F.S.D .....N° D'ordre .....

Université Saad DAHLAB de Blida

Faculté des Sciences
Département d'Informatique
Mémoire présenter par :
M^elle^ Bousmaha Sara Hayat
M^d^ Ouzine Sarah

En vue d'obtenir le diplôme de Master
Domaine: MI
Filière: Informatique
Spécialité: Informatique
Option: Ingénierie de logiciel

Sujet :

L'analyse de concept formel pour le clustering des documents XML

Devant le jury composé de :

M^d^ Fareh Massaouda	Président
M^d^ Zhef	Rapporteur
M^d^ Boumahdi Fatima	Examinateur
M^d^ Madani Amina	Promotrice

2013/2014

Résumé :

Les documents XML sont devenus de plus en plus répandus sur le Web. La plupart des algorithmes traditionnels de *data mining* ne permettent pas d'exploiter entièrement l'information contenue dans ce genre de documents.

Pour extraire des connaissances à partir des documents XML, il est très utile de faire intervenir d'autres solutions utilisées actuellement pour la fouille de données mais qui ne sont pas encore généralisées sur d'autres types de données complexes tels que les documents XML.

De ce fait, plusieurs approches proposent de les adapter et d'en sortir de nouveaux algorithmes pour traiter ce type de données. Dans ce cadre, l'Analyse de Concepts Formels (ACF) est une théorie qui est récemment utilisée pour la fouille de données. À travers notre travail, nous voulons découvrir cette théorie en soulevant la question de l'applicabilité de l'analyse de concepts formels pour la représentation et le clustering des documents XML.

Mots clés : document XML, datamining, clustering, analyse de concept formel.

Abstract :

XML documents are becoming increasingly popular on the Web. Most traditional data mining algorithms do not fully exploit the information contained in such documents.

To extract knowledge from XML documents, it is very useful to involve other solutions currently used for data mining but is not yet generalized to other complex data types such as XML documents.

Therefore, several approaches propose to adapt and out new algorithms for this type of data. In this context, the Formal Concept Analysis (FCA) is a theory that was recently used for data mining. In our work we want to explore this theory especially trying to raise the issue of the applicability of formal concept analysis for the representation and clustering XML documents.

Keywords: XML, data mining, clustering, formal concept analysis.

ملخص:

وثائق اكس ام ال تزداد شعبية على شبكة الإنترنت. معظم خوارزميات التقليدية لا تستطيع استغلال المعلومات الواردة في هذه الوثائق.
لاستخراج المعلومة من وثائق اكس ام ال، من المهم حلول أخرى لاستخراج البيانات ولكن لم يتم تعميمها بعد إلى أنواع البيانات المعقدة الأخرى مثل مستندات اكس ام ال.

لذلك، تقترح نهج جديد لمعالجة مثل هذه البيانات. في هذا السياق، وتحليل مفهوم الرسمي (ا س ف) هي نظرية التي يتم استخدامها مؤخرا لاستخراج البيانات. من خلال عملنا نحن نريد لاستكشاف هذه النظرية في محاولة لتمثيل وتجميع وثائق اكس ام ال.

الكلمات الرئيسية: اكس ام ال، استخراج البيانات، تجميع، تحليل مفهوم رسمية(ا س ف)

.

Cet opuscule serait incomplet si nous ne témoignerons pas de notre profonde gratitude envers grand **DIEU** qui nous a conduits à réaliser l'un de nos rêves par sa bienveillance et sa générosité. Toutes les expressions de remerciement ne suffiront jamais pour traduire notre reconnaissance et notre profonde gratitude.

On remercie nos très chers **parents**, qui ont toujours été là pour nous, « Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous avez donné un magnifique modèle de labeur et de persévérance. on est redevable d'une éducation dont on est fier ».

On remercie **Md Madani** notre promotrice qui nous a été le guide durant cette année, on le remercie de nous avoir fait confiance et de nous avoir donné le courage et la force pour vaincre toutes les difficultés.

