الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Nº Réf :……

Centre Universitaire
Abdelhafid Boussouf Mila

Institut des Sciences et Technologie **Département de Mathématiques et Informatique**

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Une approche basée structure pour la classification des documents XML

Préparé par :
Yahioune Khadidja
Chebbat Farah

Soutenue devant le jury

Président	**Attia mourad.**	**Grade MAA. Université**	**C.U.Abd Elhafid Boussouf.**
Examinateur	**Hadji Atmane.**	**Grade MAA. Université**	**C.U.Abd Elhafid Boussouf.**
Encadreur	**Chelli Djilali.**	**Grade MAA. Université**	**C.U.Abd Elhafid Boussouf.**

Année Universitaire : 2018/2019

DÉDICACE

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tous simplement que : Je dédie ce mémoire de master à :

À mon très cher père **Achour** : Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail et le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation le long de ces années.

À ma tendre mère **fatiha** : Tu représente pour moi la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Tu as fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études.

À mes chères sœurs **Hayat, Chourouk**, mon cher frère **Marouane** et ma grande-mère **Hadda** pour leur encouragements permanents, et leur soutien moral.

À mon cher fiancé et mon future homme **Houssam**, de m'avoir donné le courage de finir ce mémoire dans de bonne conditions,que DIEU le protège et le garde.

À mes anges **Sirage** et **Firas** que dieu leur procure bonne santé et longue vie.

À tous les membres de ma famille, mes oncles,tantes,cousins,cousines et surtout mes perles, mes cousines **Khouloud**, **Souhila**, **Rima** et **Amal**.

À mes plus proches copines, la cause et la source de ma joie :

Amira,**Rima**, **Sara**,**Sara2**, **Asma**, **Imene**.

À tous les membres de ma promotion du master 2019, et surtout **Asma**, **Yasser**, et ma binôme **Farah**.

À tous mes enseignants depuis mes premières années d'études, et surtout monsieur **Benchakhchoukhe Mostapha** et monsieur **Kara Naaman**.

À mon prof **Khenioui** M^{ed}**.Amine**, d'avoir qui n'as pas cessé de me conseiller, encourager et soutenir tout au long de mon travail, que dieu le protège.

À tous qui occupe une place dans ma vie, dans mon cœur que j'ai omis de citer.

Khadidja

DÉDICACE

Je tiens en tout premier lieu à remercier le dieu. Je voudrais dédie ce modeste travail À mes très cher parents qui m'ont tant soutenu et encouragé dans tous les domaines et surtout pour réaliser ce mémoire que dieu les protèges.
À mon marie qui m'a encouragé, que dieu le protège.À ma chère grande mère qui m'a soutenu et encourager et à l'âme de mon cher grand père.
À ma sœurs, mes deux frères, mes tantes, mes cousines,mes amies et toute Personne qui me connaît.

Farah

REMERCIEMENTS

Nous remercions en tout premier lieu **Allah**, qui nous a donné le courage, la volonté et la patience de mener à bien ce travail.
En second lieu, nous tenons à remercier notre encadreur **Mr chelli djillali**, pour l'orientation, la confiance, la patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené.
Nos vifs remerciements vont également aux membres du jury, pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions, ainsi que les enseignants du centre universitaire **Abedlhafid Boussouf** et tout les membres de l'institut des sciences et technologies en général.
Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail, et surtout notre familles.

Farah, Khadidja .

RÉSUMÉ

A l'heure actuelle, il est difficile de rechercher et d'accéder à des informations pertinentes en raison de la quantité des documents numériques disponibles .Dans ce contexte, vient ce qu'on appelle les documents XML devenu un standard pour la représentation et l'échange de données sur le web. Le nombre des fichiers du format XML augmente de plus en plus.

L'objectif principal de ce mémoire est pour chercher les documents qui répondent aux demandes des utilisateurs et de renvoyer les parties les plus importantes de ces documents, et cela à travers de la classification automatique non-supervisée qui travaille pour regrouper des documents similaires, ce qui rend la tache de trouver des informations dans une grandes quantité de fichiers facile et peu couteuse.

Mots-clés : XML, classification automatique non-supervisée.

ملـخـص

في الوقت الحالي، من الصعب البحث عن المعلومات والوصول إليها بسبب كمية الوثائق الرقمية المتاحة.

في هذا السياق، يأتي ما يسمى مستندات XML معيارًا لتمثيل وتبادل البيانات على شبكة الإنترنت. عدد ملفات التنسيقXML ينمو أكثر وأكثر.

الغرض الرئيسي من هذا العمل هو البحث عن المستندات التي تتلقى طلبات المستخدم وترجع الأجزاء الأكثر أهمية من هذه المستندات، وهذا من خلال التصنيف التلقائي غير الخاضع للرقابة الذي يعمل على تجميع وثائق مماثلة، مما يجعل مهمة البحث عن المعلومات في كمية كبيرة من الملفات سهلة وغير مكلفة.

الكلمات الأساسية: التصنيف التلقائي غير الخاضع للرقابة ,XML

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION GÉNÉRALE

La recherche d'information(RI) est un domaine historiquement lié aux sciences de l'information, on la définie comme une activité qui sert à localisé et délivré un assemble des documents à un utilisateur qui dépose une requête en fonction de son besoin en information.

Les systèmes de recherche d'information (SRI) son des outils informatiques qui réalise l'opération de la recherche d'information(RI), le but de ses systèmes et de mètre une comparaison entre une requête qui représente les besoins de l'utilisateur avec le contenu des documents ou les enregistrements.

Aujourd'hui, Le domaine informatique joue le rôle de traiter l'information pour le but de récupérer les besoins d'utilisateur à travers la construction d'index.

Notre travail dans ce mémoire vise à mener une approche basé structure pour la classification des documents XML qui sont de plus en plus reconnu comme un format standard pour la représentation des donnée sur internet. Ces documents XML possèdent une structure qui facilite leurs représentations .Nous avons connus que la majorité d'information contenu dans le champ textuel mais dans notre approche nous avons concerné juste la structure.

En plus de l'introduction, le mémoire est structuré en trois chapitres :

Dans le premier chapitre nous avons donnée des définitions générales sur le concept du document XML avec ses composants. Nous avons cités deux types de langages XML orientés documents : formatés et structurés. Les Langages XML orienté documents structurés concernent notre travail. Dans le deuxième chapitre, nous l'avons décomposé en deux partie : la première cite les différents méthodes de classification, on distingue trois méthodes et nous nous intéressons à l'algorithme K-means utilisé dans les méthodes non hiérarchiques pour la classification des documents semi-structuré. La deuxième partie contient le processus d'indexation qui est l'un des principales phases du processus de recherche d'information avec ses traitements automatisés sur un document XML. Dans le troisième chapitre nous avons met l'action sur la partie Conception et réalisation du système de classification. Nous avons cité l'architecture de notre approche et mettons les interfaces inclues dans notre application et qui sert à faire une classification sur un ensemble des documents XML comme entré.

Enfin nous terminons par une conclusion générale.

CHAPITRE 1

LES DOCUMENTS XML

1.1 Introduction

Dans ce chapitre, nous donnons les informations sur les documents XML qui sont nécessaire à l'étude de notre approche : la classification des documents XML.

1.2 La définition du XML

XML (eXtensible Markup Language) est un langage pour structurer des contenus et définir une classe d'objets de données. Le langage est basé sur le concept de balisage des données. Il est standardisé comme une recommandation par le W3C depuis 1998. Un document XML contient des déclarations, des éléments, des commentaires, des définitions de caractères spéciaux et d'instructions de traitement. Il devient Plus simple que SGML, plus complexe mais moins confus et plus performant que HTML.[1]. La figure ci-dessus présante un document XML.

1.3 Quelques définitions

* **Le path :**

  le path d'un nœud e_i est la suite de nœuds visités à partir de la racine pour atteindre le nœud e_i. Alors, la longueur d'un path est égale au nombre de nœuds qui le construit.

  Exemple : le path du noeud < it > est "body.sec.st.it", sa longueur est égale à 4. Comme indiqué dans(Figure 1.1).

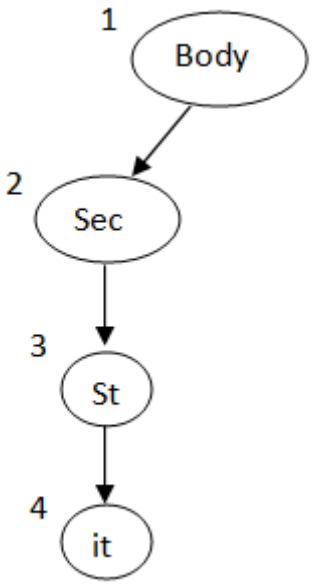

FIGURE 1.1 – Le path du nœud<it>.

* **sous-paths :**

  L'ensemble des sous-paths d'un nœud e_i est l'ensemble des sous-séquences du path de celui-ci.

  Exemple : un sous-path du nœud < it > est "sec.st.it", sa longueur est égale à 3. Comme il est représante dans(Figure 1.2).

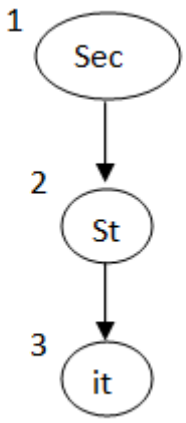

FIGURE 1.2 – Le sous-path du nœud<it>.

* **Un path textuel :**
  est un path qui se termine par un terme du contenu texte d'un nœud.
  Exemple : le path sec.au. "name", est un path textuel de longueur est égale à 3.
  La (Figure 1.3), montre l'example du path textuel.

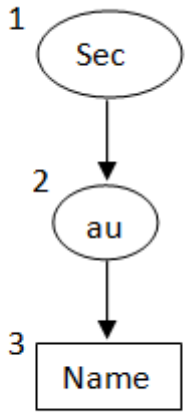

FIGURE 1.3 – Le path textuel.

Le format interne d'un document XML c'est un arbre.[2]

La représentation d'arbre de ce document est illustré dans la figure suivante (Figure 1.5).

1.4 Langages XML orientés données

Ils permettent d'enregistrer et de transporter des données informatiques structurées (comme par exemple des données gérées par des bases de données) selon des formats ouvert (c'est à dire dont on connaît la syntaxe) et facile à manipuler (les structures arborescentes XML étant plus riches que des fichiers à plat par exemple). Les langages XML orientées données servent surtout à l'échange ou la sérialisation des données des programmes informatiques.[3]

```
<?xml version="1.0" ?>
<body>
  <sec sno="01">
    <st>Title of<it>Section 1</it>: </st>
    <p>This is a paragraph</p>
    <au>Name of the first author</au>
  </sec>
  <sec sno="02">
    <st>Title 2</st>
    <p1>Another type of paragraph</p1>
    <au> Name of the second author</au>
    <bib>References</bib>
  </sec>
  <sec sno="03">
    <st>Title 3</st>
    <p>One paragraph</p>
    <bib>References</bib>
  </sec>
</body>
```

FIGURE 1.4 – Document Test.xml.

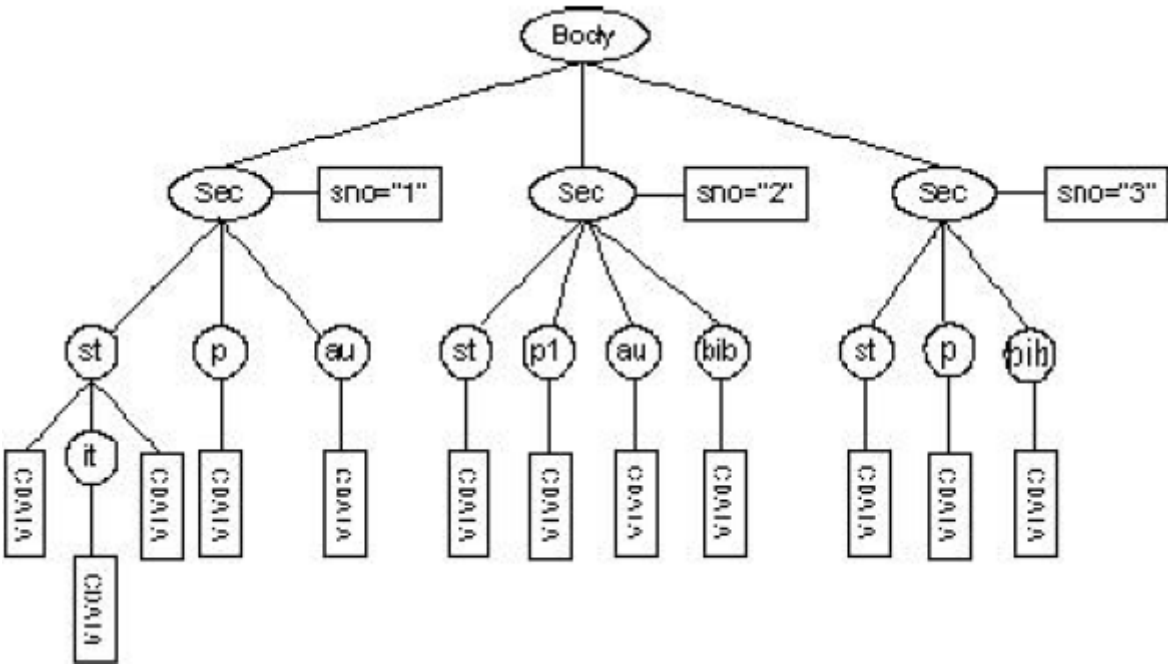

FIGURE 1.5 – Arbre représentant le document XML.

1.5 Langages XML orientés documents

Ils permettent de représenter informatiquement des documents numériques. Les formats de documents faisant appel à des représentations fortement arborescentes,

XML est un candidat idéal pour cet usage. En fait SGML, l'ancêtre de XML, a été inventé pour l'informatique documentaire. Aujourd'hui la très grande majorité des formats de représentation de document sont en XML.[3]

Il existe deux types du Langages XML orientés documents :

1.5.1 Langages XML orienté documents formatés

Ils définissent des formats de mise en forme de document (structure physique), en général pour un support donné.[3]

1.5.2 Langages XML orienté documents structurés

Ils définissent des formats de structuration de document (structure logique), en général pour un métier donné, plus ou moins précis selon la généralité du langage.[3]

1.6 Les types du langage

Il existe plusieurs types de langage.

1.6.1 Les langages de programmation

Permettent de créer des programmes, des applications mobiles, des sites internet, des systèmes d'exploitation, etc....
Certains langages de programmation sont extrêmement populaires. En Mai 2012.
Les langages de programmation les plus populaires sont le C, le Java, le C++, l'Objective-C, le PHP, le Basic, le Python, le Perl et le JavaScript.[4]

1.6.2 Les langages de requêtes

Permettent quant à eux d'interroger des structures qui contiennent des données. Parmi les langages de requête les plus connus, on peut par exemple citer le SQL pour les bases de données relationnelles, le SPARQL pour les graphes.[4]

1.6.3 les langages de description

Permettent de décrire et structurer un ensemble de données selon un jeu de règles et des contraintes définies. On peut par exemple utiliser ce type de langage pour décrire l'ensemble des livres d'une bibliothèque, ou encore la liste des chansons d'un CD, etc. Parmi les langages de description les plus connus, on peut citer le SGML, le **XML** ou encore le HTML.[4]

1.7 Origine du XML

Le langage XML n'a pas été créé pour s'amuser. En effet, sa création avait pour objectif de répondre à un besoin très précis : l'échange de données. Dans les débuts d'internet, les ordinateurs et les programmes échangeaient des données via des fichiers. Mais malheureusement, Ces fichiers avaient bien souvent des règles de formatage qui leur étaient propres. Par exemple, les données étaient séparées des points, des virgules, des espaces, des tirets, etc.... Le problème avec ce système est qu'il fallait sans cesse adapter les programmes au format du fichier ce qui représentait une charge de travail importante.[5]

1.8 Structure d'un document XML

Un document XML peut être découpé en deux parties : **le prologue et le corps.**[3]

1.8.1 Le prologue

Il est facultatif et comprend une déclaration XML, indiquant la version du langage XML utilisé et le codage des caractères dans le document. Chacune de ces informations est optionnelle mais leur ordre est obligatoire.[3]
Voici à quoi va ressembler un prologue(figure 1.6) :

```
<?xml version="1.0" encoding="UTF-8"?>
```

FIGURE 1.6 – Le prologue d'un document XML.

Cette figure indique que le document est codé en utilisant un langage XML de version 1.0, avec des caractères codés selon la norme UTF-8.

