuil = 1

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	89,92	84,37	39,74
Divers	33,98	30,75	59,56

Seuil = 4

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	77,31	87,66	53,77
Divers	42,38	7,76	34,26

Seuil = 6

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	76,84	87,19	57,44
Divers	47,09	11,08	36,84

Seuil = 10

	Couverture 90%	Couverture 95%	Couverture 99%
Pornographie	78,81	90,3	64,22
Divers	46,91	10,71	19,85

Fig 12.résultats pour une évaluation effectuée par dimensions pour chaque cluster

Ce deuxième jeu de résultat correspond à une évaluation s'effectuant par dimensions sur les clusters. Au vu des résultats on peut considérer que cette évaluation obtient de meilleurs résultats que l'évaluation en rayon max. Ici encore la couverture à 90 % semble obtenir les meilleurs résultats.

On peut donc penser qu'un trop grand nombre d'attributs perturbe la classification et qu'un petit nombre très représentatif de la classe à évaluer est plus efficace dans le cadre d'une classification basée sur le simpleKmeans.

Comme nous l'avons défini dans la problématique, la principale difficulté apportée par la solution One-classe résulte dans la capacité à sélectionner les attributs représentant la classe étudiée.

Au vu des résultats de nos expérimentations nous pouvons estimer que notre proposition est fonctionnelle et que le processus de sélection d'attributs offre des résultats satisfaisants, permettant d'envisager une bonne classification par la suite.

Les résultats nous montrent que nous arrivons à obtenir une bonne distinction de la classe étudiée (ici pornographie).

Cependant les résultats sur la catégorie du divers restent à améliorer.

Chapitre 4. Conclusion et perspectives

Nous avons présenté dans ce rapport une solution de méthode de classification basée sur un principe One-Class. Le choix d'une optique One-classe se justifie par l'impossibilité de couvrir un ensemble représentant le domaine du « divers » en opposition à une classe définie.

Notre proposition regroupe deux axes principaux, (i) la sélection d'attributs dans un contexte One-class, et (ii) la classification non supervisée et l'évaluation d'un corpus de type One-class.

Pour mettre en place la solution de sélection d'attributs, nous avons appliqué différents traitements permettant de contrecarrer l'absence de plusieurs classes. Notre algorithme peut s'appliquer sur n'importe quel corpus textuel et obtient à ce jour des résultats concluant d'après nos tests.

La deuxième étape de la classification, regroupant la génération d'un modèle ainsi que les techniques d'évaluations attachées à celui-ci, a été conduite via l'utilisation d'une méthode de classification non supervisée de type kmeans. Bien que nous ayons apporté des modifications ayant pour but d'améliorer ce clustering un travail reste à produire sur cette partie afin d'augmenter les capacités du filtre.

Les poursuites à donner à cette étude s'orienteront vers l'utilisation des notions de structure de la page Web[12] pour la sélection d'attributs; le développement d'un algorithme intégré au kmeans concernant le repérage et l'isolement des clusters denses au cour du processus d'optimisation mis en oeuvre; et les tests d'autres méthodes de clustering tel que le fuzzy C-means.

BIBLIOGRAPHIE

[1] Salton, G., Wong, A. and Yu, C. T. (1976). *Automatic Indexing Using Term Discrimination and Term Precision Measurements*, Information Processing & Management 12: 43--51, Pergamon Press Ltd., ISSN 0306-4573.

[2] D. M. J. Tax. One-Class Classification: Concept-Learning in the Absence of Counter-Examples. PhD thesis, Delft University of Technology, June 2001

[3] Mark A. Hall, Georey Holmes. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining, IEEE transactions on knowledge and data engineering, Vol. 15, No. 3, May/June 2003

[4] Yang, Y. & Pedersen, Jan O. (1997), *A Comparative Study on Feature Selection in Text Categorization*, in 'Proceedings of ICML-97, 14th International Conference on Machine Learning', pp. 412--420.

[5] Liu, T., Liu, S., Chen, Z. & Ma, W. (2003), *An Evaluation on Feature Selection for Text Clustering*, in 'Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)'.

[6] Fayyad, U. M. & Irani K. B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022-1027. Morgan Kaufmann.

[7] Wai-Chiu Wong, Ada Wai-Chee Fu, Incremental Document Clustering for Web Page Classification, Chinese University of Hong Kong, July 2000.

[8] M. Beigbeder et A. Mercier. Étude des distributions de tf et de idf sur une collection de 5 millions de pages HTML. in actes de l'atelier «Recherche d'information; un nouveau passage à l'échelle», associé à Inforsid 2003 Nancy, France, 3 juin 2003.

[9] S. L. Chiu – Fuzzy Model Identification Based on Cluster Estimation. Journal of Intelligent and Fuzzy System, 2 :267-278, 1994.