On remercie infiniment **Mr Smati** pour son aide, sa patience et son attention durant cette année, on le remercie pour les informations très utiles qu'il a mises à notre disposition et on lui souhaite tout le bonheur et la prospérité.

On adresse mes sincères remerciements à tous les **professeurs**, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté à se rencontrer et répondre à nos questions durant nos recherches.

Et enfin, on remercie tous nos **Ami(e)s** qu'on aime Pour leur sincère amitié et confiance, et à qui on doit notre reconnaissance et notre attachement.

À tous ces intervenants, on présente nos remerciements, notre respect et notre gratitude.

dédicaces

Merci Allah (**mon dieu**) de ma voir donné la capacité d'écrire et de réfléchi, la force d'y croire, la patience d'aller jusqu'au bout du rêve et le bonheur de lever mes mains vers le ciel et de dire « Ya Kayoum »

Je dédies ce modeste travail **A mon père**, école de mon enfance ,qui a été mon ombre durant toute ces années des études ,qui a veillé tout au long de ma vie a m'encourager ,a me donner l'aide et a me protéger

A celle qui ma donner la vie, le symbole de la tendresse, qui s'est sacrifice, pour mon bonheur et ma réussite, **A ma mère**

Que dieu les gardes et les protéges

A ma adorable sœur **Abire**
A mes frères **Mouhamed** ,**Ayoub**

A mes amies
A tous ce qui me sont chères
A tous ce qui m'aiment
A tous ce que j'aime
Je dédis ce travail

BOUSMAHA SARA HAYAT

Dédicaces

Je tiens à dédier ce modeste travail à :

- La femme qui a donné de son mieux pour moi et qui a longtemps attendu ce jour: ma chère maman que DIEU la protège.

- L'homme qui a durement travaillé pour que je puisse réaliser l'un de mes rêves et de ses rêves et ma aidé dans touts les étapes de mes étude : mon cher mari BACHIR que DIEU le protège et je lui souhaite tout le bonheur et la réussite.

Sans oublier le sourire de notre maison mon fils YASSER, je lui souhaite tout le bonheur et la réussite.

- Ma chère sœur Souhila et leur fille Asmaa et leur mari Yahia que je la remercie infiniment pour son soutient, je lui souhaite tout le bonheur.

- Ma chère sœur Habiba et leur mari Hakim, lui souhaitent une vie pleine de bonheur.

- Ma chère sœur Karima et leur mari Mourad, lui souhaitent une vie pleine de bonheur.

- Mes chers frères Abd Kader et Mouhamed que DIEU la protègent.

OUZINE SARAH

Sommaire

Chapitre 2 : Le Clustering des données

Chapitre 3 :L'Analyse de Concepts Formels pour le clustering :

ETAT DE L'ART

Chapitre 4 : Une nouvelle approche de Clustering des documents XML basé Sur ACF

Chapitre 5 : Expérimentation et test

Listes de figures

Liste des Tableaux

Listes de Formules

Introduction générale

1. Contexte :

Nous vivons, de nos jours, dans une société de technologies et de progrès. L'être humain est assisté dans son quotidien par différentes formes de technologies pour l'accomplissement de ses tâches. L'information qui représente notre capital vital, est également gérée et suivie par divers outils pour nous permettre de mieux l'utiliser. L'augmentation quasi-exponentielle des connaissances de l'homme ainsi que leur spécialisation, inévitable, dans des domaines d'intérêt très variés, conduit à la production d'un volume d'information sans précédent.

Avec l'émergence de nouveaux standards de représentation de l'information et de stockage tel que le XML, qui sont devenus aujourd'hui très populaires et sont en train de s'imposer. Ils permettent de représenter l'information sous une forme enrichie et adaptée à des besoins spécifiques. Ce type de formats permet de représenter conjointement les données et une information sur leur structure.

Les documents XML présentent l'avantage de posséder une structure explicite qui facilite leur présentation et leur exploitation dans différents contextes. Cependant, très souvent, la majeure partie de l'information reste contenue dans les champs textuels. Il est donc devenu primordial de concevoir des méthodes permettant d'exploiter à la fois la structure et le contenu textuel de ces documents.