1.8.2 Le corps

Le corps d'un document XML est constitué de l'ensemble des balises qui décrivent les données. Il y a cependant une règle très importante à respecter dans la constitution du corps :" une balise en paires unique doit contenir toutes les autres ". Cette balise est appelée " élément racine du corps ".[3]

1. **Nœud dans un document XML**
   Un document XML est composé de nœuds, un nœud peut être :[6]
   - Le document lui-même, représenté par son nœud racine.
   - Un espace de nom, qui est définie dans notre chapitre.
   - Des commentaires, par exemple :< !- -mon fichier d'adresses- > (Figure 1.7)

   - **Elément de données :**
     C'est un texte encadré par une balise de début et une balise de fin, les éléments de données peuvent être imbriqués. Le contenu textuel de ses éléments devient aussi un nœud
     .

- **Attribut :**

  Spécifié par un couple " nom="valeur" " qualifiant une balise comme <personne id= "p01" />.

  La figure ci-dessus montre les composants d'un document XML.

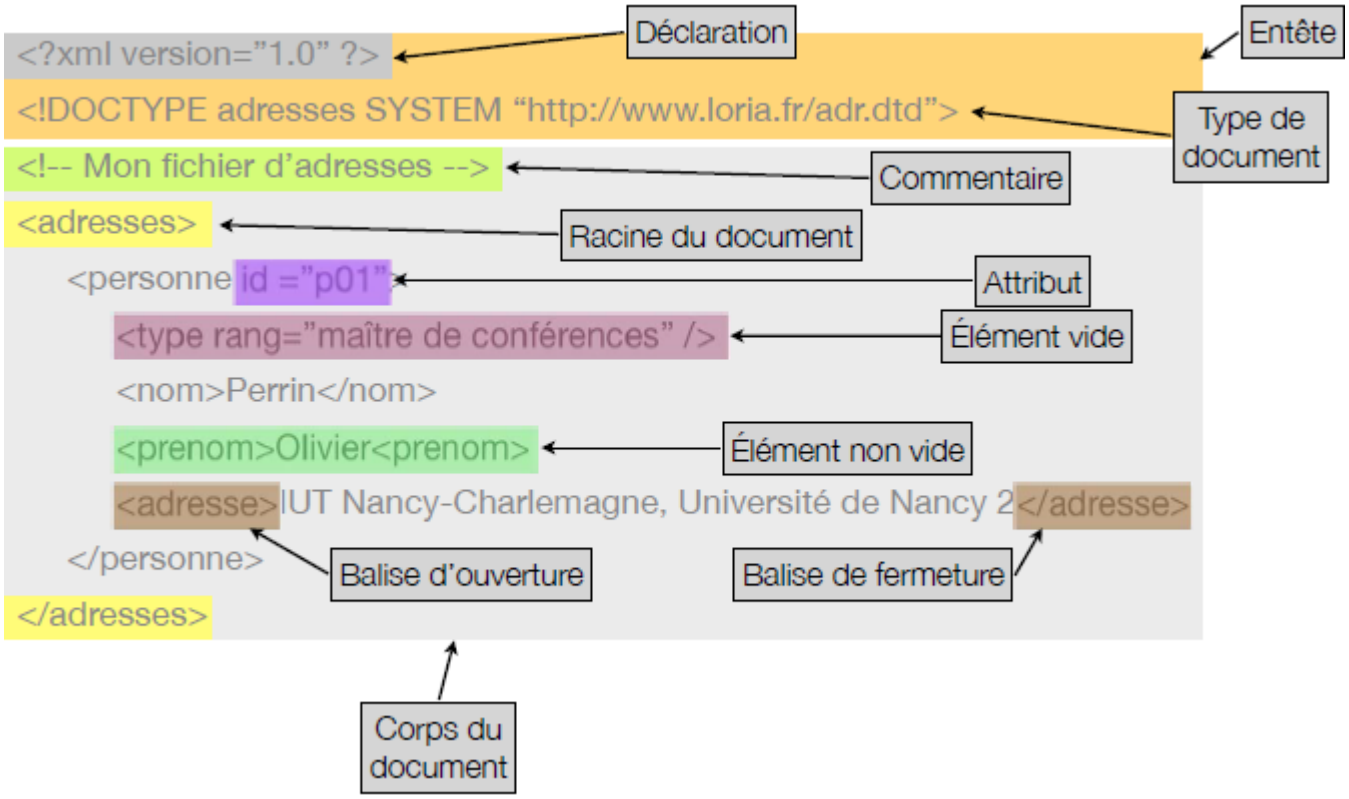

FIGURE 1.7 – Un document XML avec ses composants.

2. **Relation entre les nœuds dans un document XML**

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- un premier marin -->
<marin  id="12">
        <nom>Surcouf</nom>
        <prenom>Robert</prenom>
        <remarque  lang="FR">Capitaine corsaire</remarque>
</marin>
```

FIGURE 1.8 – Exemple simple d'un document XML.

A partir de la (figure 1.8) on définit les relations suivantes :

- **relation parent :**
  tous les éléments et les attributs ont un parent. Dans notre exemple, marin est parent de nom et prénom.
- **relation enfant :**
  un nœud peut avoir autant d'enfants que l'on veut. Dans notre exemple, marin à deux enfants : nom et prénom, le nœud nom n'a pas d'enfant.
- **relation frère (sibling) :**
  deux nœuds sont frères s'ils ont même père. Dans notre exemple, nom et prénom sont frères.
- **relation ancêtre (ancestor) :**
  les ancêtres d'un nœud sont son père, le père de son père, etc...jusqu'au nœud racine, ancêtre de tous les nœuds du document XML.
- **relation descendants :**
  les fils, fils des fils, etc.... Le nœud racine du document a pour descendance l'intégralité des nœuds du document XML.

3. **Valeurs atomiques dans un document XML**

   Les valeurs atomiques, au sens où l'on ne peut pas les diviser, sont les nœuds qui n'ont ni parents, ni enfants. Sur l'exemple dans la (Figure 8) précédent :
   Robert : est un nœud texte, atomique.

"12" : est un nœud texte également, atomique.[6]

1.8.3 Notion d'espace de nom

Comme on le voit déjà sur notre exemple simple Dans la (Figure 1.8), certaines de nos balises ont des noms très génériques : Les balises **nom** et **prénom** par exemple. Cela peut poser un problème en cas d'échange de documents XML entre deux systèmes différents : **que se passera-t-il si un autre système a également défini des éléments nom et prénom, Mais possédant un sens différent, voire des contraintes différentes ?** Le problème se pose dans les mêmes termes lorsque l'on crée des classes Java, et est résolu de la même façon. En Java, on range nos éléments et classes dans des espaces de nom, implémentés sous forme de packages. Cette notion existe également en XML.[6]

Un espace de nom en XML est défini par deux choses : une référence, et le nom de cet espace qui est une URL, et sa référence une simple chaîne de caractères. On expliquer ça dans la (Figure 1.9).

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- un premier marin -->
<galilee:marin   id="12"
        xmlns:galilee="http://www.galilee.org/cours-xml">

        <galilee:nom>Surcouf</galilee:nom>
        <galilee:prenom>Robert</galilee:prenom>
        <galilee:remarque   lang="FR">Capitaine corsaire</galilee:remarque>
</galilee:marin>
```

FIGURE 1.9 – Exemple écrit dans l'espace de nom.

Comme on peut le voir, cette déclaration se fait par l'ajout d'un attribut standard de notre nœud racine : **xmlns :galilee**, qui prend pour valeur **http ://www.galilee.org/cours-xml**. Cette déclaration définit un espace de nom, de valeur "http ://www.galilee.org/cours-xml ", référencé par le préfixe galilee. Ensuite, les éléments de notre document ont été préfixés par galilee. De façon à les attacher à l'espace de noms que l'on vient de définir.

1.9 Description d'un modèle XML

Un document XML répond en général à un modèle, lui-même écrit dans un fichier. Un tel fichier porte un " descripteur de documents XML". Il existe deux formats pour de tels fichiers : le premier est appelé DTD (Document Type Définition), le second est Schéma XML. Le premier type de descripteur est assez simple, et écrit dans un langage propre et le second est plus complexe, mais aussi plus riche, et a l'avantage d'être lui-même écrit en XML. Dans les deux cas, il est possible de déclarer dans un fichier XML, que le contenu de ce fichier doit se conformer à ce qui est déclaré dans la DTD ou le XML Schéma associé. Dans ce cas, il est également possible par API, de tester la validité des documents analysés ou créés en fonction de la DTD ou du XML Schéma associé.[6]

1.10 DTD Document Type Définition

Lors de son lancement, XML a été perçu comme une réelle chance pour les développeurs d'avoir à disposition un langage simple d'utilisation, portable sans difficulté d'une machine et d'une application à une autre, et libre de droits. Dans le premier temps, un fichier XML, si on voulait le standardiser en utilisant un vrai langage général de description, devrait dépendre d'une DTD.[3]

Les DTDs utilisent un langage spécifique (non XML) pour définir les règles structurelles.

1.10.1 Association d'une DTD à un document XML

XML offre plusieurs manières d'associer une DTD à un document (DTD interne au document XML, externe mais déclarée dans le document XML, etc...).

Comme il est possible à une personne de concevoir sa propre structure et ses noms

d'éléments pour ses documents, les DTDs permettent de créer des modèles pour ces types de documents. On peut effectuer une vérification pour être sûr que d'autres documents respectent ces modèles, tandis que d'autres développeurs peuvent produire des documents compatibles.[7]

```
< ?xml  version="1.0 encoding="ISO-8859-1" ?>
<famille>
   <pere>S.Y </pere>
   <mere>M.Y nee M.B</mere>
   <enfants>
      <enfant>J.Y</enfant>
      <enfant>N.Y</enfant>
      <enfant>L.Y</enfant>
   </enfants>
</famille>
```

FIGURE 1.10 – Document XML décrivant une famille.

Un document XML est dit valide s'il est bien formé et respecté, en plus, aux règles grammaticales présentés par une DTD ou un XML-Schéma. Considérons le document de la (figure 1.10) décrivant une famille. Aucune règle ne contrôle la structure de ce document. Si on supprime l'élément <pere>, ou si l'on ajoute d'autres éléments <enfant>, cela n'empêche pas le document d'être affiché par un Navigateur. Le but d'une DTD est de définir les éléments qui peuvent être utilisés dans un document XML et de spécifier les relations entre ces éléments. Chaque balise correspond à un élément de la DTD. Dans le document de la (figure 1.10), nous avons un élément racine nommé <famille> dont la définition est :

- < !ELEMENT famille (pere, mere, enfants ?)> :
  Cet élément contient les éléments <pere>, <mere> et <enfants>. Ces éléments sont listés selon l'ordre dans lequel ils doivent apparaître logiquement dans le document. Le symbole " ? " signifie que l'élément <enfants> est optionnel. Une famille peut être valide avec seulement les deux éléments <pere> et <mere>.

Les éléments <pere> et <mere> sont définis par :

- < ! ELEMENT pere (#PCDATA)>.
- < ! ELEMENT mere (#PCDATA)>.
  PCDATA correspond à du texte qui sera analysé par le parseur XML. Le troisième élément <enfants>, ne contiendra pas de texte, mais un ou plusieurs éléments <enfant>. Ajoutons les éléments <enfants> et <enfant> :
- < !ELEMENT enfants (enfant +)>.
- < !ELEMENT enfant (#PCDATA)> .

L'élément <enfants> peut contenir des éléments <enfant>. Le signe + signifie qu'il y a au moins un élément <enfant>, et que cet élément peut être répété autant de fois que nécessaire. L'élément <enfant> contiendra du texte comme les éléments <pere> et <mere>.

En résumé la DTD associée est la suivante :

< ? Xml version=1.0 encoding='ISO-8859-1' ?>
< !ELEMENT famille (pere, mere, enfants ?)>
< !ELEMENT pere (#PCDATA)>
< !ELEMENT mere (#PCDATA)>
< !ELEMENT enfants (enfant +)>
< !ELEMENT enfant (#PCDATA)>

Pour décrire les éléments de la DTD, on emploie un formalisme emprunté à la théorie des langages, celui des expressions régulières. Par exemple :

- **" enfant + " :** signifie que l'élément <enfant> apparaît au moins une fois.
- **" enfant * " :** signifie que l'élément <enfant> peut ne pas apparaître du tout ou apparaître une ou plusieurs fois.
- **Si on a (pere, mere) :** les éléments <pere> et <mere>, doivent apparaître simultanément dans cet ordre, c'est comme si nous avions écrit (pere et mere).
- **" enfants ? " :** signifie que l'élément <enfants> est optionnel, c'est-à-dire apparaissant une fois ou pas du tout.

- **Si on écrit (pere|mere) :** choix entre <pere> et <mere>, l'ordre d'apparition n'a pas d'importance.[8]

Mais ce format de description, hérite de SGML, souffre de nombreuses déficiences :

- **Premièrement :** les DTD ne sont pas au format XML. Cela signifie qu'il est nécessaire d'utiliser un outil spécial pour manipuler un tel fichier, différent de celui utilisé pour l'édition du fichier XML.

- **Deuxièmement :** les DTD ne supportent pas les " espaces de nom ".En pratique, cela implique qu'il n'est pas possible d'importer des définitions de balises définies par ailleurs dans un fichier XML défini par une DTD.
- **Troisièmement :** le " typage " des données est extrêmement limité.
  Un fichier de DTD peut contenir principalement deux types de déclarations :

* **des déclarations d'éléments :** Indiquent les éléments pouvant être inclus dans un document et L'organisation du contenu de chaque élément (éléments fils ou texte).
* **des déclarations d'attributs :** Définissent les attributs pouvant être associés à un élément ainsi que leur type. On à un autre exemple concernant une DTD d'un document XML valide correspond (Figure 1.11, Figure 1.12).

```
1  <!ELEMENT document (paragraphe+)>
2  <!ATTLIST document type CDATA #REQUIRED>
3  <!ELEMENT paragraphe (#PCDATA)>
```

FIGURE 1.11 – Exemple de DTD.

```
1  <?xml version='1.0' encoding='iso-8859-1'?>
2  <!DOCTYPE document SYSTEM "document.dtd">
3  <document type='memo'>
4   <paragraphe>Xxxxxxxxx</paragraphe>
5   <paragraphe>Xxxxxxxxx</paragraphe>
6   <paragraphe>Xxxxxxxxx</paragraphe>
7  </document>
8
```

FIGURE 1.12 – Exemple de document XML valide.

1.11 Apports des schémas

Conçu pour pallier les précitées des DTD, XML Schéma propose des nouveautés en plus des fonctionnalités fournies par les DTD :[7]

* Le typage des données est introduit, ce qui permet la gestion de Booléens, d'entiers, d'intervalles de temps...il est même possible de créer de nouveaux types à partir de types existants.
* La notion d'héritage. Les éléments peuvent hériter du contenu et des attributs d'un autre élément.
* Le support des espaces de nom.
* Les indicateurs d'occurrences des éléments peuvent être tout nombre non négatif.
* Les schémas sont très facilement concevables par modules.