[10] J.C. Bezdek – Pattern Recognition with fuzzy Objective Function Algorithm. New York, Plenum Press, 1981

[11] Lucene, solution de recherche de texte de la fondation Apache <http://lucene.apache.org/java/docs/index.html>

[12] Min Zhang, Ruihua Song, and Shaoping Ma, DF or IDF? On the Use of HTML Primary Feature Fields for Web IR, the 12th World Wide Web conference, (www2003), poster, Hungarian, 2003.

ANNEXES

Table des matières

Diagramme des classes de notre algorithme d'optimisation du clustering SimpleKmeans présent dans la solution Weka.....	52
TF-IDF.....	53
Capture d'écran de JavaDoc.....	54
Résultat selection d'attributs.....	56

Diagramme des classes de notre algorithme d'optimisation du clustering SimpleKmeans présent dans la solution Weka.

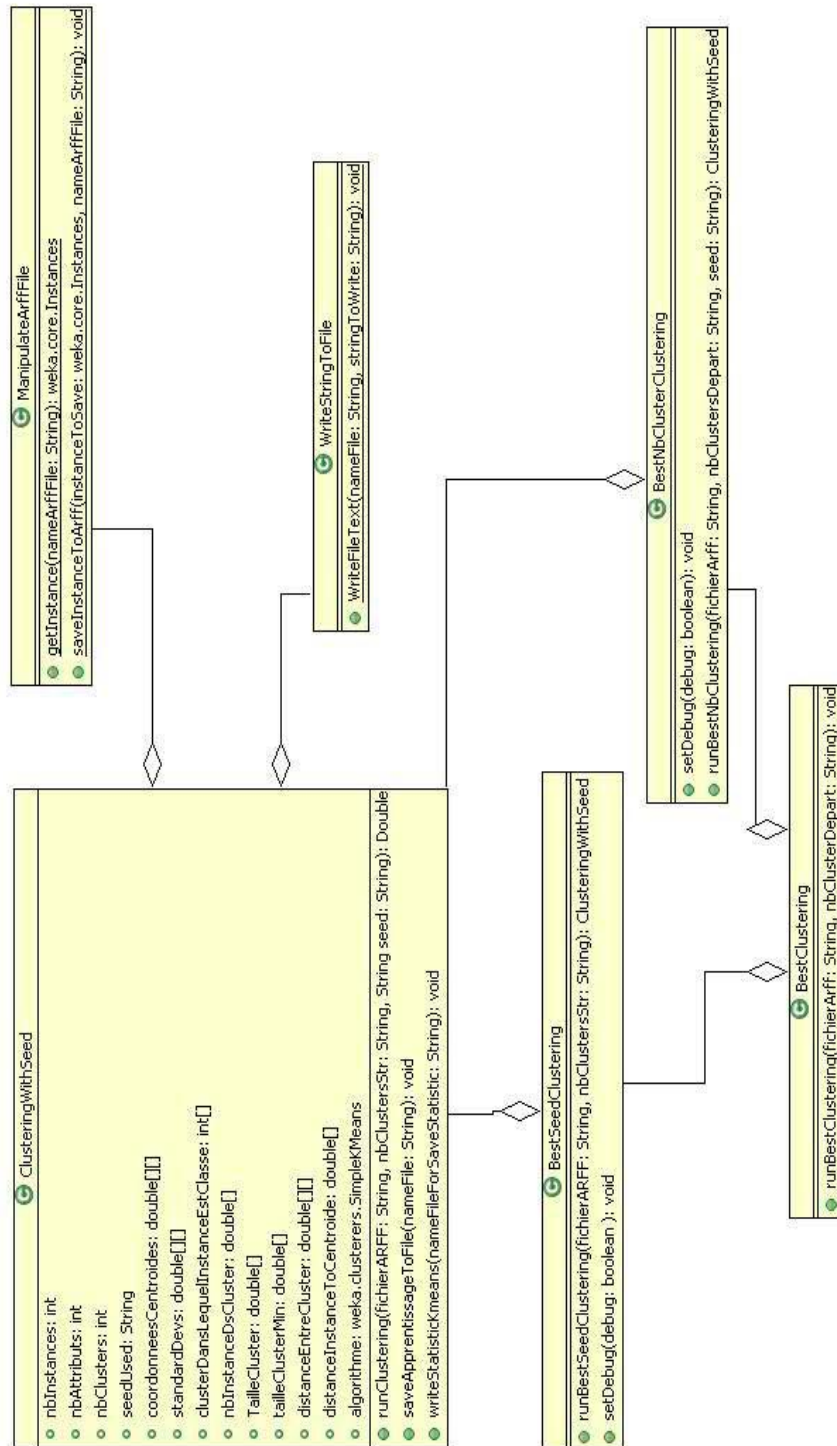


Fig 1: Diagramme des classes de notre application du simpleKmeans

TF-IDF

<http://www.grappa.univ-lille3.fr/~gilleron/tfidf.pdf>

Capture d'écran de JavaDoc

The screenshot displays the JavaDoc interface for the 'weka.manipulation' package. On the left side, there are two panels: 'All Classes' and 'All Packages'. The 'All Packages' panel lists various sub-packages under 'weka.manipulation', including 'couverture', 'createOneClass', 'evaluationIris', 'featureSelection', 'FonctionCourant', 'fuzzyCluster', 'normalisation', and 'oneClass.stemm'. The 'All Classes' panel lists numerous classes, such as 'ApprentissageClusterer', 'BestClustering', 'Chrono', 'ClusteringWithSeed', 'CompareArff', 'CopyOfApprentissageClusterer', 'Couverture', 'CreateOneClass', 'EvalClusteringIris', 'FeatureSelection', 'Fonctions', 'FuzzyClusterer', 'InformationOnArffFile', 'ManipulateArffFile', 'MergeValue', 'NormalisationLineaire', 'OptimizedClustering', 'PorterStemmer', 'RemoveAttribut', 'RemoveInstance', 'RunFuzzyCluster', 'RunFuzzyClusterLocal', 'Sparsing', 'StemArffFile', 'StemEnglishArffFile', 'TermeFrequency', 'TestClassificationWeka', 'TestClusterer', 'TestClustererForApp', and 'TestStandardDevs'.

The right side of the interface shows the 'Overview' page for the 'weka.manipulation' package. It includes navigation links for 'Package', 'Class', 'Use', 'Tree', 'Deprecated', 'Index', and 'Help'. Below these links is a table titled 'Packages' that lists the sub-packages and their corresponding class counts:

Package	Class Count
weka.manipulation	
weka.manipulation.couverture	
weka.manipulation.createOneClass	
weka.manipulation.evaluationIris	
weka.manipulation.featureSelection	
weka.manipulation.FonctionCourante	
weka.manipulation.fuzzyCluster	
weka.manipulation.normalisation	
weka.manipulation.oneClass.stemmer	
weka.manipulation.simpleKmeans	
weka.manipulation.termesFrequency	