La fouille des documents XML (*XML mining*) est l'application et l'adaptation des techniques de fouille de données et de fouille de texte existantes, afin de tenir compte des spécificités des documents XML. Le but est d'extraire des unités de connaissances significatives et réutilisables. Ces unités peuvent avoir plusieurs formes, par exemple : des règles d'association, des patterns, des catégories de documents similaires…Nous nous intéressons plus particulièrement aux techniques de classification de documents XML : plus précisément, la classification non supervisée (*clustering*).

2. Problématique :

La fouille dans les documents XML comprend deux catégories :

- La fouille dans la structure des documents XML (*XML Structure Mining*) qui s'intéresse à l'extraction d'informations à partir de la structure des documents XML.
- La fouille dans le contenu des éléments XML (*XML Content Mining*) qui regroupe les méthodes de fouille sur le contenu des documents. La fouille sur le contenu utilise habituellement les techniques de *text mining* pour extraire l'information contenue dans le document.

Les techniques classiques de fouille de données (*data mining*) et de fouille de texte (*text mining*) ne permettent pas d'exploiter entièrement l'information contenue dans ce genre de documents c'est-à-dire que la nature des documents XML présente des informations diverses sur les données. Les techniques de fouille traditionnelles sont basées sur une représentation linéaire des documents (*bag of words*). Cette représentation est assez brute et ne permet pas d'effectuer des traitements poussés sur les documents, vu que cela revient à traiter des chaînes de caractères.

3. Objectifs :

Puisque les techniques classiques de fouille de données ne tiennent pas compte des aspects spécifiques des documents XML, nous allons intervenir d'autres solutions utilisées actuellement pour la fouille de données mais qui n'ont pas encore généralisées sur d'autres types de données complexes tels que les documents XML pour exploiter l'information contenue dans ces documents.

L'Analyse de Concepts Formels (l'ACF) est une théorie qui est récemment utilisée pour la fouille de données. Elle consiste à restructurer la théorie des treillis de Galois afin de faciliter son utilisation dans des applications du monde réel et de permettre l'interprétation de ses notions en dehors du cadre théorique aussi bien par des mathématiciens que par des non-mathématiciens.

Notre principal objectif est d'exploiter l'ACF pour la représentation et le clustering des documents XML en prenant en considération soit la structure seule soit le contenu seul soit la structure et le contenu des documents XML. Le but est de proposer une nouvelle approche et de faire des propositions logicielles pour l'aide à la classification non supervisée de collections de documents XML en se basant sur l'ACF.

4. Organisation de mémoire :

Ce mémoire est organisé autour de cinq chapitres :

- **Chapitre 1 : Les documents XML**

  Dans ce chapitre, nous avons cité les notions essentielles du langage de balisage extensible XML, sa structure de base, ses principaux composants, les techniques de description des documents telles que les DTD et les schémas XML et les conditions de validation d'un document XML.

- **Chapitre 2 : Le clustering des données**

  Ce chapitre a permis de présenter les concepts de base du *data mining*, une étude détaillée liée au clustering. Nous avons traité les différentes étapes de clustering, les méthodes de clustering.

- **Chapitre 3 : L'Analyse de Concepts Formels pour le clustering : État de l'art**

  Dans ce chapitre nous avons présenté les notions de base relatives à l'ACF telles que les contextes formels, les concepts formels, les treillis de concepts… Nous avons étudié l'applicabilité de l'ACF pour le clustering des documents en parcourant les travaux réalisés dans ce cadre. À la fin, une discussion des points forts et des points faibles de l'ACF est présentée, afin de l'appliquer pour la fouille des documents XML.

- **Chapitre 4 : Une nouvelle approche de clustering des documents XML basé sur l'ACF**

  Dans ce chapitre, nous avons proposé les différentes phases constituant notre nouvelle approche de fouille dans les documents XML.