1.11.1 Le but d'un schéma

Le but d'un schéma est de définir une classe de documents XML. Il permet de décrire les autorisations d'imbrication et l'ordre d'apparition des éléments et de leurs attributs, tout comme une DTD.
Un premier point intéressant est qu'un fichier Schéma XML est un document XML. Cela permet à un tel document d'être manipulé de la même manière que n'importe quel autre fichier XML, et en particulier par une feuille de style XSL. Le vocabulaire de

XML Schéma est composé d'environ 30 éléments et attributs.
Ce vocabulaire est, a priori de manière bizarrement récursive et " auto-référent ", défini dans un Schéma. Mais il existe également une DTD.[7]

1.11.2 W3C XML Schéma

Les XML Schéma ont été proposés par le W3C pour permettre de dépasser les limites des DTD.[3] On notera en particulier :

* une syntaxe XML.
* l'extension de l'expression des règles d'organisation structurelle (héritage, réutilisation, etc...).
* l'ajout d'un langage de typage des éléments (particulièrement utile pour le format XML orientés données).

```
1  <!ELEMENT document (paragraphe+)>
2  <!ATTLIST document type CDATA #REQUIRED>
3  <!ELEMENT paragraphe (#PCDATA)>
```

FIGURE 1.13 – Exemple de DTD.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="document">
4      <xs:complexType>
5        <xs:sequence>
6          <xs:element maxOccurs="unbounded" ref="paragraphe"/>
7        </xs:sequence>
8        <xs:attribute name="type" use="required"/>
9      </xs:complexType>
10   </xs:element>
11   <xs:element name="paragraphe" type="xs:string"/>
12 </xs:schema>
```

FIGURE 1.14 – Exemple de W3C XML Schéma correspondant.

L'example de la (Figure 1.14) montre le XML schéma correspondant à la DTD de la(Figure 1.13).

1.12 Document bien formé et document valide

Un document XML doit respecter certaines règles grammaticales présentées dans la Spécification XML. Les documents XML doivent obéir à ces règles pour être considérés comme étant bien formés " Well formed document ".
Par conséquent, s'il y a non-respect d'une de ces règles, dans un document, il sera dit non bien formé. Tandis qu'un document XML sera dit valide s'il est bien formé et satisfait, en outre, aux règles d'une grammaire exprimée par une DTD ou un XML-schéma.[1]

1.13 Les objectifs du XML

L'objectif du XML est de faciliter les échanges de données entre les machines. A cèle s'ajoute un autre objectif important : décrire les données de manière aussi bien compréhensible par les hommes qui écrivent les documents XML que par les machines qui les exploitent. Un des points forts de XML est que son format est universellement admis par toutes les plates-formes.

En effet, quel que soit le système sur lequel on travaille, un document XML est créé initialement à partir d'un fichier texte (comme par fichier Bloc-notes ou WordPad sous Windows), que l'on sauvegarde sous l'extension XML. Dés qu'un tel fichier est sauvegardé, un document XML est systématiquement créé. Le XML se veut également compatible avec le web afin que les échanges de données puissent se faire facilement à travers le réseau internet.

Le XML se veut donc standardisés, simple, mais surtout extensible et configurable afin que n'importe quel type de données puisse être décrit.[9]

1.14 Conclusion

Dans ce chapitre, on a présenté des généralités sur les documents XML, Qui seront nécessaires pour la suite de ce travail. Le chapitre suivant sera consacré pour les différentes méthodes de classification et d'indexation.

CHAPITRE 2

CLASSIFICATION ET INDEXATION DES DOCUMENTS XML

2.1 Introduction

L'opération de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information (SRI). Les SRI classiques travaillant habituellement sur des collections de documents plats (texte), se focalisent uniquement sur les contenus, ignorant totalement la notion de structure et de liens pouvant éventuellement exister entre les documents. Mais avec l'évolution des corpus documentaires, notamment avec l'émergence des données semi-structurées sur le Web, la donnée a complètement changé.

Ce chapitre est organisé en deux parties, nous présentons dans la première partie la définition de la classification pour comprendre le thème de classification et les étapes principales d'une classification comme une procédure automatisée. Ensuite,

nous présentons la notion d'une distance entre les individus à classer car cette notion est essentielle pour la majorité des techniques de classification en citant quelques mesures souvent citées dans les littératures spécialisées, nous citons ces mesures selon le type de variable. Aussi nous avons consacré à présenter les différentes techniques de classification. Nous divisons ces techniques en plusieurs familles : les méthodes hiérarchiques et non hiérarchiques, et bayésienne.

Dans la deuxième partie, on s'intéresse de parler sur l'indexation des documents semi-structurés. Nous avons basé sur la structure qui offre un apport sémantique non négligeable, comme il est dit : "Ignorer la structure du document revient à ignorer sa sémantique ".

2.2 Notion de classe

Traditionnellement la notion de classe dans la classification documentaire a souvent été synonyme de thème, on parle alors de classification thématique. La tâche revient à partitionner un ensemble de documents autour de thématiques que l'on désigne communément par classes. Par exemple, de savoir si un document concerne le thème botanique, le thème philosophie ou bien le thème physique. Aujourd'hui la tâche de classification a évolué avec les besoins. Cette évolution a conduit à nouvelles problématiques pour lesquelles les classes ne correspondent plus nécessairement à des thématiques. A ce titre, la tâche de filtrage (text filtering) proposée par la campagne TREC (Text Retrieval Conference) permet de bien comprendre la problématique de classification. Par exemple, un système de filtrage est défini comme un processus extrait du flux d'informations (News, emails, actualités journalières), celles qui sont susceptibles d'intéresser un ou plusieurs utilisateurs ayant des besoins en information stables. TREC-4 propose d'évaluer un ensemble de classifieurs binaires indiquant si un nouveau document qui se présente est accepté ou pas.
Voici quelques exemples concrets de classification binaire.[8]

* Un logiciel qui détecte si un message est un Spam ou non.

* Un système qui sélectionne des informations pertinentes dans des flux de dépêches les envoie à différents organismes concernés.
* Un système d'agent s'intéresse aux flux de type Newsgroup et avertit un utilisateur dés qu'une nouvelle peut potentiellement l'intéresser.

2.3 classification

La classification est une méthode d'analyse des documents XML qui vise à regrouper en classes homogènes un ensemble d'observations.

Une définition formelle de la classification, qui puisse servir de base à un processus automatisé, amène a se poser les questions suivantes [10] :

- Comment les objets à classer sont-ils définis ?
- Comment définir la notion de ressemblance entre objets ?
- Qu'est-ce qu'une classe ?
- Comment sont structurées les classes ?
- Comment juger une classification par rapport à une autre ?

2.4 Le ranking

Contrairement à la classification définie dans la section précédente, qui émet une décision dure sur l'appartenance ou non d'un objet à une classe, le ranking (classement) ordonne (ou trie) ces objets par ordre de leur pertinence pour une classe donnée. On peut aussi dire que le ranking permet d'ordonner les classes par ordre de leur pertinence pour un objet donné. En d'autres termes, la tâche de ranking se base sur le calcul de la valeur d'une fonction de score définie par : Score : O * C $\rightarrow$ [0,1]. La valeur de Score(o, c) appartenant à l'intervalle [0,1] représente la pertinence de l'objet o pour la classe c. Voici deux exemples typiques d'application du ranking.[8]

- Le filtrage de documents en fixant un seuil de tolérance par rapport au score de ranking.
- Le classement (ranking) de pages Web par rapport à une thématique fixée par l'utilisateur.

Ainsi, la fonction Score (o, c) nous informe sur le degré de pertinence de l'objet o pour la classe c. Plus ce degré est proche de 1, très pertinent est l'objet pour cette classe. Le calcul de la fonction de score permet alors de classer les objets par ordre de pertinence afin de savoir si un objet est plus pertinent ou moins pertinent qu'un autre pour une classe donnée. Une bonne partie des systèmes probabilistes utilisent des algorithmes de classification automatique basée sur le calcul d'une fonction de score comme celle qu'on vient de définir ci dessus. Par exemple pour chaque document et chaque classe, on calcule la probabilité P(d/c) que le document d soit pertinent pour la classe c.

2.5 Le clustering

Le clustering (ou regroupement) est la tâche qui s'intéresse à trouver de manière automatique une organisation cohérente à un groupe d'objets. Elle correspond à la tâche de classification non-supervisée. Les problèmes liés au clustering ont été largement étudiés dans les domaines de l'apprentissage, en reconnaissance des formes et en analyse de documents. Une autre application importante de la tâche de clustering est en rapport avec la tâche de RD (Recherche Documentaire). Elle consiste à créer à priori sur un ensemble de documents un ensemble de clusters. Pour une requête donnée, le système de RD va tout d'abord chercher le cluster le plus pertinent pour la requête, puis trier les documents par ordre de pertinence dans ce cluster (ranking).

Cette méthode permet de limiter la complexité de la recherche dans les grandes bases de données et de renvoyer un sous ensemble de documents tous pertinents pour la requête puisque le choix du cluster est évalué à partir des caractéristiques communes des documents qui le composent. C'est une technique qui permet habituellement d'aug-

menter la précision d'un système de RD.

En effet, d'une part, elle réduit, de manière plus que significative, la complexité temporelle de la recherche sur de grandes bases documentaires, d'autre part, les documents retournés par le moteur de recherche, en réponse à une requête, sont tous pertinents, car ils présentent logiquement des caractéristique similaires, voire identiques pour cette requête (le cluster a été construit sur la base de caractéristiques communes aux documents qui s'y trouvent).[8]
On rencontre deux grandes familles de méthodes de clustering :

* Les systèmes fondés sur un calcul de similarité qui rassemblent un sous-ensemble de documents similaires dans un même cluster. Cette similarité peut par exemple ne concerner que la structure du document, tout comme elle peut impliquer un mélange de modèles (structure + contenu).
* Les systèmes probabilistes fondés sur un mélange de modèles (structure+contenu).

2.5.1 Les étapes d'une classification automatique

Les différentes étapes d'une classification automatique sont [11] :

1. Le choix des données.
2. Le calcul des ressemblances entre les individus à partir du tableau initial.
3. Le choix d'un algorithme de classification et exécution.
4. L'interprétation des résultats :
   - évaluation de la qualité de la classification.
   - description des classes obtenues.

2.5.2 similarité et dissimilarité

Une dissimilarité est une fonction d qui à tout couple (x1, x2) associe une valeur dans R+, et telle que :[11],[2]

✓ d(x1,x2) = d(x2,x1)≥ 0.

✓ d(x1,x2) = 0 → x1 = x2.

Autrement dit, moins les unités x1 et x2 se ressemblent, plus le score est élevé. Remarquons qu'une distance est une dissimilarité, puisque toute distance possède les deux propriétés précédentes ainsi que l'inégalité triangulaire. Toutes les distances connues, en particulier la distance euclidienne, sont donc des exemples de dissimilarité. A l'inverse, une autre possibilité consiste à mesurer la ressemblance entre observations à l'aide d'une similarité.

Une similarité est une fonction s qui à tout couple (x1, x2) associe une valeur dans R+, et telle que :

✓ s(x1,x2) = s(x2,x1) ≥ 0.

✓ s(x2,x1) ≥ s(x2,x1) .

Contrairement à la dissimilarité, plus les unités x1 et x2 se ressemblent plus le score est élevé.

- Les données pour une classification automatique sont de l'un des deux types suivant :
  - ✓ Un tableau individus-modalités et une distance permettant de calculer la similarité entre deux vecteurs individus.
  - ✓ Un tableau de distances entre individus.

2.5.3 Calcul de distance

Pour mesurer la similarité de deux vecteurs individus, on utilise en général la distance euclidienne usuelle. Notée d, elle est définie comme la racine de la somme des carrés des différences des coordonnées :

$$d(x, y) = \sqrt{\sum_{j=1}^{m} (x_j - y_j)^2}$$

m est le nombre de modalités. Certaines méthodes de classification tiennent compte

d'un système de poids défini sur les individus.[2]
On recherche une partition des individus optimisant un critère qui tend :

- à ne regrouper deux individus que s'ils sont très semblables.
- à ne séparer que des individus qui sont suffisamment différents.

Les parties de la partition ainsi constituée sont appelées classes ou clusters. La démarche consiste dans un premier temps à choisir :

- un critère mesurant la qualité d'une partition.
- un algorithme qui tend à trouver une partition qui optimise le critère.

2.5.4 Critère de qualité d'une partition

Une bonne méthode de classification produira des clusters de bonne qualité. Par définition un cluster est de bonne qualité si :

✓ la similarité intra-classe est très élevée.

✓ la similarité inter-classe est très faible.

Pour mesurer et interpréter la qualité d'une partition en k classes, on définit un ensemble de critères basés sur la décomposition de l'inertie totale (T) en inertie inter-classe (B) et inertie intra-classe (W) [2] :

$$\underbrace{\sum_{a=1}^{k} d^2(x_s, G)}_{T} = \underbrace{\sum_{i=1}^{k} \sum_{s \in c_i} d^2(x_s, G_i)}_{W} + \underbrace{\sum_{i=1}^{k} n_i d^2(G_i, G)}_{B}$$

d est la distance euclidienne, G le centre (la moyenne) de tous les objets xs, Gi le centre des xs objets appartenant à Ci et ni = card(Ci). Ainsi pour ce critère, chercher une partition qui maximise l'inertie inter-classe (donc qui tend à disperser au mieux les classes), revient à chercher une partition minimisant l'inertie intra-classe (donc qui tend à obtenir des classes les plus compactes).

2.6 Les méthodes de classification

On distingue principalement trois méthodes : les méthodes non hiérarchiques, les méthodes hiérarchiques et la méthode bayésienne.

2.6.1 Les méthodes non hiérarchiques

La méthode des non hiérarchiques permet de construire par itération, à partir d'une famille absolument quelconque de noyaux, une partition en k classes. Un noyau se définit comme un ensemble de p points proches les uns des autres. Chacune des classes comprend les éléments qui sont les plus proches d'un des noyaux. A partir de cette partition, il est possible de définir une nouvelle famille de noyaux en associant à chaque classe de la partition l'ensemble de p points qui en est le plus proche. Cette construction itérative revêt également une part d'arbitraire puisque la partition obtenue dépend du choix initial des noyaux. Pour compenser dans une certaine mesure ce défaut, il est recommandé d'appliquer la méthode plusieurs fois de suite, en partant à chaque fois d'une famille différente de noyaux tirés au hasard.[11]

- **L'algorithme K-means**

  L'algorithme K-means, créé par Macqueen en 1967, est l'algorithme de clustering le plus connu et le plus utilisé car il s'avère être trés simple à mettre en œuvre et efficace. Il suit une procédure simple de classification d'un ensemble d'objets en un certain nombre K de clusters, K fixé a priori.

  Dans cet algorithme, chaque cluster est caractérisé par son centre (prototype) qui se trouve être la moyenne des éléments composant le cluster.[2]

  L'algorithme K-means s'exécute en 4 étapes :

  ► Choisir aléatoirement K objets qui forment ainsi les K clusters initiaux.

  ► Affecter les objets à un cluster. Pour chaque objet x, le prototype qui lui est assigné est celui qui est le plus proche de l'objet, selon une mesure de distance, (habituellement la distance euclidienne est utilisée).

  ► Une fois tous les objets placés, recalculer les centres des K clusters.

► Répéter les étapes 2 et 3 jusqu'à ce que plus aucune réaffectation ne soit faite. Même si l'algorithme termine toujours, on n'est pas assuré d'obtenir la solution optimale

En effet, l'algorithme est très sensible au choix aléatoire des K centres initiaux. C'est pourquoi, on utilise souvent l'algorithme des K-means de nombreuses fois sur un même ensemble de données pour essayer de minimiser cet effet tout en sachant que des centres initiaux les plus espacés possibles donnent de meilleurs résultats.[2]

▲ **Les avantages de k-means**

Nous pouvons citer quelques avantages des méthodes k-means par :

* Simple.
* Compréhensible.
* Applicables à des données de grande taille.[10]

▼ **Les inconvénients de k-means**

* Le nombre des classes doit être fixé au départ, dans le pratique, on ne connait pas le nombre optimale de clusters.