Below the table, there is another 'Overview' section with the same navigation links, but it is currently empty.

Fig 2: Capture d'écran de la javaDoc de l'ensemble des classes et des packages

weka.manipulation.couverture

weka.manipulation.createOne

weka.manipulation.evaluation

weka.manipulation.featureSel

weka.manipulation.FonctionC

weka.manipulation.fuzzyClust

weka.manipulation.normalisat

weka.manipulation.oneClass.s

weka.manipulation.simpleKme

weka.manipulation.termesFre

weka.manipulation.simpleKme

Classes

[ApprentissageClusterer](#)

[ApprentissageClustererTest](#)

[BestClustering](#)

[BestNbClusterClustering](#)

[BestSeedClustering](#)

[ClusteringWithSeed](#)

[CopyOfApprentissageClusterer](#)

Fonctions

[TestClassificationWeka](#)

[TestClusterer](#)

[TestClustererForApp](#)

[TestStandardDevs](#)

Overview **Package** Class [Use Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV PACKAGE](#) [NEXT PACKAGE](#) [FRAMES](#) [NO FRAMES](#)

Package weka.manipulation.simpleKmeans

Class Summary	
ApprentissageClusterer	Effectue le clustering suivant l'algorithme simple kmeans d'un fichier arff (ce fichier arff ne doit pas comporter d'attribut class.)
ApprentissageClustererTest	Effectue le clustering suivant l'algorithme simple kmeans d'un fichier arff
BestClustering	Realise un clustering de type simpleKmeans optimis�� pour un fichier Arff donn��
BestNbClusterClustering	
BestSeedClustering	
ClusteringWithSeed	Effectue le clustering suivant l'algorithme simple kmeans d'un fichier arff (ce fichier arff ne doit pas comporter d'attribut class.)
CopyOfApprentissageClusterer	Effectue le clustering suivant l'algorithme simple kmeans d'un fichier arff (ce fichier arff ne doit pas comporter d'attribut class.)
Fonctions	Deprecated.
TestClassificationWeka	
TestClusterer	
TestClustererForApp	
TestStandardDevs	

Overview **Package** Class [Use Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV PACKAGE](#) [NEXT PACKAGE](#) [FRAMES](#) [NO FRAMES](#)

Fig 3: Capture d'  cran de la javaDoc du package simpleKmeans

Résultat selection d'attributs

@attribute	sex	numeric
@attribute	adult	numeric
@attribute	click	numeric
@attribute	girl	numeric
@attribute	video	numeric
@attribute	link	numeric
@attribute	pictur	numeric
@attribute	sexi	numeric
@attribute	porn	numeric
@attribute	page	numeric
@attribute	fuck	numeric
@attribute	sexual	numeric
@attribute	cock	numeric
@attribute	galleri	numeric
@attribute	copyright	numeric
@attribute	pic	numeric
@attribute	anal	numeric
@attribute	toi	numeric
@attribute	hardcor	numeric
@attribute	erot	numeric
@attribute	reservo	numeric
@attribute	model	numeric
@attribute	contact	numeric
@attribute	web	numeric
@attribute	pussi	numeric

Fig 4: Premiers attribut résultant de notre sélection d'attributs