- **Chapitre 5 : Tests et résultats**

  Dans ce chapitre, nous décrivons les différents outils utilisés dans notre projet, ensuite nous présenterons une démonstration globale des différentes tâches accomplies par notre application.

La conclusion dresse quelques points importants sur nos contributions dans cette recherche et enfin les perspectives envisagées pour complémenter ce travail.

1. Introduction :

Dans ce chapitre, nous présentons les grandes lignes du langage XML : comment écrire un document XML, étudier sa structure de base, ses principaux composants, aborder les techniques de description de documents telles que les DTD et les schémas XML et voir les conditions de la validation d'un document XML. **[1].**

2. XML :

XML (*eXtensible Markup Language*), langage de balisage extensible standardisé par le W3C (*World Wide Web Consortium*), est un langage de description et d'échange de documents structurés **[2]**.

Donc il s'agit d'un format de documents permettant de représenter des données et leurs sens sous forme de texte, tout en conservant une certaine organisation, et de définir d'autres langages de description structurés spécifiques à certains domaines **[2]**.

3. Historique du XML :

L'historique suivant retrace les grandes étapes qui ont conduit à la naissance de XML. L'ancêtre de XML est le langage SGML qui a été introduit en 1986 **[3]** par C. Goldfarb. SGML a été conçu pour des documentations techniques de grande ampleur. Sa grande complexité a freiné son utilisation en dehors des projets de grande envergure. En 1991**[3]**, T. Berners-Lee a défini le langage HTML pour le WEB. Ce langage est une version simplifiée à l'extrême de SGML, destinée à une utilisation très ciblée. XML est, en quelque sorte, intermédiaire entre SGML et HTML. Il évite les aspects les plus complexes de SGML tout en gardant suffisamment de souplesse pour une utilisation généraliste. La version 1.0 de XML a été publiée en 1998**[3]** par le consortium W3C (World Wide Web Consortium) **.** Une redéfinition XHTML de HTML 4.0 à travers XML a été donnée en 1999**[3]**. Une seconde version 1.1, qui est simplement une mise à jour pour les caractères spéciaux en lien avec Unicode, a, ensuite, été publiéeen2004 **[3].**

4. Les avantages de XML :

Les principaux avantages qui peuvent motiver le choix de XML sont :

- Lisibilité **[2]** : les balises peuvent être définies librement pour marquer les éléments préservant la sémantique des composants d'un document en utilisant des termes assez intuitifs.
- Séparation de la structure et de la présentation **[4]** : Dans un document XML, L'information de formatage est écrite dans un langage de style et est stockée séparément. Une balise indique ce que l'information signifie, non pas comment l'afficher.
- Modularité et réutilisabilité **[2]** : Tout utilisateur de XML peut définir librement sa propre structure de document XML (DTD ou Schéma XML).

XML offre un mécanisme simple et puissant pour partager et réutiliser des parties de structures types.

- Accès à des sources d'information hétérogènes **[2]** : XML propose un format d'échange de données normalisé, général, indépendant de toute plate-forme et de tout SGBD, et suffisamment puissant pour que l'immense majorité des données puissent être efficacement représentées et manipulées.
- Contrôle de validité **[2]** : La validation d'un document XML garantit que la structure de données utilisée respecte sa DTD ou son schéma XML.

5. XML n'est pas ?

- Ce n'est pas un langage de programmation, tels Basic, C, Prolog, Perl, Java, etc.
- Ce n'est pas un format propriétaire, tel le format de Word, ou PDF
- Ce n'est pas non plus une base de données, mais en un certain sens XML permet le stockage des informations structurées

6. Structure d'un document XML :

Tout document XML se compose **[2]** :

- D'un prologue : C'est l'entête du document dont la présence est facultative mais conseillée.
- D'un arbre d'éléments : C'est le corps du document XML.
- De commentaires et d'instructions de traitements, dont la présence est facultative, et qui pourront moyennant certaines restrictions, apparaitre aussi bien dans le prologue que dans l'arbre d'éléments.