* Ne détectent pas les données bruitées
* Le résultat dépend du tirage initial des centres des classes.[10]

Ces méthodes donnent la plus part de temps une partition localement optimale, il est donc conseillé d'effectuer plusieurs exécutions et comparer les différentes résultats obtenues. Une suggestion consiste à appliquer la méthode des k-means dans une première étape à plusieurs sous-ensembles de données extraits de l'ensemble total, et la meilleure partition obtenue fournit les centres à utiliser au départ de l'algorithme appliqué à l'ensemble total des données.

- **La méthodes k- medoïds**

Dans des méthodes de k-medoïds une classe est représentée par un de ses individus (médoïde). C'est une méthode itérative combinant la réaffectation des individus dans des classes avec une intervention des médoïdes et des autres individus. C'est une méthode simple parce qu'elle couvre n'importe quel type de variables. Quand des medoids sont choisis, des classes sont définies comme sous-

ensembles des individus près des médoïdes les plus proches par rapport à une mesure de distance choisie. Il est alors plus judicieux de choisir comme centre de groupe un individu présent dans le groupe et non un individu calculé. La médoïde d'un groupe est l'individu possédant la dissimilarité moyenne la plus faible d'avec les autres individus du groupe.[10] Nous présentons ici quelques versions des méthodes de k-medoïds :

- **La version PAM**

  K- medoïds ou "Partition Around medoïds" (PAM) désigne que chaque classe est représentée par l'un de ces individus (médoïdes) Cette version a un ensemble de principes on les cité au dessue :

  1. Choisir un ensemble de médoïdes.
  2. Affecter chaque individu au médoïde le plus proche.
  3. Itérativement remplacer chaque médoïde par un autre si cela permet de diminuer la distance globale.

Données : k le nombre maximum de classes désiré.
Début

- Sélectionner aléatoirement k représentants M1, M2, . . ., MK
- Calculer le coût global CG_{ih} d'une permutation pour chaque paire (M_i, O_h) ou M_i est le représentant d'une classe et Oh un autre objet

  $$(CG_{ih} = \sum_j dist(i,h) - dist(i,j)$$
- Sélectionner la paire (M_i, O_h) pour laquelle CG_{ih} est minimal
- Si le coût différentiel CG_{ih} pour la paire sélectionnée est négatif, alors échanger les rôles de M_i et Oh et retourner en (2)
- Retourner les classes correspond aux représentants M1, M2, . . ., MK

Fin.

FIGURE 2.1 – Le déroulement de l'algorithme PAM.

L'avantage de l'algorithme PAM est qu'il est plus robuste que les méthodes k-means en présence de bruit et l'inconvénient vient de la complexité des calculs ($O[(k-n)^2]$) pour chaque itération ce qui est plus coûteux en cas de k et n assez grand. Ce qui nous conduit à dire que cet algorithme est intéressant et efficace en

cas de données de petite taille. D'ou la proposition d'autres algorithmes comme CLARA pour classer les données de taille moyenne peut être importante.

► **La version CLARA**

Cet algorithme a été introduit par Kaufman et Rousseuw en 1990 pour traiter des données de taille moyenne ou grande, elle effectue une recherche locale des représentants en opérant sur plusieurs échantillons de données de taille s extraite de l'échantillon total. Ensuite l'algorithme de PAM sera appliqué à chacun d'entre eux et le résultat obtenu est le meilleur parmi les différents résultats. Les points de l'échantillon total sont alors affectés aux représentants résultants. L'expérience montre qu'en pratique 5 échantillons, de taille 40 + 2k points, suffisent pour obtenir des résultats satisfaisants.[10]

Données : k le nombre maximum de classes désiré.
Début Pour i = 1 à 5
Répéter
- Tirer un échantillon de 40 + 2.k objets
- Appliquer PAM sur l'échantillon
- Calculer la qualité (Comme dans l'algorithme PAM) de la partition engendrée sur l'ensemble des données
- Si la qualité trouvée est la plus faible déjà déterminée
- Mémoriser les représentants déterminés à l'étape 3.

FIN

FIGURE 2.2 – Le déroulement de l'algorithme CLARA.

▼ **Les inconvénients de cet algorithme**

Cet algorithme à des inconvénients nous avons les cités au-dessus :

* Paramètres d'échantillonnage choisis expérimentalement.
* Exploration réduite de l'espace de recherche.

2.6.2 classification hiérarchique

La méthode hiérarchique génère une partition de l'espace de données (documents XML), mais aussi une succession de partitions des données. Celles-ci sont souvent

représentées sous la forme d'un dendrogramme.. Un dendrogramme est un arbre de partitions successives de l'espace de données (figure 2.3). Selon que l'on parcourt le dendrogramme de haut en bas ou de bas en haut, la méthode sera dite descendante (division) ou ascendante (agglomération).

Alors Il y a deux grandes méthodes de hiérarchiques : La Classification Ascendante Hiérarchique CAH et la Classification Descendante Hiérarchique CDH.[12]

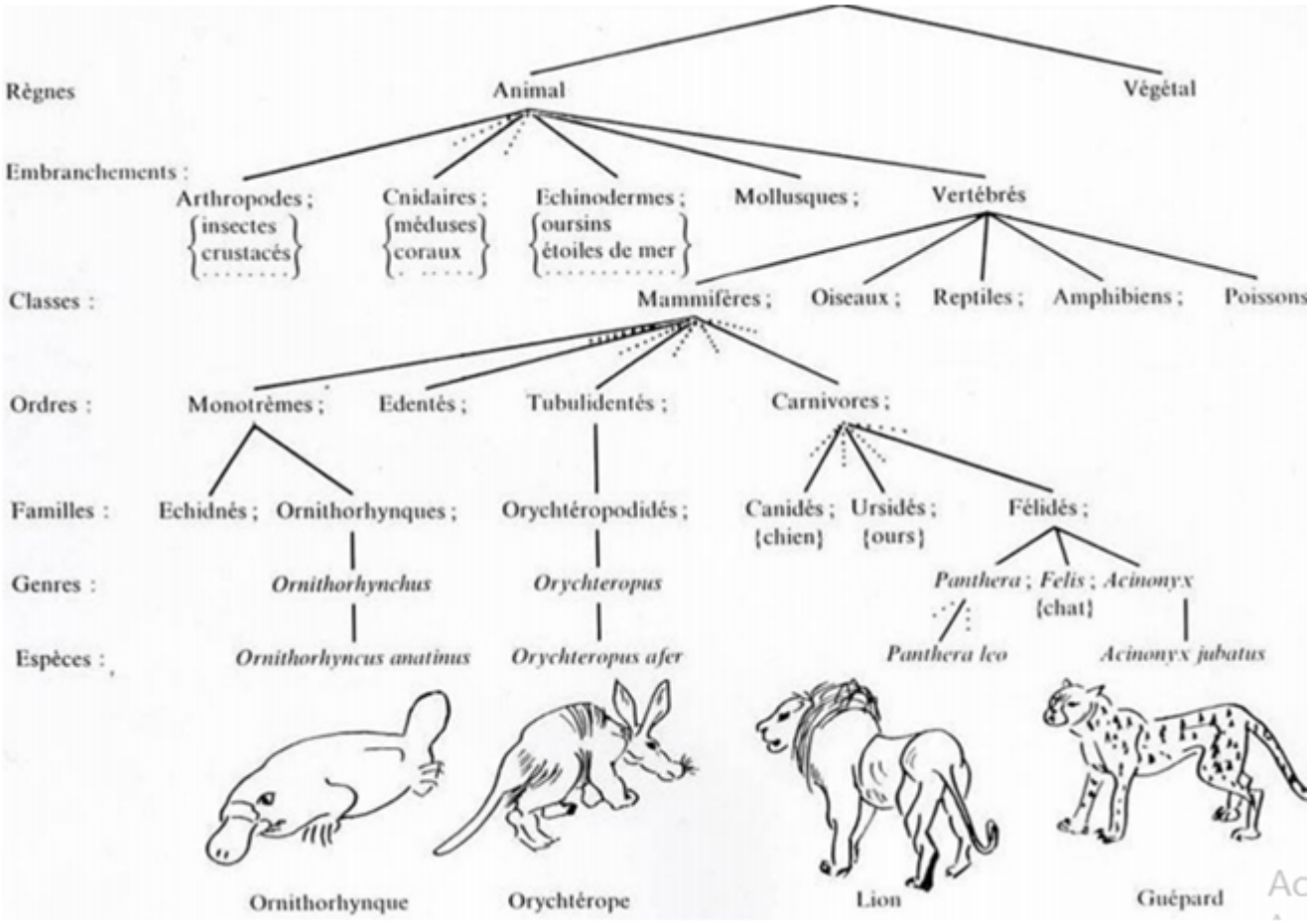

FIGURE 2.3 – Exemple de hiérarchie (le règne animal).

- **Classification ascendante hiérarchique**
  La classification ascendante hiérarchique (CAH) est une méthode de classification itérative dont le principe est simple.

  1. On commence par calculer la dissimilarité entre les N objets.

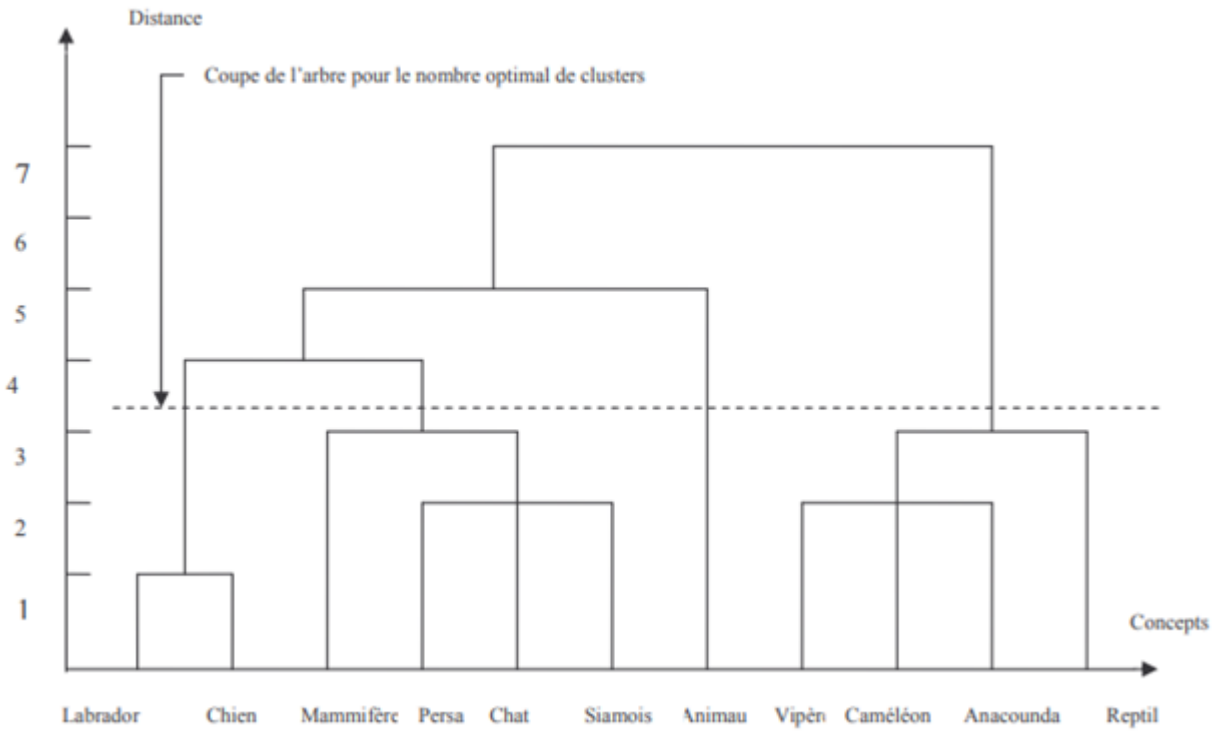

FIGURE 2.4 – Exemple de dendrogramme

2. Puis on regroupe les deux objets dont le regroupement minimise un critère d'agrégation donné, créant ainsi une classe comprenant ces deux objets.
3. On calcule ensuite la dissimilarité entre cette classe et les N-2 autres objets en utilisant le critère d'agrégation. Puis on regroupe les deux objets ou classes d'objets dont le regroupement minimise le critère d'agrégation.

On continue ainsi jusqu'à ce que tous les objets soient regroupés.

✓ **Les critères d'agrégation**

De nombreux critères d'agrégation ont été proposés les plus connus sont :

∗ **Le critère du saut minimal :**

La distance entre deux classes $_{C_1}$ et $_{C_2}$ est définie par la plus courte distance séparant un individu de $_{C_1}$ et un individu de $_{C_2}$.

$$D(C_1, C_2) = \min(d(x, y), x \in C_1, y \in C_2)$$

* **Le critère du saut maximal :**

  La distance entre deux classes $_{C_1}$ et $_{C_2}$ est définie par la plus grande distance séparant un individu de $_{C_1}$ et un individu de $_{C_2}$.

$$D(C_1, C_2) = \max(d(x, y), x \in C_1, y \in C_2)$$

* **Le critère de la moyenne :**

  Ce critère consiste à calculer la distance moyenne entre tous les éléments de $_{C_1}$ et tous les éléments de $_{C_2}$.

$$D(C_1, C_2) = \frac{1}{n_{C_1} n_{C_2}} \sum_{x \in C_1} \sum_{y \in C_2} d(x, y)$$

  Avec :

  n_{C_1} : Le cardinal de $_{C_1}$.

  n_{C_2} : Le cardinal de $_{C_2}$.

* **Le critère de Ward :**

  Ce critère ne s'applique que si on est muni d'un espace euclidien. La dissimilarité ρ entre deux individus doit être égale à la moitié du carré de la distance euclidienne d. Le critère de Ward consiste à choisir à chaque étape le regroupement de classes tel que l'augmentation de l'inertie intra-classe soit minimal.

$$D(C_1, C_2) = \frac{n_{C_1} n_{C_2}}{n_{C_1} + n_{C_2}} d^2(g_{C_1}, g_{C_2})$$

  Avec :

  g_{C_1} : centre de gravité de $_{C_1}$. g_{C_2} : centre de gravité de $_{C_2}$.

* **Le critère des centres de gravité :**

  La distance entre deux classes $_{C_1}$ et $_{C_2}$ est définie par la distance entre leurs centres de gravité.

$$D(C_1, C_2) = d(g_{C_1}, g_{C_2})$$

La difficulté du choix du critère d'agrégation réside dans le fait que ces cri-

tères peuvent déboucher sur des résultats différents. Selon les plus parts des références le critère le plus couramment utilisé est celui du Ward.
Nous présentons dans la suite quelques algorithmes basés sur le principe de classification ascendante hiérarchique.[10]

1. **Algorithme CURE :**

   CURE (Clustering Using REpresentative) a été proposé par Guha et al, cet algorithme utilise un échantillon représentatif de l'échantillon total pour réduire la complexité temporelle des calculs. Cet échantillon sera divisé en sous-ensembles qui sont regroupés en sous-classes. Les sous-classes seront agrégées hiérarchiquement en utilisant la distance entre deux sous-classes $_{C_1}$ et $_{C_2}$, la plus petite distance entre un représentant de $_{C_1}$ et un représentant de $_{C_2}$ jusqu'à obtenir k de classes demandées.[10]

   Les détails de cette méthode sont décrits dans l'algorithme suivant :

Données : k le nombre maximum de classes désiré.
Début
(1) Extraire un échantillon $\mathcal{I}_s$ de taille s de l'ensemble $\mathcal{I}$ d'individus
(2) Diviser l'échantillon $\mathcal{I}_s$ en p sous ensemble de taille s/p
(3) Déterminer une partition partielle de chaque sous-ensemble en s/pq sous-classes avec $q > 1$
(4) Éliminer les sous-classes d'effectif faible.
(5) Déterminer dans chaque sous-classes un ensemble de c points (la caractérisant points bien répartis dans la sous-classe).
(6) Agréger de façon hiérarchique les sous-classes $d(C_1, C_2) = min[d(x, y); \forall x$ représentant de C_1 et $\forall y$ représentant de $C_2]$
(7) Arrêter la procédure d'agrégation quand on obtient k classes.
(8) Classer l'ensemble $\mathcal{I}$ total en utilisant les c points représentant les k classes obtenues après l'agrégation. Chaque objet est affecté à la classe possédant le représentant qui lui est le plus proche.
FIN

FIGURE 2.5 – L'algorithme CURE.

2. **Algorithme BIRCH :**

BIRCH (Balanced Itérative Reducing and Clustering using Hiérarchies) a été développé par Zhang et al en 1996. C'est un algorithme qui travaille efficacement sur de gros jeux de données. L'idée principale du BIRCH est d'effectuer une classification sur un résumé compact des données, au lieu des données originales. C'est pourquoi il peut traiter un grand volume de données en utilisant une mémoire limitée. Il est incrémental, i.e. il a besoin d'un seul balayage du jeu de données. Il essaie de minimiser le coût d'entrée/sortie en organisant les données traitées en une structure d'arbre équilibré avec une taille limitée. La communauté du domaine de classification trouve que BIRCH est l'un des meilleurs algorithmes qui peuvent traiter de gros jeux de données. Initialement, BIRCH a été crée pour traiter des données numériques, une extension a été proposée pour traiter des données de type catégoriel. Nous en parlerons au niveau algorithmique sans aborder en détail la mesure (le calcul) utilisée dans l'algorithme.[10]
Deux phases principales caractérisent cette méthode :

* **Phase (1) :**
  Construction en mémoire d'un arbre de sous-classes et de leurs vecteurs caractéristiques (CF-tree) résumant les données volumineuses sur disque.
* **Phase (2) :**
  application en mémoire d'une technique classique de clustering sur les sous-classes les plus fines de l'arbre déterminé en (Figure 2.1)
  Le vecteur caractéristique d'une classe CF peut prendre les valeurs N, LS, SS.
  N : C'est le nombre d'objets $_{O_i}$ dans la classe.
  LS $=\sum X_i$
  SS $=\sum X_i^2$
  X_i **:** Le vecteur des variables décrivant l'objet $_{O_i}$.

Le vecteur caractéristique d'une classe C composée de deux sous-classes $_{C_1}$

et $_{C_2}$ est la somme des vecteurs caractéristiques de $_{C_1}$ et $_{C_2}$ permet un calcul incrémental des vecteurs caractéristiques des classes construites. A partir des vecteurs caractéristiques d'un ensemble de classes, on peut déterminer sans accéder aux données :

* Les centres de gravités et diam êtres des classes.
* Les distances inter-classes moyennes et distances intra-classes classiques (moyennes ou entre centres de gravité) Ce qui permet de réaliser la phase (2) à partir des vecteurs caractéristiques des classes, sans accéder aux données de base.[10]

- **Classification descendante hiérarchique**

  Les méthodes de classification descendante hiérarchique partent d'un ensemble d'individus I et construisent, de manière itérative une partition de l'ensemble d'individus. 40 A l'inverse de la classification ascendante hiérarchique, à chaque étape de l'algorithme il y a deux processus à faire :

  1. Chercher une classe à scinder.
  2. Choisir un mode d'affectation des objets aux sous-classes.

  Parmi les algorithmes les plus anciens, l'algorithme de Williams et Lambert, divise la classe la plus grande en deux classes. Hubert a proposé de diviser la classe de plus grand diamètre. Aucun des deux n'a pas justifié son choix de division. Cette méthode de classification construit sa hiérarchie dans le sens inverse, en commençant par une grande classe contenant tous les objets. A chaque étape, elle divise une classe en deux classes plus petites jusqu'à ce que toutes les classes ne contiennent qu'un seul individu. Ceci veut dire que pour n individus, la hiérarchie est construite en n-1 étapes. Dans la première étape, les données sont divisées en deux classes au moyen des dissimilarités. Dans chacune des étapes suivantes, la classe avec le diamètre le plus grand se divise de la même façon. Après n-1 divisions, tous les individus sont bien séparés. La dissimilarité moyenne entre l'individu x qui appartint à la classe C qui contient

n individus et tous les autres individus de la classe C est définie par [10] :

$$d_x = \frac{1}{n-1} \sum_{x \in C, y \neq x} d(x, y)$$

Nous présentons dans la suite quelques algorithmes basés sur le principe de classification descendante hiérarchique.

► **Algorithme Williams et Lambert :**

L'algorithme de Williams et Lambert (1959) que nous décrivons ici, est particulièrement rudimentaire. Il n'est applicable que sur des variables qualitatives, il sélectionne l'une des variables pour servir de critère d'affectation : tous les individus présentant, pour cette variable, la même modalité sont rangés dans la même classe (si les variables sont à plus de deux modalités le nœud correspondant aura plus de deux branches). La variable retenue est celle qui, dans la classe C à scinder, est la plus reliée à toutes les autres. Comme il s'agit de variables qualitatives la relation est mesurée par le X^2 de contingence. On calcule donc les X^2 de contingence de toutes les variables prises deux à deux, et l'on retient celle pour laquelle la somme de ses X^2 est maximum. La méthode de Williams et Lambert est bien adaptée au traitement de tableaux présentant un grand nombre d'observations et peu de variables qualitatives. La table des X^2 de contingence entre variables est alors rapide à obtenir, par comparaison au temps qu'il faudrait pour calculer, par exemple, la matrice de Jaccard relative aux individus.

La facilité à interpréter le résultat est une propriété parmi les avantages de cet algorithme.

Par contre, il possède les inconvénients suivants :

* les résultats sont en général grossiers. Cela tient au fait que les groupes d'individus se définissent rarement par leurs réponses strictement identiques à une série de questions mais bien plutôt par un pourcentage élevé de réponses semblables sur l'ensemble des questions.

* Notons encore que les niveaux des nœuds de la hiérarchie ne sont plus définis que par l'ordre dans lequel ils apparaissent et il n'est pas naturel de leur associer un indice montrant la cohésion du groupe d'objets associés à ce nœud. On pourrait imaginer un programme semblable travaillant sur des variables quantitatives. Il y faudrait ajouter une étape supplémentaire : une fois choisie la variable de scission, il faudrait choisir une valeur-seuil pour cette variable ; en dessous de ce seuil les objets seraient rangés dans l'une des sous-classes, au dessus de ce seuil les objets seraient affectés à l'autre sous-classe. Toutefois une telle procédure présenterait les mêmes avantages et les mêmes inconvénients que celle de Williams et Lambert.

Données : k le nombre maximum de classes désiré.
Début
(1) Extraire un échantillon $\mathcal{I}_s$ de taille s de l'ensemble $\mathcal{I}$ d'individus
(2) Diviser l'échantillon $\mathcal{I}_s$ en p sous ensemble de taille s/p
(3) Déterminer une partition partielle de chaque sous-ensemble en s/pq sous-classes avec $q > 1$
(4) Éliminer les sous-classes d'effectif faible.
(5) Déterminer dans chaque sous-classes un ensemble de c points (la caractérisant points bien répartis dans la sous-classe).

(6) Agréger de façon hiérarchique les sous-classes $d(C_1, C_2) = min[d(x, y); \forall x$ représentant de C_1 et $\forall y$ représentant de $C_2]$
(7) Arrêter la procédure d'agrégation quand on obtient k classes.
(8) Classer l'ensemble $\mathcal{I}$ total en utilisant les c points représentant les k classes obtenues après l'agrégation. Chaque objet est affecté à la classe possédant le représentant qui lui est le plus proche.
FIN

FIGURE 2.6 – Algorithme Williams et Lambert.

► **Algorithme TSVQ :**

TSVQ (Tree Structured Vector Quantization) a été proposé par Gersho et Gray (1992). Cet algorithme utilise l'algorithme K-moyennes (k=2) pour le partitionnement. Utilise la somme des distances par rapport au centroïde (au

lieu de la distance moyenne).

- **Les Avantages de classification descendante hiérarchique(CDH) :**
  Par rapport à la plus part des algorithmes en classification automatique, l'algorithme de classification descendante hiérarchique ne nécessite pas l'utilisation d'un seuil arbitraire pour la formation des classes qui peut éventuellement mener la recherche d'une partition dans une direction non réaliste. Si l'algorithme d'échange ne privilégie que les aspects locaux, il est initialisé avec une partition liée par des relations de filiations avec les partitions précédemment obtenues. Cela donne à l'algorithme un certain aspect global.[10]

- **Les inconvénients de classification descendante hiérarchique(CDH) :**
  Les résultats sont en général grossiers, les niveaux des nœuds de la hiérarchie ne sont plus définis que par l'ordre dans lequel ils apparaissent.[10]

- **Les avantages des méthodes hiérarchiques :**
  - Flexibilité concernant le niveau de la granularité.
  - Facilité de manipuler toute forme de similitude ou de distance.
  - Applicabilité à tout type d'attribut.
- **Les inconvénients des méthodes hiérarchiques :**
  - La difficulté de choisir la droite arrêtant des critères.
  - La plupart des algorithmes hiérarchiques ne révisent pas des classes (d'intermédiaire) une fois ils sont construits.

2.6.3 Classification Bayésienne

La classification bayésienne naïve ("Naïve Bayes Classification"), est l'une des techniques d'apprentissage supervisé (doit être connu a l'avance les classes) les plus utilisées dans les domaines de classification de documents et recherche d'informations (IR).est un algorithme simple ,robuste qui se base sur le théorème de Bayes pour cela nommée bayésienne, et naïve Parce que pour utiliser cet algorithme nous supposons que les variables explicatives sont indépendantes, ce qui dans la réalité est faux. Mais

on l'accepte quand même pour pouvoir l'utiliser.[13]

Le théorème de Bayes c'est comme suit :

$$P\left(\frac{A}{B}\right) = P\left(\frac{B}{A}\right).P(A)/P(B)$$

2.7 Le processus de recherche d'information

Le schéma représente les différentes étapes de processus en U pour la recherche d'information (Figure 2.3).ce processus est composé principalement des documents et requête qui représente le contenu d'information, aussi l'opération pour traiter la requête puis sélectionner le document à partir du besoin d'utilisateur.

2.7.1 La notion de document et de requête

Le document représente le conteneur élémentaire d'information, nous appelons document toute unité qui peut constituer une réponse à un besoin en information exprimé par un utilisateur. Dans notre travail les documents sont des documents XML. De la même manière Que Java a mis à notre disposition un langage indépendant de toute plate-forme, nous avons également besoin de la même indépendance lorsqu'il s'agit de l'échange de données. XML nous en offre le moyen.[14]

- **Requête :**
  Une requête constitue l'expression du besoin en informations de plusieurs systèmes utilisent des langages différents pour décrire la requête : par une liste de mots clés, en langage naturel, en langage booléen, en langage graphique.[14]

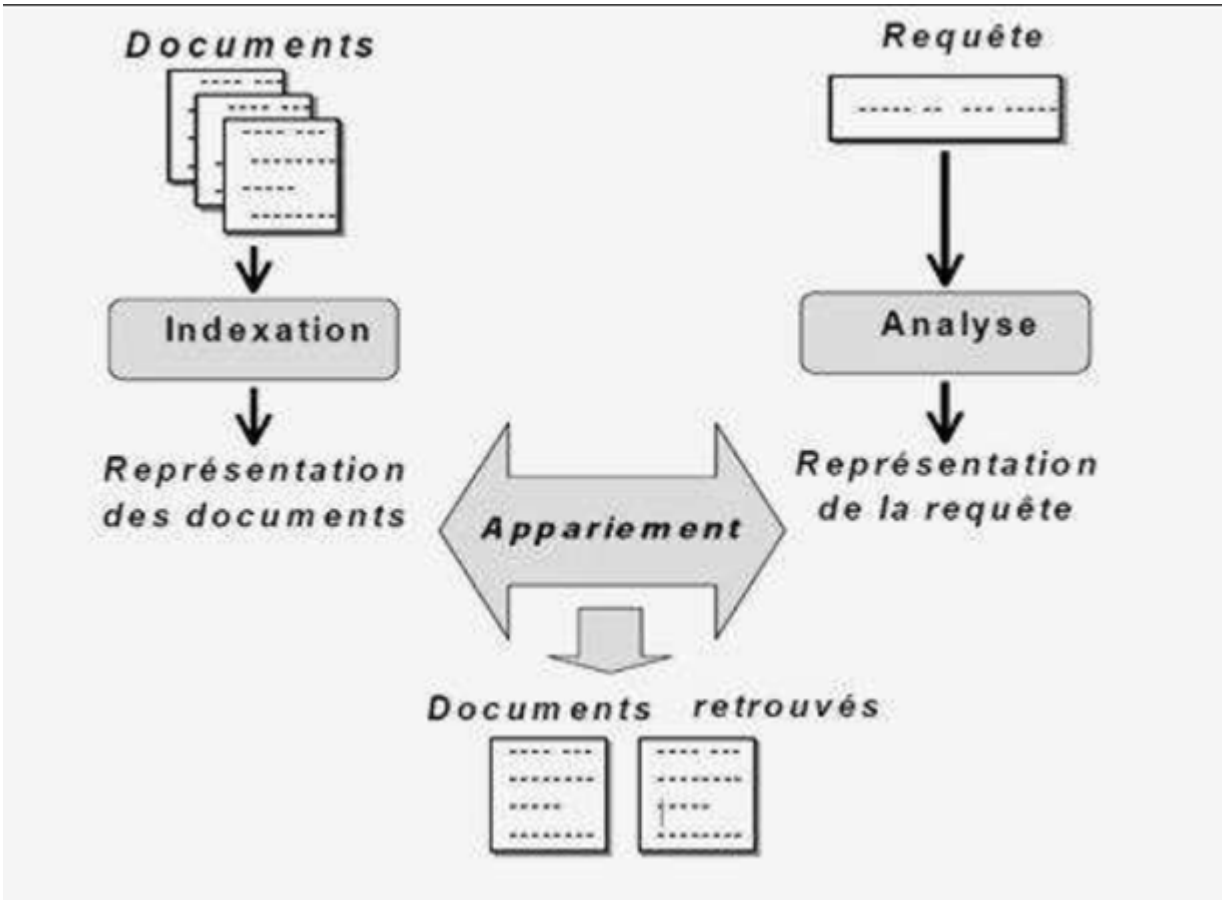

FIGURE 2.7 – processus en U pour la recherche d'information.

2.7.2 Principales phases du processus de recherche d'information

L'objectif fondamental d'un processus de RI est de sélectionner les documents XML "les plus proches" du besoin en information de l'utilisateur décrit par une requête. Ceci induit deux principales phases dans le déroulement du processus : indexation et appariement requête/documents.[14]

2.7.3 Le processus d'indexation

L´étape d'indexation consiste à analyser, chaque document de la collection afin de créer un ensemble de mots-clés .Ces mots-clés seront plus facilement exploitables par le système lors du processus ultérieur de recherche. L'indexation permet ainsi de créer une

représentation des documents dans le système. Son objectif est de trouver les concepts les plus importants du document (ou de la requête), qui formeront le descripteur du document. [15]

L'indexation peut être :

- **Manuelle :**
  chaque document est analysé par un spécialiste du domaine ou par un documentaliste.
- **Automatique :**
  le processus d'indexation est entièrement informatisé.
- **Semi-automatique :**
  le choix final revient au spécialiste ou au documentaliste, qui intervient souvent pour choisir d'autres termes significatifs.
  - ► L'indexation manuelle permet d'assurer une meilleure pertinence dans les réponses apportées par le SRI. Elle présente toutefois plusieurs inconvénients : deux indexeurs différent peuvent présenter des termes différents pour caractériser un même document, et un indexeur à deux moments différents peut présenter deux termes distincts pour représenter le même concept. De plus, le temps nécessaire à sa réalisation est très important.
  - ► Dans le cas d'une indexation semi-automatique, les indexeurs utilisent un thesaurus ou une base terminologique, qui est une liste organisée de descripteurs (mots clés) obéissant à des règles terminologiques propres, et reliés entre eux par des relations sémantiques.
  - ► Enfin, l'indexation automatique, regroupe un ensemble de traitements automatisés sur un document.

  On distingue :
  L'extraction automatique des mots des documents, l´élimination des mots vides, la lemmatisation (radicalisation ou normalisation), le repérage de groupes de mots, la pondération des mots, la création de l'index.
  Le volume des collections c'est un paramètre qui définie le choix et l'intérêt d'une

méthode par rapport aux autres. Le résultat de l'étude montre que les avantages et inconvénients de chacune des approches s'équilibrent : le choix d'une méthode Doit être fait en fonction du domaine, de la collection et de l'application considérée.

En bref, l'indexation constitue la description du document et de son contenu en vue de Faciliter son exploitation. On distingue à ce titre deux catégories d'indexation [8] :

* **L'indexation par type :**
  on parle aussi à l'indexation structurelle De documents. A partir du rôle de métadonnée qui joué par la structure des documents semi-structuré de type XML. Ces documents se prêtent à cette catégorie d'indexation.
* **L'indexation par concepts ou mots-clés :**
  on parle ici d'indexation statistique, qui vise plutôt le contenu du document pour facilité les opérations des recherches. Dans cette catégorie le système sélectionne les termes dans une liste de mots liés par des équivalences.
  Dans le cadre de documents structuré ou semi-structuré on trouve deux catégorie d'indexation : l'indexation basé sur la structure et autre sur le contenu. Mais dans ce travail, nous nous intéressons juste à l'indexation de document XML sur la base de leur structure.
  La clé principale qui permet l'indexation structurelle des documents XML c'est la structure. La structure permet ainsi de détecter la classe appropriée d'un Groupe de document partageant la même structure ou ayant des structures similaires.
  Par conséquent, l'accès à un ensemble de documents d'une même classe, se fera sur la base d'une structure commune représentant l'ensemble des documents de cette classe.
  Pour accéder aux documents XML, il va donc falloir y accéder par une requête qui permettra au système de sélectionner au préalable la classe appropriée. L'idée est que, si des documents XML partagent des structures similaires, ils sont plus à même de correspondre à La partie structurelle d'une même requête.

Ainsi, nous citons certaines approches d'indexation structurelle ,et les caractéristiques qui les distinguent. L'information structurelle est effectivement exploitée pour représenter structurellement des collections de documents XML. Nous nous intéressons, plus particulièrement dans ce travail, à l'indexation permettant d'établir une classification structurelle de ces documents. La structure étant la clé principale qui permettra alors d'indexer structurellement les documents XML et donc d'y accéder plus facilement lors de sessions d'interrogation ou de recherche d'information.

2.7.4 Les traitements d'une indexation automatique

L'indexation automatique, que nous décrivons en détail dans cette partie, regroupe un ensemble de traitements automatisés sur un document.[15]

- **L'analyse lexicale :**

  L'analyse lexicale est le processus qui permet de convertir le texte d'un Document en un ensemble de termes. Un terme est une unité lexicale ou un radical [69]. Ce traitement permet de reconnaitre les espaces de séparation des mots, des chiffres, les ponctuations, etc.

- **L'élimination des mots vides :**

  Un des problèmes majeurs de l'indexation consiste à extraire les termes Significatifs et à éviter les mots vides (pronoms personnels, prépositions...). Les mots vides peuvent aussi être des mots athématiques (les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple contenir, appartenir, etc).

  On distingue deux techniques pour éliminer les mots vides :

  * L'utilisation d'une liste de mots vides (aussi appelée anti-dictionnaire).
  * L'élimination des mots dépassant un certain nombre d'occurrences dans la collection. Même si l'élimination des mots vides a l'avantage évident de réduire le nombre de termes d'indexation, elle peut cependant réduire le taux de rappel, c'est à dire la proportion de documents pertinents renvoyés par le système par rapport à l'ensemble des documents pertinents.

► **La lemmatisation :**

Un mot donné peut avoir différentes formes dans un texte, mais leur sens reste le même ou très similaire. On peut par exemple citer économie, économiquement, économétrie, économétrique, etc. Il n'est pas forcément nécessaire d'indexer tous ces mots alors qu'un seul suffirait à représenter le concept véhiculé. Pour résoudre le problème, une substitution des termes par leur racine, ou lemme, est utilisée.

Frakes et Baeza-yates distinguent cinq types stratégiques de lemmatisation : la table de consultation (dictionnaire), l'élimination des affixes, la troncature, les variétés de successeurs ou encore la méthode des n-gramme. Cette phase de passage à la forme canonique n'est pas obligatoire. Elle présente le principal avantage d'indexer par exemple le mot "camions" et le mot "camion" de la même façon ("camion"), ce qui évite à l'utilisateur de devoir entrer les formes de pluriel des noms ou les formes conjuguées des verbes lors de sa recherche.

Cependant, dans certains cas, le passage à la forme canonique supprime la sémantique originale du mot. Par exemple, la forme conjuguée " portera " du verbe " porter " sera indexée sous "porte ", de la même façon que le mot " portes ".

Ainsi, lorsque l'utilisateur formulera une requête avec le verbe " porter ", il aura très certainement, parmi la liste des documents résultats, des documents non pertinents relatifs au nom " porte ". Si la lemmatisation a pour but d'augmenter le rappel, la précision (c'est-à-dire la proportion de documents pertinents par rapport au nombre de documents renvoyés par le système) en fait souvent les frais.

Pour solutionner ce problème, C.J. Crouch propose une méthode en deux temps, dont les résultats s'avèrent encourageants :

* Une première recherche est effectuée, en utilisant une lemmatisation des mots.
* Les documents sont ensuite réordonnés en fonction de la présence des Termes non-lemmatisés de la requête dans leur contenu.

► **La pondération des termes :**

La pondération des termes permet de mesurer l'importance d'un terme dans un document. Cette importance est souvent calculée à partir de considérations et

interprétations statistiques (ou parfois linguistiques). L'objectif est de trouver les termes qui représentent le mieux le contenu d'un document. Si on dresse une liste de l'ensemble des mots différents d'un texte quelconque classés par ordre de fréquences décroissantes, on constate que la fréquence d'un mot est inversement proportionnelle à son rang de classement dans la liste. Cette constatation est énoncée formellement par la loi de Zipf [15] :

Rang*fréquence = constante.

La loi de distributions des termes suit alors la courbe présentée sur la (figure 8). Zipf explique la courbe hyperbolique de la distribution des termes Parce qu'il appelle le principe du moindre effort : il considère qu'il est plus facile pour un auteur d'un document de répéter certains termes que d'en utiliser de nouveaux. La relation entre la fréquence et le rang des termes permet de sélectionner les termes représentatifs d'un document : on élimine respectivement les termes de fréquences très élevées car ils ne sont pas représentatifs du document (on peut par exemple citer les mots outils), et les termes de fréquences très faibles (ce qui permet d'éliminer les fautes de frappes et les néologismes). Ce processus est illustré sur la (figure 2.8). En utilisant cette approche, le nombre de termes faisant partie de l'index d'une collection peut être réduit considérablement.

A partir de ces constatations, des techniques de pondération ont vu le jour. La plupart de ces techniques sont basées sur les facteurs TF et idf, qui permettent de combiner les pondérations locale et globale d'un terme :

* **TF (Term Frequency) :**
  cette mesure est proportionnelle à la fréquence du terme dans le document (pondération locale). Elle peut être utilisée telle quelle ou selon plusieurs déclinaisons (Log (Tf), présence/absence,...).
* **idf (Inverse of Document Frequency) :**
  ce facteur mesure l'importance d'un terme dans toute la collection (pondération globale). Un terme qui apparait souvent dans la base documentaire ne doit pas avoir le même impact qu'un terme moins fréquent. Il est généralement exprimé comme suit : log (N/df), où df est le nombre de documents contenant le terme

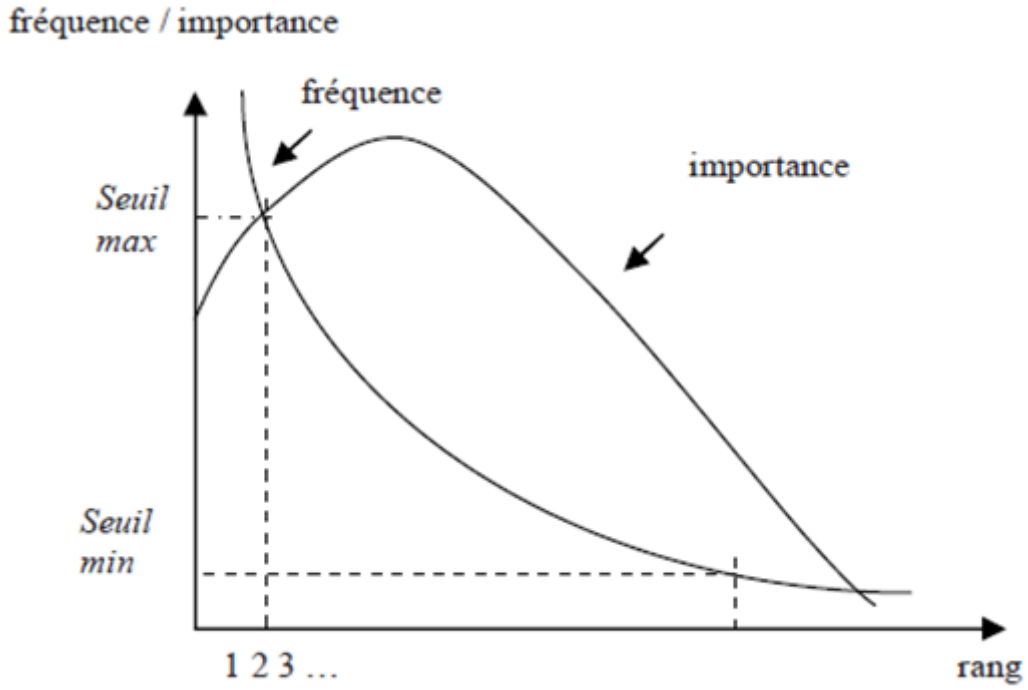

FIGURE 2.8 – Importance d'un terme en fonction de sa fréquence d'apparition dans un document.

et N est le nombre total de documents de la base documentaire La mesure tf * idf donne une bonne approximation de l'importance du terme dans le document, particulièrement dans les corpus de documents de taille homogène. Cependant, elle ne tient pas compte d'un aspect important du document : sa longueur. En général, les documents les plus longs ont tendance à utiliser les mêmes termes de façon répétée, ou à utiliser plus de termes pour décrire un sujet. Par conséquent, les fréquences des termes dans les documents seront plus élevées, et les similarités à la requête seront également plus grandes. Pour pallier cet inconvénient, Robertson et Singhal et al, proposent d'intégrer la taille des documents à la formule de pondération : on parle de facteur de normalisation.

- **Les différentes techniques de création des index :**

Afin de répondre plus rapidement à une requête, des structures de stockage particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation. Les moyens de stockage les plus répandus sont les suivants : les fichiers inverses ("inverted files"), les tableaux de suffixes ("suffix

arrays") et les fichiers de signatures ("signature files"). Les fichiers inverses sont actuellement le meilleur choix possible pour la plupart des applications. Les fichiers inverses sont composés de deux éléments principaux [15] :

* **Le vocabulaire :** qui est l'ensemble de tous les mots différents du texte.
* **Les occurrences (posting) :** pour chaque mot, il s'agit de la liste de toutes les positions dans le texte pour lesquelles le mot apparait. La (figure 9) montre un exemple de vocabulaire et d'occurrences.

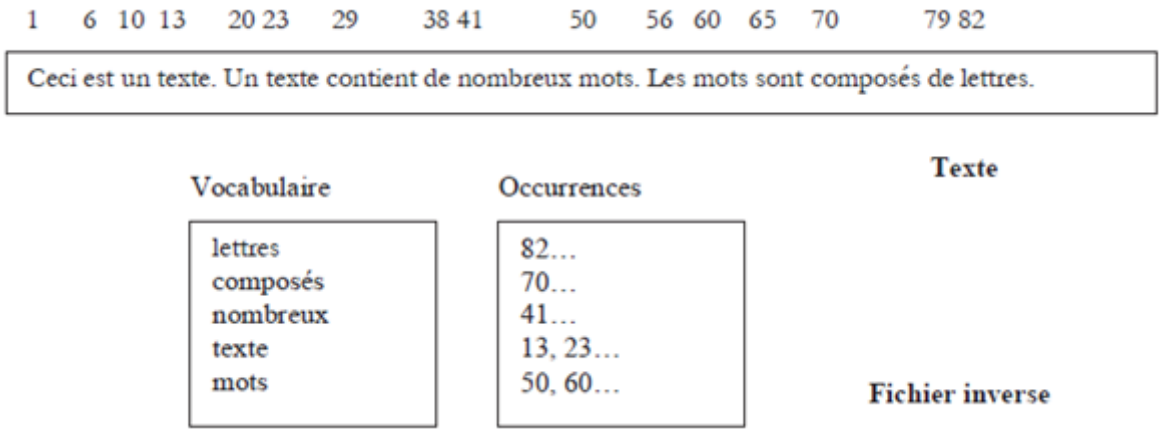

FIGURE 2.9 – Un texte simple et le fichier inverse correspondant.

2.8 Conclusion

Dans ce chapitre nous avons présenté les différentes méthodes de classification des documents XML ainsi que les techniques d'indexations que nous utiliserons dans notre approche. Le chapitre suivant sera consacré pour l'étude de l'approche proposée.

CHAPITRE 3

CONCEPTION ET RÉALISATION DU SYSTÈME DE CLASSIFICATION

3.1 Introduction

Dans ce chapitre nous allons présenter la conception de notre approche, voir le déroulement du processus de classification des documents XML. Nous entamons la description de l'étape de réalisation de notre approche, pour but de générer un code exécutable répond à notre besoin.

3.2 Architecture du notre approche

Dans ce travail, nous proposons une méthode de classification des documents semi-structurés. Notre approche consiste en une représentation de documents XML, basée sur leurs paths ou sous-paths, en utilisant seulement la structure .Nous avons implé-

menté un algorithme de sélection des paths. Ensuite le résultat sera reçu par un autre module (classification), ce dernier utilise un algorithme simple de classification dit k-means. Le résultat final sera un ensemble de classe ou cluster (figure 3.1).

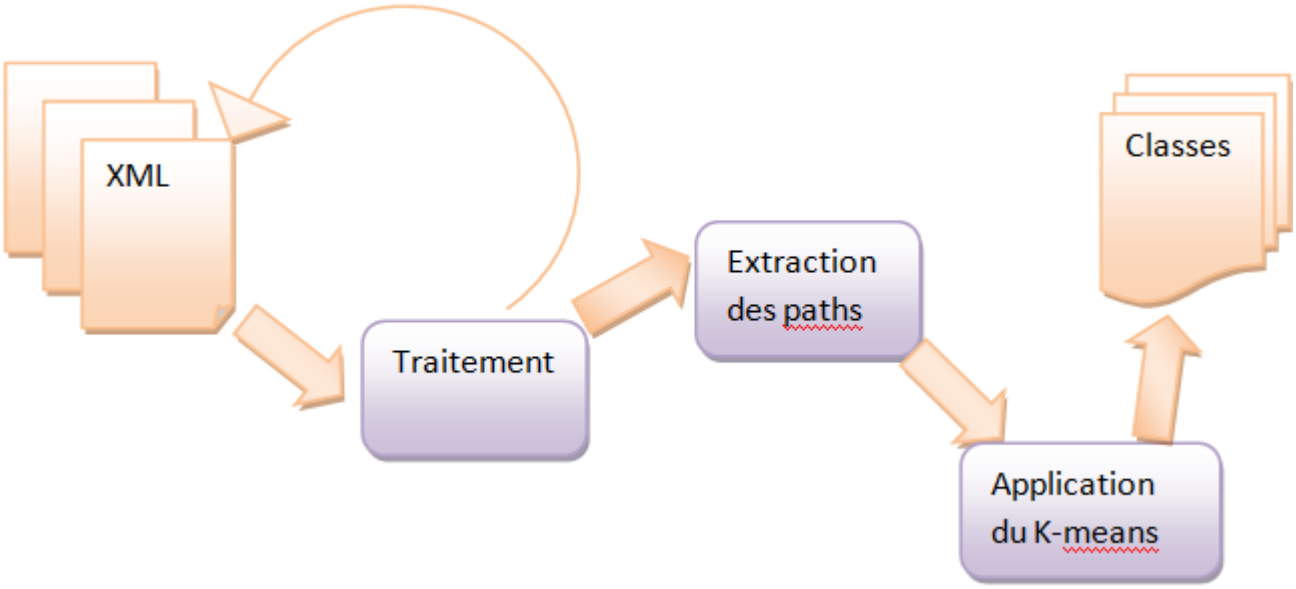

FIGURE 3.1 – la chaine de traitement de l'approche.

3.3 Parsing

Pour parser les documents XML, Nous avons développé un premier programme en Java. on a utilisé l'API JDOM du langage Java. Le programme développé consiste en un module qui s'appuie sur DOM, pour effectuer un premier parsing. Ce module prépare les documents XML pour le traitement. Il les sauvegarde en mémoire sous forme d'arbre.(Figure 3.2).