Nous illustrons à travers un exemple les différents composants d'un document XML :

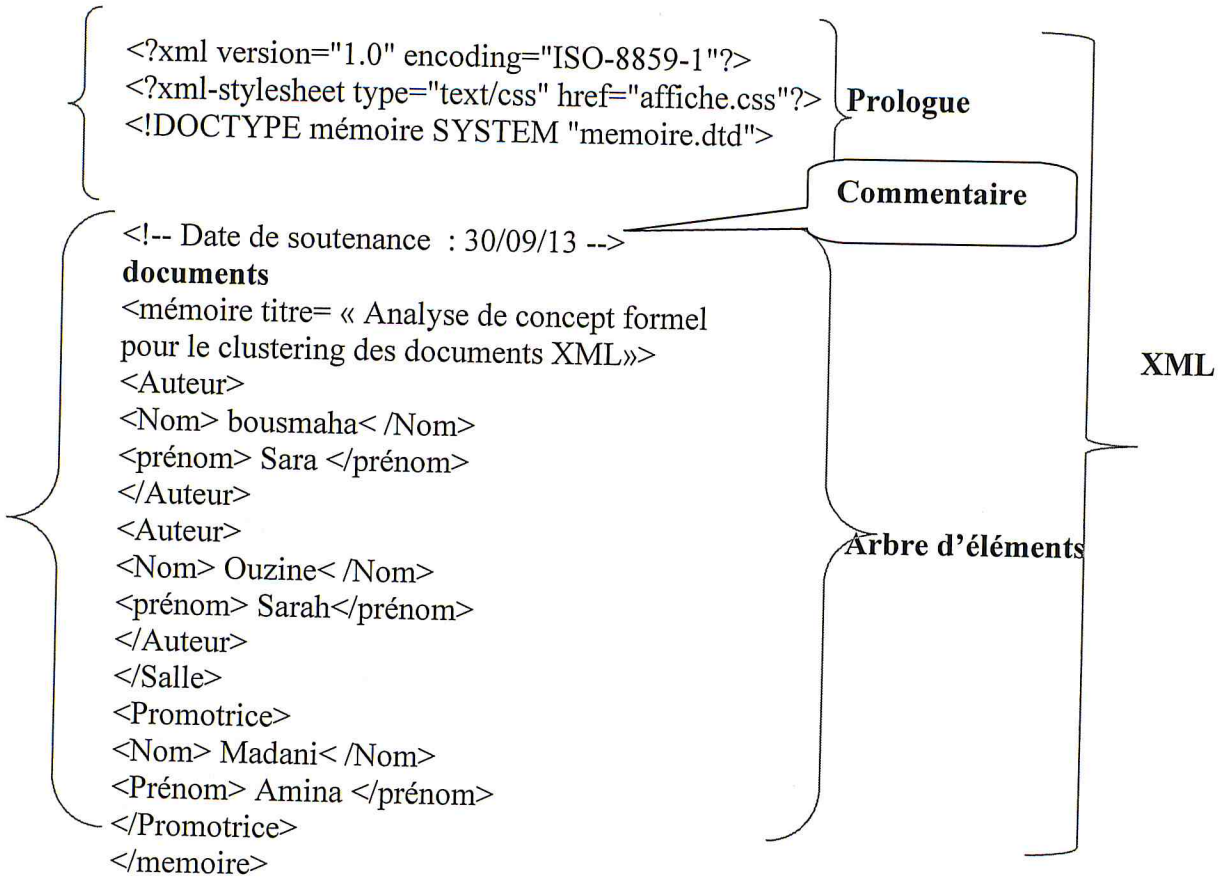


Figure 1.1 : Structure d'un document XML.

6.1. Le prologue :

Le prologue XML est composé d'une déclaration XML, d'instructions de traitement et éventuellement d'une ou plusieurs déclarations de type de documents **[2]**.