```
public class fenetre extends javax.swing.JFrame {
 File  f= null; Element root; Element balise; Element parent; Vector v;
    File[] files = null; int data[];  int noofcluster; int centroid[][];File f1;
 Paths path = new Paths();  int p =0 ;
  public fenetre() {
      initComponents(); root=null; balise=null;
    }
    @SuppressWarnings("unchecked")

    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        JFileChooser chooser =new JFileChooser(".");
            chooser.setMultiSelectionEnabled(true);
        int res=chooser.showOpenDialog(this);
      SAXBuilder builder=new SAXBuilder();
        Document document=null;
```

FIGURE 3.2 – XML vers arbre.

3.3.1 descreption des outils

- **Le langage de programmation java**

  Pour implémenter notre application Nous avons choisi le langage de développement java avec l'IDE NetBeans car java est un langage orienté objet simple et portable " Il peut être utilisé sous différentes plates formes sans aucune modification ", java possède aussi une riche bibliothèque de classes. Nous avons utilisé l'environnement NetBeans. C'est un outil de développement intégré, open source, puissant et rapide pour la programmation d'applications.

  Netbeans est un IDE qui supporte une large variété de langages de programmation et d'outils de collaboration. L'environnement de base comprend les fonctions générales suivantes : (Figure 3.2)

  * Configuration et gestion de l'interface graphique des utilisateurs.
  * support de différents langages de programmation.
  * Comprend un explorateur de bases de données qui supporte toutes les bases relationnelles.

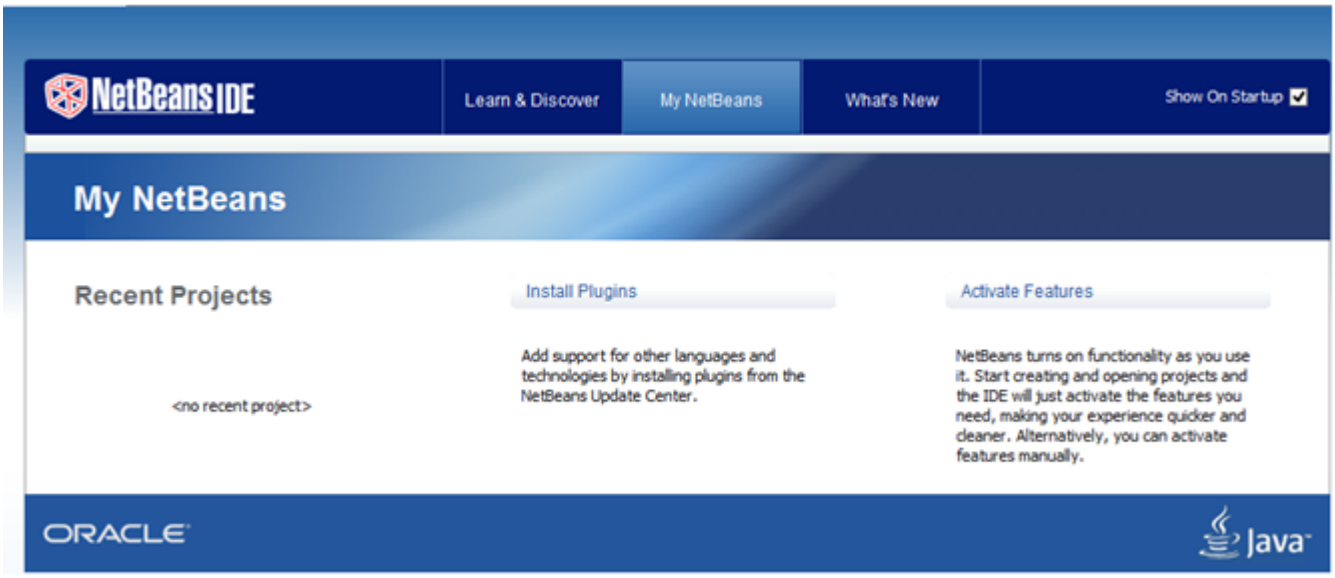

FIGURE 3.3 – l'interface netbeansIDE.

- **L'API JDOM**

  Développée indépendamment de Sun Microsystems. Elle permet de manipuler des données XML plus simplement qu'avec les API classiques. Son utilisation est pratique pour tout développeur Java et repose sur les API XML de Sun. JDOM permet de construire des documents, de naviguer dans leur structure, d'ajouter, de modifier, ou même de supprimer leur contenu.

  Au plus JDOM nous apporte La simplicité, Il est en vérité très laborieux de développer des applications complexes autour de XML avec DOM, qui indiquant le, n'a pas été développé spécifiquement pour Java. JDOM utilise DOM pour manipuler les éléments d'un Document Object Model spécifique (créé grâce à un constructeur basée sur SAX).[16]

- **DOM**

  DOM (Acronyme de Document Object Model), est l'API standardisée par le W3C, Son rôle d'après le w3c est de fournir une représentation mémoire d'un document XML sous la forme d'un arbre d'objets et d'en permettre la manipulation (parcours, recherche et mise à jour). Comme celui de (Figure 3.4).

  Le parsing du fichier est valable si le document XML est bien formé et seulement s'il est valide. Si le fichier XML ne répond pas à ces critères le parseur DOM refuse

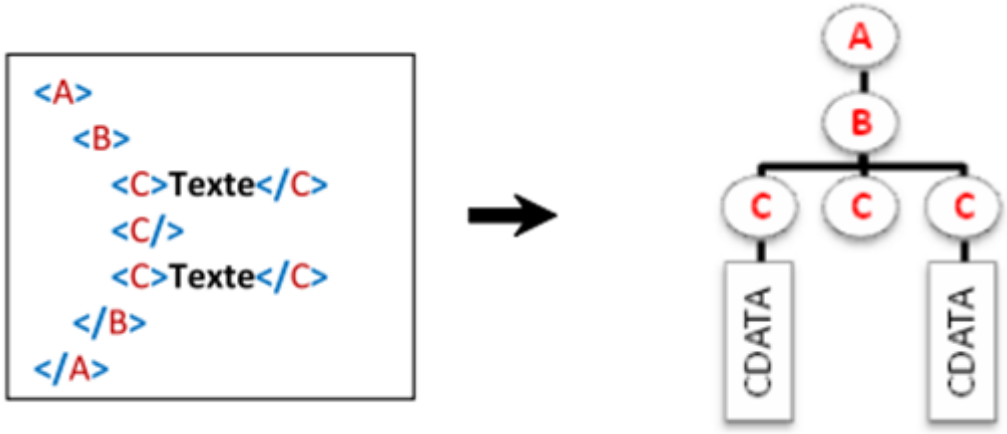

FIGURE 3.4 – Modèle d'arbre DOM.

de créer l'arborescence d'objets en mémoire.[8]

Mais une fois l'arbre créé en mémoire, il devient accessible à l'application grâce à des méthodes d'accès en consultation et en modification, fournies par l'API DOM. Pour mieux comprendre la portée des nœuds (objets) de cet arbre en mémoire, on donne un deuxième exemple concret illustré par la (Figure 3.4) et (Figure 3.5). Le fichier XML de la (Figure 3.4) décrivant des étudiants, a pour arbre DOM en mémoire.