6.1.1. La déclaration XML :

La déclaration XML est la première ligne d'un document XML. Elle permet de donner les caractéristiques globales du document. Parmi ces caractéristiques, nous citons **[2]** :

- La version du langage XML (1.0 ou 1.1).
- Le jeu de caractères d'encodage du document XML « *encoding* » (UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 à ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP), lorsque l'encodage n'est pas précisé, la valeur UTF- 8 est utilisée par défaut.

- La déclaration de document autonome : (*standalone="yes"*) spécifie si le document est autonome ou s'il dépend, pour son fonctionnement, d'autres fichiers ou de toutes autres ressources externes (*standalone="no"*).

Une déclaration XML est encapsulée par <? et ?> et a la forme suivante :

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

6.1.2. Les instructions de traitement :

Les instructions de traitement sont destinées aux applications qui traitent les documents XML. Elles sont l'analogue des directives #... du langage C qui s'adressent au compilateur. Elles peuvent apparaître aux mêmes endroits que les commentaires à l'exception du contenu de la DTD **[2].**

Les instructions de traitement sont délimitées par les chaînes de caractères '<?' et '?>'. Les deux caractères '<?' sont immédiatement suivis du nom XML de l'instruction. Le nom de l'instruction est ensuite suivi du contenu. Ce contenu est une chaîne quelconque de caractères ne contenant pas la chaîne '?>' utilisée par l'analyseur lexical pour déterminer la fin de l'instruction. Le nom de l'instruction permet à l'application de déterminer si l'instruction lui est destinée **[2].**

<? xml-stylesheet type="text/css" href="affiche.css"?>

6.1.3. Les déclarations de type de document :

La déclaration de type définit la structure du document. Elle précise en particulier quels éléments peuvent contenir chacun des éléments. Cette déclaration de type peut prendre plusieurs formes suivant que la définition du type est interne, c'est-à-dire incluse dans le document ou externe. Elle a la forme générale suivante qui utilise le mot clé DOCTYPE.

<! DOCTYPE memoire SYSTEM "memoire.dtd">

6.2. Les commentaires :

Les commentaires sont utilisés pour ajouter du texte explicatif à un document XML. Ils peuvent inclure n'importe quel caractère sauf le « -- » **[2]**, ils prennent la forme suivante :

<!-- Date de soutenance : 21/01/14 -->

6.3. L'arbre d'éléments :

Le corps du document est constitué de son contenu qui est organisé de façon hiérarchique. L'unité de cette organisation est l'*élément*. Chaque élément peut contenir du texte simple, comme un fichier, d'autres éléments, comme un répertoire, ou encore un mélange des deux.

Comme dans une arborescence de fichiers, il y a un élément appelé *élément racine* qui contient l'intégralité du document.

6.3.1. Éléments :

Un *élément* est formé d'une balise ouvrante, d'un contenu et de la balise fermante correspondante, Le contenu, peut être composé de sous éléments. Un élément prend la forme suivante :

<nom> contenu </nom>

Le nom d'un élément figurant dans les balises ouvrante et fermante doit respecter les règles suivantes **[2]**, **[5]** :

- Il doit être formé de caractères alphanumériques, du tiret-souligné (_), du signe moins (-) ou du point (.). Le caractère deux points (:) est utilisable mais il a un sens particulier qui sera décrit plus tard.
- Il ne doit pas contenir de caractère d'espacement ou de fin de ligne.
- Son premier caractère doit être alphabétique ou un tiret-souligné (_).
- Il ne peut pas commencer par la chaine « xml » qu'elle soit en minuscules, majuscules, ou un mélange des deux.
- Les majuscules et minuscules sont différenciées.

Un élément peut contenir d'autres éléments, des données, des références à des entités **[2]**. Il peut être vide quand il n'y a pas de contenu mais il peut contenir des attributs **[6]**. Le document XML ci-dessous contient :

- *mémoire* : élément racine contenant trois éléments fils : *Auteur* et *Salle*
- *Auteur* : élément contenant des données et deux éléments fils *Nom, prénom.*
- *Nom* : élément contenant des données.
- *prénom* : élément contenant des données.
- *salle* : élément vide sans contenu.