```
< ?xml  version="1.0">
<section nom="ing2">
< !-- description etudiants  -->
   <etudiants>
      <etudiant>
         <matricule>178005844</matricule>
         <nom>XXXXXXXX</nom>
         <prenom>YYYYYYYY</prenom>
      </etudiant>
   </etudiants>
</section>
```

FIGURE 3.5 – Document XML décrivant un étudiant.

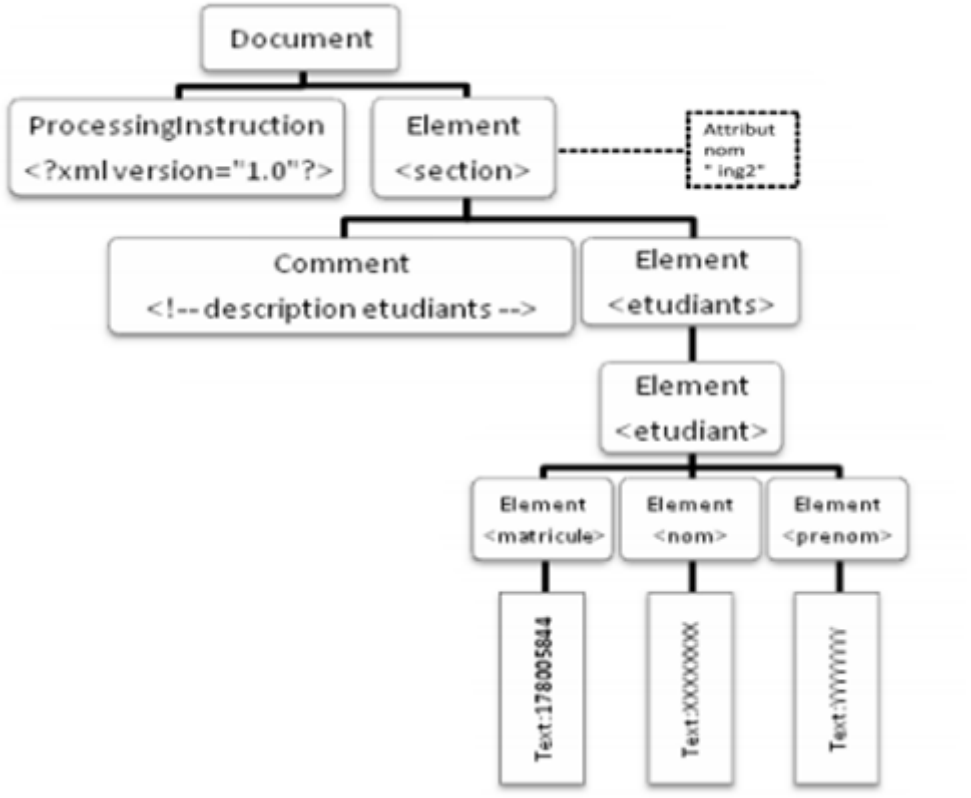

FIGURE 3.6 – Modèle d'arbre DOM du document de la (Figure 3.4).

DOM peut être indépendant du langage dans lequel il sera implémenté (PHP, java, etc...). Il n'est qu'une spécification qui peut être utilisée, doit être implémentée par un éditeur tiers. Ce mécanisme est coûteux en termes de temps et de performance. De plus, il a un coût de stockage très élevé car il est très coutant en mémoire. En revanche, il autorise des traitements en profondeur du document XML, puisque l'arbre d'objets est entièrement chargé en mémoire centrale. Il existe un autre type d'API dénommée SAX complètement différente du DOM à laquelle on fait appel, notamment dans des situations où l'on n'a pas vraiment besoin de charger complètement l'arbre en mémoire, ou dans des cas d'applications spécifiques où le modèle d'arbre que l'on veut créer, serait très différent de l'arbre généré par l'API DOM.

Pour le traitement des documents XML on a réalise une fonction qui sélectionne un ou plusieurs documents XML à classer. La (figure 3.6) suivante illustre le processus.

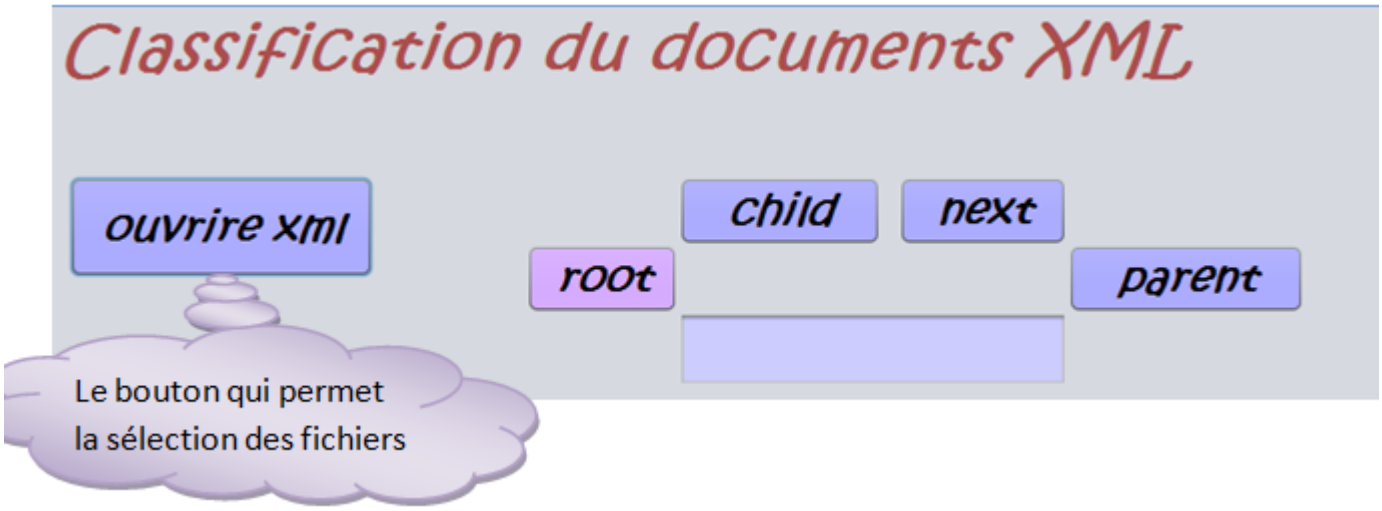

FIGURE 3.7 – Le bouton de sélection des fichiers.

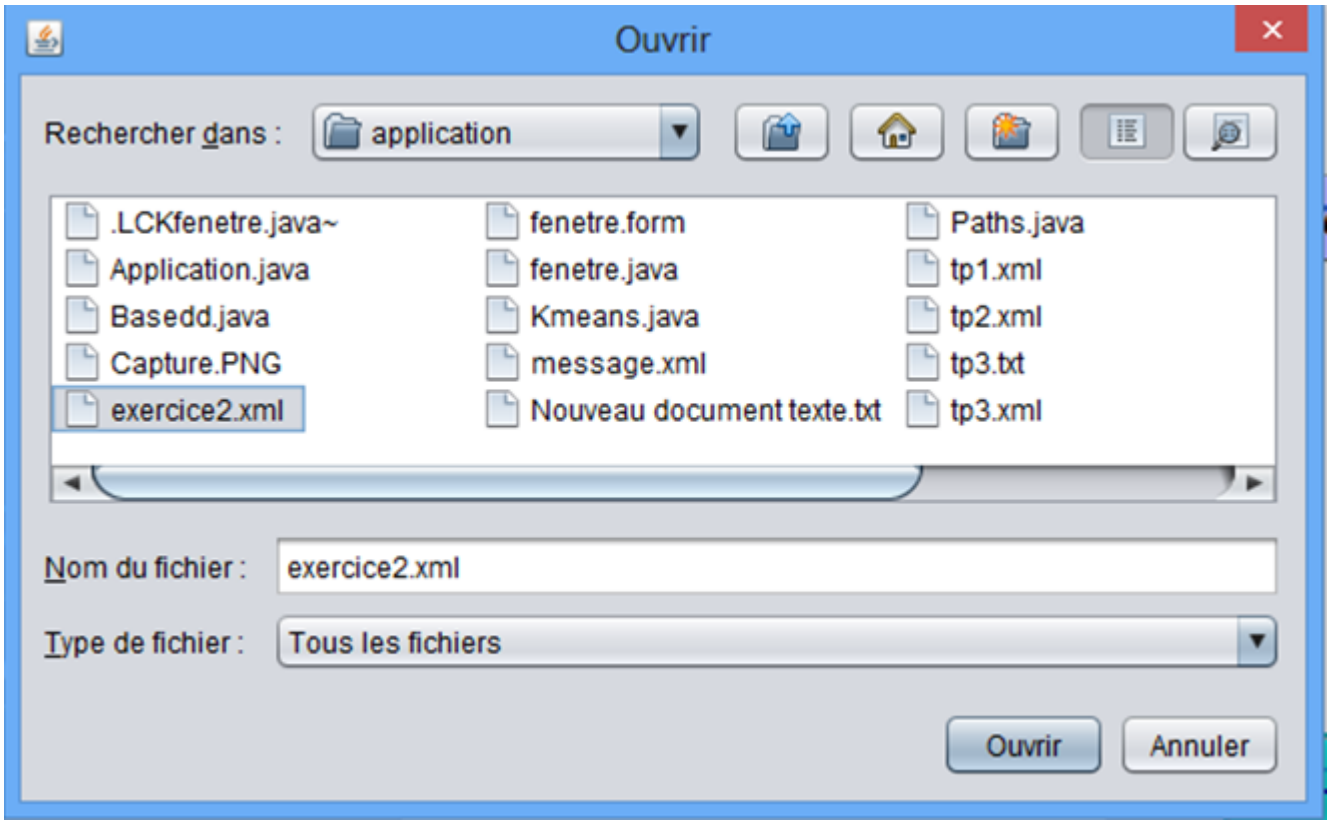

FIGURE 3.8 – La sélection d'un ou plusieurs fichier.

Si on sélectionne un document en peut parcourir ce derniers(Exercice.XML par example), on utilisant l'api jdom, on peut afficher l'élément root (racine) de ce docu-

ment, ou bien les fils de chaque nœud ou bien les parents comme le montre les figures ci-dessus.

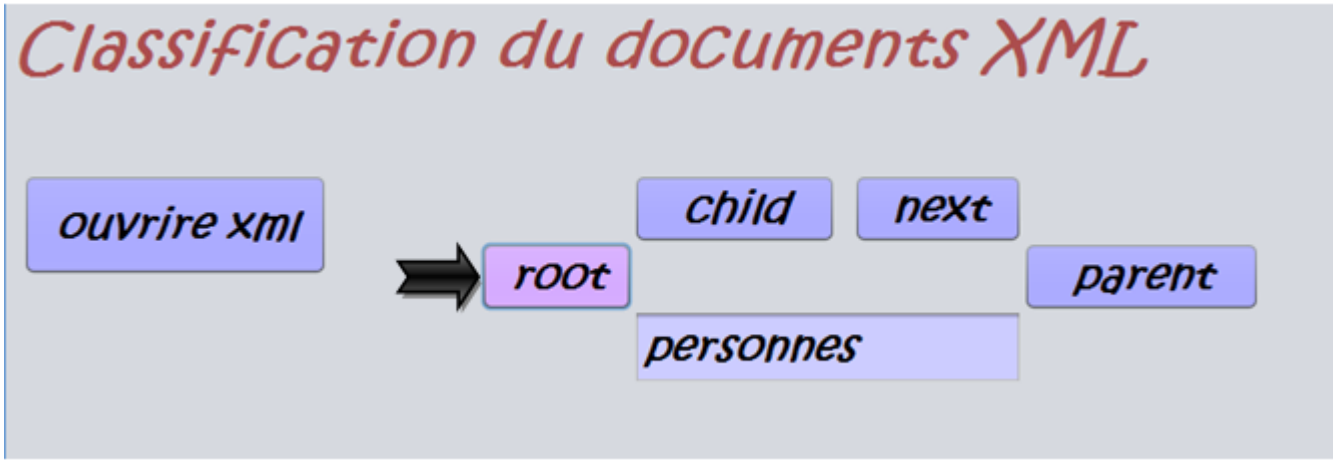

FIGURE 3.9 – L'élément root du fichier sélectionné.

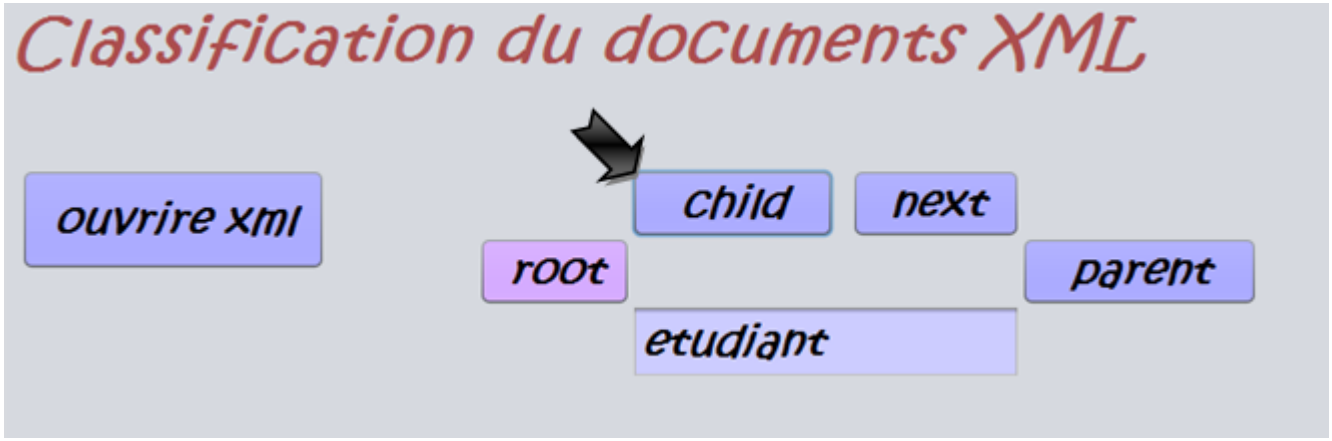

FIGURE 3.10 – L'élément child du fichier sélectionné.

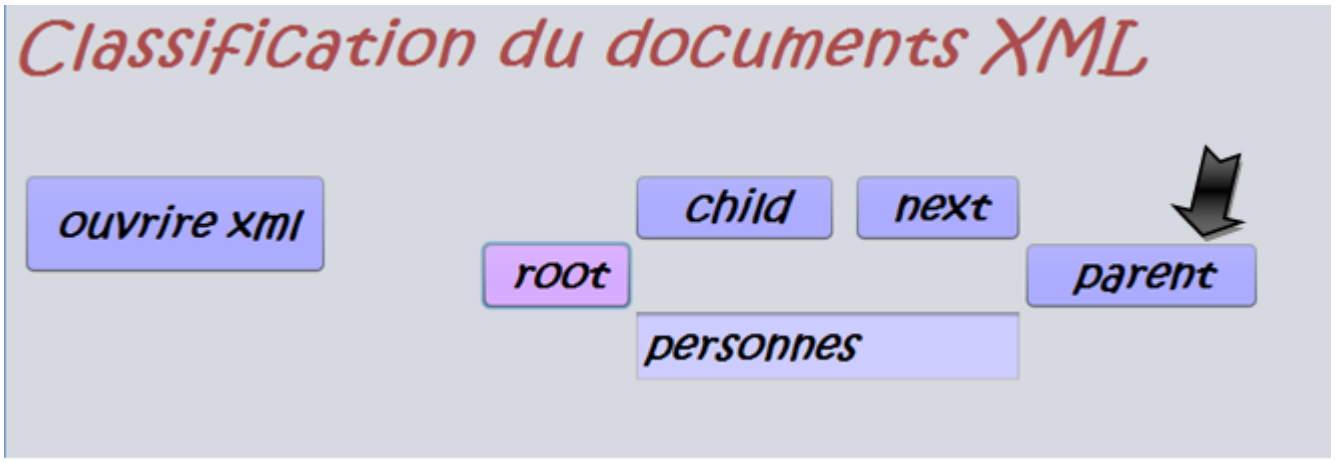

FIGURE 3.11 – L'élément parent du fichier sélectionné.

3.4 Extraction des paths

Pour l'extraction des paths nous avons développé un deuxième module. Ce dernier utilise un ensemble de critères selon notre besoin. Pour cela nous utilisons le langage XPATH.

- **XPATH**

  Xpath est un langage simple de requêtes .L'objectif premier de XPath est de définir la manière d'adresser des parties d'un document XML, cette spécification fournit également les moyens pour traiter des chaînes de caractères, des nombres et des booléens. XPath utilise une syntaxe compacte et non-XML pour faciliter son utilisation dans des URI et des attributs de balises XML, XPath agit sur les structures abstraites et logiques d'un document XML, plutôt que sur sa syntaxe apparente. Le nom XPath vient de l'utilisation d'une écriture de type "chemins d'accès", comme les URL, pour se déplacer à l'intérieur de la structure hiérarchique d'un document XML. Aussi XPath est également conçue pour avoir un sous-ensemble naturel de règles de comparaison (pour tester si un nœud correspond à une forme), cette utilisation de XPath est décrite dans XSLT XPath représente les documents XML comme un arbre de nœuds. Il y a plusieurs types de nœuds, parmi lesquels les nœuds d'éléments, d'attributs et de texte. Ainsi, le

nom d'un nœud est une paire composée d'une partie locale et d'un espace de noms URI pouvant être nul, elle est appelée nom expansé. La forme syntaxique de base de XPath est l'expression. Une expression répond à la règle de production Expr. Une expression s'applique à 4 types possibles d'objets :

* ensemble de nœuds (une suite non-ordonnée de nœuds sans doublon).
* booléen (vrai ou faux).
* nombre (un réel).
* chaîne de caractères (une suite de caractères UCS).[17]

- **Description de l'algorithme de construction des Paths**

  ce module est la suite du premier module parssing . On crée une classe Path.java qui permet de parcourir tous les éléments de l'arbre en partant de la racine, puis de construire les paths qui doivent respecter quelques conditions. Donc le résultat de cette étape est un ensemble de paths.

  Dans la classe Path on importe les packages XML nécessaire ensuite on crée le document Builder, le document file et le Build XPATH. De plus on crée l'expression qui permet l'extraction des sous-paths **" String expression = "//*";"**, la condition de cette expression et de retourner tous les nœuds élément de nos document. Par la suite dans l'algorithme on ajoute un conteur pour calculer la longueur de chaque path qui est retourné par l'expression. Au même temps on a crée une autre fonction qui permet de calculer la TF (l'occurrence) de chaque path dans chaque document XML.(Figure 3.12).