6.3.2. Les attributs :

Les balises ouvrantes peuvent contenir des *attributs* associés à des valeurs. L'association de la valeur à l'attribut prend la forme *attribute='value'* ou la forme *attribute="value"* où *attribute* et *value* sont respectivement le nom et la valeur de l'attribut.

7. DTD :

Le rôle d'une DTD (*Document Type Definition*) est de définir précisément la structure d'un document. Il s'agit d'un certain nombre de contraintes que doit respecter un document pour être *valide*. Ces contraintes spécifient quelles sont les éléments qui peuvent apparaître dans le contenu d'un élément, l'ordre éventuel de ces éléments et la présence de texte brut. Elles définissent aussi, pour chaque élément, les attributs autorisés et les attributs obligatoires **[2].**

Les DTD ont l'avantage d'être relativement simples à utiliser mais elles sont parfois aussi un peu limitées. Les schémas XML permettent de décrire de façon plus

précise encore la structure d'un document. Ils sont plus sophistiqués mais plus difficiles à mettre en œuvre. Les DTD sont donc particulièrement adaptées pour des petits modèles de documents. En revanche, leur manque de modularité les rend plus difficiles à utiliser pour des modèles plus conséquents **[2].**

7.1. Déclaration de la DTD :

La déclaration de la DTD du document doit être placée dans le prologue. La DTD peut être interne, externe ou mixte. Elle est *interne* si elle est directement incluse dans le document. Elle est *externe* si le document contient seulement une référence vers un autre document contenant la DTD. Elle est finalement mixte si elle est constituée d'une partie interne et d'une partie externe **[2].**

La déclaration de la DTD est introduite par le mot clé DOCTYPE et a la forme générale suivante où *root-element* est le nom de l'élément racine du document.

< !DOCTYPE *root-element* ..... >

Le nom de l'élément racine est suivi du contenu de la DTD dans le cas d'une DTD interne ou de l'URL du fichier contenant la DTD dans le cas d'une DTD externe.

7.1.1 DTD interne :

Lorsque la DTD est incluse dans le document, sa déclaration prend la forme suivante où son contenu est encadré par des caractères crochets '[' et ']'.

< !DOCTYPE *root-element* [declaration] >

Les déclarations *déclarations* constituent la définition du type du document.

7.1.2 DTD externe :

Lorsque la DTD est externe, celle-ci est contenue dans un autre fichier dont l'extension est généralement .dtd. Le document XML se contente alors de donner l'adresse de sa DTD pour que les logiciels puissent y accéder. L'adresse de la DTD peut être donnée explicitement par une URL ou par un FPI (*Formal Public Indentifier*). Les FPI sont des noms symboliques donnés aux documents **[2].**

7.1.3 DTD mixte :

Il est possible d'avoir simultanément une DTD externe adressée par URL ou FPI et des déclarations internes. La DTD globale est alors formée des déclarations internes suivies des déclarations externes. La déclaration prend alors une des deux formes suivantes On retrouve un mélange de la syntaxe des DTD externes avec les mots clés SYSTEM et PUBLIC et de la syntaxe des DTD internes avec des déclarations encadrées par les caractères '[' et ']'.

7.2. Contenu de la DTD :

Une DTD est constituée de déclarations d'éléments, d'attributs et d'entités. Chacune de ces déclarations commence par la chaîne '<!' suivi d'un mot clé qui indique le type de déclaration. Les mots clés possibles sont ELEMENT, ATTLIST et ENTITY. La déclaration se termine par le caractère '>'.

7.2.1 Entités :

Une entité peut être considérée comme un alias permettant de réutiliser des informations au sein du document XML ou de la définition DTD.

Il existe trois classifications principales pour les entités : les entités générales, les entités paramètres et les entités externes.