```
    public  void paths(File file) {
try {
    XPath xPath =  XPathFactory.newInstance().newXPath();
    String expression = "//*[not(*)]";
    DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = builderFactory.newDocumentBuilder();
    Document document = builder.parse(file);
    document.getDocumentElement().normalize();

    NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(document, XPat
   Basedd base = new Basedd();
```

FIGURE 3.12 – L'extraction des paths.

Puis stockés ces résultats dans une base de données qui appelé **" dbxml "** constituée d'une table nommé **" fichier "**. Chaque table a l'entête suivante ilustré dans la figure ci-dessus.

FIGURE 3.13 – La base de donné créé.

id : Clé primaire.

Nom-doc : nom de Document XML.

Paths : les paths de ce document.

Longueur : la longueur de chaque path.

Tf-path : L'occurrence du Path dans le document.

Pour créé cette base de données on utilise :

- **XAMPP**

  (X pour cross-plateforme (LAMPP pour Linux, WAMPP pour Windows,...), A pour Apache, M pour MySQL, P pour PHP, P pour Perl) est un ensemble de logiciels permettant facilement de créer une interface web interagissant avec une base de données SQL.

Nous avons aussi fait un bouton Paths qui affiche la base de donnée qui contient des fichiers représenté par des id avec ces paths, longueur et TF de ces paths dans chaque document.

Comme le montre la figure ci-dessus.

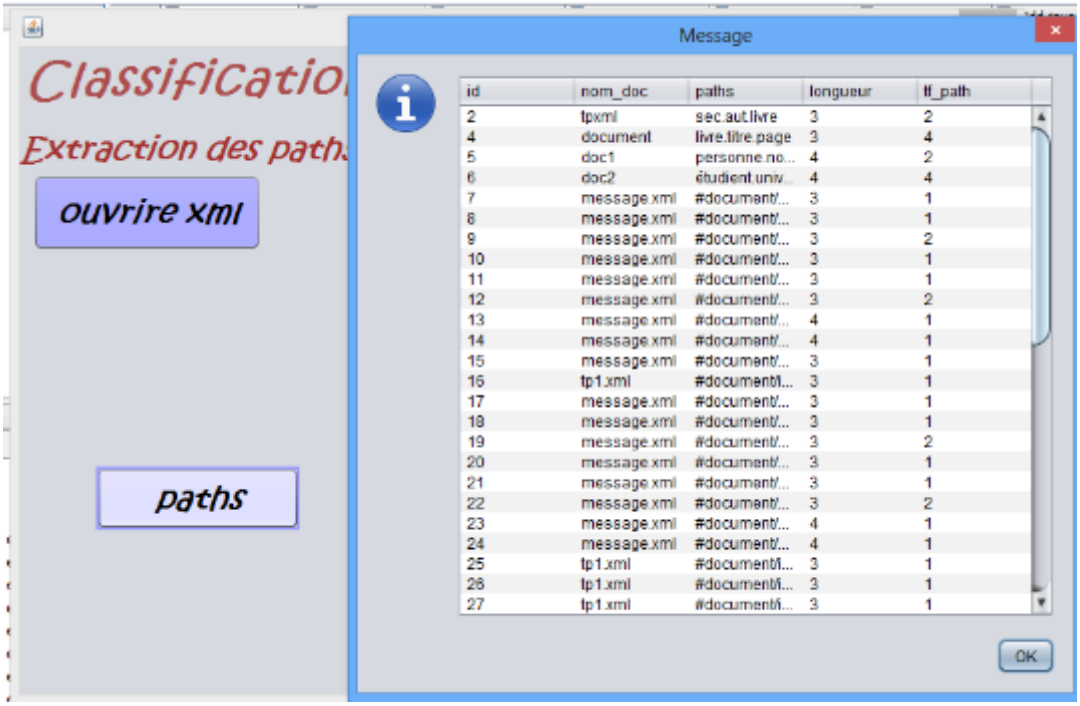

FIGURE 3.14 – L'affichage du contenu de base de donné.

3.5 Classification

On fait cette classification avec l'algorithme k-means (voir les détails dans le deuxième chapitre). Nous utilisons le langage java pour l'implémentation. Comme nous avons mentionné dans le deusième chapitre L'algorithme K-means s'exécute en 4 étapes :

- **Etape 1 :**

  Choisir aléatoirement K objets qui forment ainsi les K clusters initiaux. Dans notre algorithme les données sont les documents XML représentées par leurs noms. Nous avons choisis K objets à partir de ces documents qui doit être différents. Ces derniers seront les centroïdes des classes.

- **Etape 2 :**

  Affecter les objets à un cluster. Pour chaque objet x, le centroïdes qui lui est assigné est celui qui est le plus proche de l'objet, selon une mesure de distance. Dans cette étape nous avons implémenté une fonction pour mesurer cette distance. Nous avons utilisé la distance euclidienne. Illustrons ces étapes c'est dire comment est

le détail :

* Premièrement le document doit être existant dans la base de données.
* Deusièment on sélectionne les colonnes paths et tf-path.
* Troisièment on compare les paths du document avec les paths du centroïde.
  * Si les paths sont égaux on calcule la distance qui se base sur les tf.

$$D(d_i, d_j) = \sqrt{\sum_{l=1}^{n} \sum_{tf=1}^{m} (tf_{d_i} - tf_{d_j})^2}$$

i : allant de 1 jusqu'à le nombre de document totales.

j : allant de 1 jusqu'à le nombre de classe fixer au début.

l : sont des paths égaux.

tf : c'est l'occurrence de ses paths.

  * On choisi la distance minimale et en classe le document.

► **Etape 3 :**

Une fois tous les objets placés, recalculer les centres des K clusters. Pour mettre les nouveaux centroïde en calcule les distances entre tous les objets de la même classe puis leurs moyennes et on choisi la plus petite ainsi le nouveau centroïde est élu.

Voici le détail du déroulement de cette étape :

m : le nombre des tf.

p : le nombre des paths.

n : le nombre totale des documents de la classe.

pour **i** allant de 1 à n

pour **j** allant de 1 à n

1. Calculer la distance euclidienne entre les objets de la classe.

$$d(d_{x_i}, d_{x_j}) = \sqrt{\sum_{l=1}^{p} \sum_{tf=1}^{m} (tf_{x_i} - tf_{x_j})^2}$$

2. Calculer la moyenne de distance de chaque objet avec les autres objets de la même classe.

pour **i** allant de 1 à n

$$Moy = \sum_{j=1}^{n} \left(\frac{d_{x_i}, d_{x_j}}{n} \right)$$

3. On prend le min de ses moyennes comme nouveau centroïd.

$$\min(Moy(d_{x_i}, d_{x_j}))$$

4. On fait ses étapes pour tous les classes trouvées.

► **Etape 4 :**

Répéter les étapes 2 et 3 jusqu'à ce que plus aucune réaffectation ne soit faite.

3.6 Conclusion

Dans ce chapitre, nous avons proposé une approche basé structure pour la classification des documents XML.

Donc, nous avons décrit et conçu notre approche en définissant l'architecture proposée. En suite nous avons présenté l'environnement de développement de notre approche.

Nous pouvons dire qu'on a réalisé une partie importante dans notre travail par l'extraction des paths avec ses longueur et occurrence (tf), puis développé un algorithme K-means pour faire notre classification qu'il utilise les résultats trouvé
(paths, tf ...) comme entré.

CONCLUSION GÉNÉRALE

L'objectif de notre travail est de développer un outil permettant de classifier des documents XML.

Pour mener à terme notre travail, nous avons donné un aperçu général sur les documents XML, ainsi qu'un tour d'oraison sur les méthodes de classifications et d'indexations.

Ce modeste travail nous a offert l'occasion d'approfondir nos connaissances pratiques et théoriques dans le domaine de recherche d'information. En effet nous avant eu l'occasion de nous initier au langage de programmation objet java et de nous familiariser avec le logiciel netbeans qui est caractérisé par sa simplicité et sa puissance. Cependant xpath nous a beaucoup aidé dans la phase d'analyse des documents XML c'est à dire l'extraction des paths. Ainsi que l'application de l'algorithme k-means.

Le logiciel obtenu est un outil qui permet de faciliter la classification des documents XML. Ce logiciel peut être sujet à des améliorations telles que l'ajout du contenu et la construction d'index basé structure et contenu.

Pour conclure nous souhaitons que notre travail soit une base pour la réalisation des travaux ultérieurs s'appuyant sur la structure et le contenu.

BIBLIOGRAPHIE

[1] OLIVIER, P. SGBD, XML, Amie ou ennemies ? .IUT Nancy charlemagne, Université Nancy 2,152 p.

[2] FEGAS, M. Classification de documents XML, Application au corpus, 18p.

[3] STEPHANE, C. Introduction à XML : principes, syntaxe, schémas et manipulations, 70 p.

[4] http://fr.wikipedia.org/wiki/ les types du langage.

[5] http://fr.wikipedia.org/wiki/ Origine du XML.

[6] Stéphane, C. XML extensible markup language [en ligne] (12/04/2019). http://stpxmlUL35struct.html.org/doc/XML/Co /XMLUL35struct.html .

[7] Hunter, D et al. Initiation à XML. Traduction Française : Lemainque,F. Adam, L. Raspaud, C. Wrox Press Edition Eyrolles. (2001).

[8] AiT EL HADJ, A. Modèle de représentation et d'appariement de documents XML selon une indexation de l'information structurelle. Thèse de doctorat : Informatique. Université Mohamed bougera de Boumerdes faculte des sciences, (2011). 142p.

[9] Either, K. Housser, A. XML Web Training autoformation en 30 sessions. OEM Edition. (2000).

[10] MOUNZER, B. Contribution aux méthodes de classification non supervisée via des approches pré topologiques et d'agrégation d'opinions[en ligne].thèse de doctorat en statistique-informatique. Université Claude Bernard- Lyon I, (2007). 171p.

[11] Lebarbier, E. Mary-Huard, T. Classification non supervisée.

[12] MBAO, M. Distance sémantique et carte conceptuel[en ligne]. Recherche en informatique. Université Montpellier II : Sciences et technique du Lang doc. (2007). 43p.

[13] DJEFFAL, A. (2017-2018). Classification Classification Bayésienne[en ligne].2ème année Master Systèmes d'Information, (2017-2018). 21p.

[14] ABBASSI, M. les systèmes de recherche d'information. 20 p.

[15] SAUVAGNAT, K. modèle flexible pour la recherche d'information dans des corpus de documents semi-structuré. Thèse de doctorat de l'université Paul Sabatier: informatique. 232 p.

[16] Belaid, A. JDOM [en ligne]. 33p.

[18] W3C. langage XML path (Xpath)version 1.0[en ligne],(16 novembre 1999).