7.2.1.1 Les entités générales :

Les entités générales sont les entités les plus simples. Elles permettent d'associer un alias à une information afin de l'utiliser dans le document XML **[7].** La syntaxe d'une entité générale :

```
<!ENTITY nom "valeur">
```

Pour utiliser une entité générale dans notre document XML, il suffit d'utiliser la syntaxe suivante : &nom;

Afin d'illustrer un peu plus clairement, voyons tout de suite un exemple :

```
<!ENTITY samsung "Samsung">
<!ENTITY apple "Apple">

<telephone>
   <marque>&samsung;</marque>
   <modele>Galaxy S3</modele>
</telephone>
<telephone>
   <marque> &apple;</marque>
   <modele>iPhone 4</modele>
</telephone>
```

Au moment de son interprétation, les références aux entités seront remplacées par leurs valeurs respectives, ce qui donne une fois interprété :

```
<telephone>
   <marque Samsung</marque>
   <modele>Galaxy S3</modele>
</telephone>
<telephone>
   <marque Apple</marque>
```

<modele>iPhone 4</modele>
</telephone

7.2.1.2. Les entités paramètres :

Contrairement aux entités générales qui apparaissent dans les documents XML, les entités paramètres n'apparaissent que dans les définitions DTD. Elles permettent d'associer un alias à une partie de la déclaration de la DTD **[7].** La syntaxe d'une entité paramètre :

<!ENTITY % nom "valeur">

Pour utiliser une entité paramètre dans notre DTD, il suffit d'utiliser la syntaxe suivante : %nom;

Prenons par exemple cet exemple où des téléphones ont pour attribut une marque :

<telephone marque="Samsung" />
<telephone marque="Apple" />

Normalement, pour indiquer que l'attribut marque de la balise <telephone/> est obligatoire et qu'il doit contenir la valeur Samnsung ou Apple, nous devons écrire la règle suivante :
<!ATTLIST telephone marque (Samsung|Apple) #REQUIRED>

A l'aide d'une entité paramètre, cette même règle s'écrit de la façon suivante :
<!ENTITY % listeMarques "marque (Samsung|Apple) #REQUIRED">
<!ATTLIST telephone %listeMarques; >

Encore une fois, au moment de son interprétation, les références aux entités seront remplacées par leurs valeurs respectives.

7.2.1.3. Les entités externes :

Les entités externes rapatrient leur contenu à partir d'une autre ressource externe désignée par une URI **[8]**. Elles peuvent être également des entités générales ou des entités paramètres accessibles à partir de fichiers externes. Elles prennent la syntaxe suivante :

<!ENTITY nom-de-l'entité SYSTEM "URI">

Dans cet exemple, une entité paramètre externe est définie par :

<!ENTITY % mpeg SYSTEM "http://www.mpeg.com">

Il existe en réalité 2 types d'entités externes : les analysées et les non analysés.

7.2.2. Déclaration d'élément :

Les déclarations d'éléments constituent le cœur des DTD car elles définissent la structure des documents valides. Elles spécifient quels doivent être les enfants de chaque élément et l'ordre de ces enfants.

La déclaration d'un élément prend la forme suivante où *élément* et *type* sont respectivement le nom et le type de l'élément. Le type de l'élément détermine quels sont ses contenus autorisés.

<!ELEMENT élément type >

7.2.2.1. Contenu pur d'éléments :

Le contenu d'un élément est *pur* lorsqu'il ne contient aucun texte et qu'il est uniquement constitué d'éléments qui sont ses enfants.

<!ELEMENT élément *regexp* >

Le nom de l'élément est donné par l'identifiant *élément*. L'expression rationnelle *regexp* décrit les suites autorisées d'enfants dans le contenu de l'élément.

7.2.2.2. Contenu textuel :

La déclaration de la forme suivante indique qu'un élément peut uniquement contenir du texte. Ce texte est formé de caractères, d'entités qui seront remplacées au moment du traitement.

<!ELEMENT élément *(#PCDATA)* >

7.2.2.3. Contenu mixte :

La déclaration de la forme suivante indique qu'un élément peut uniquement contenir du texte et les éléments element1, …, elementN. C'est la seule façon d'avoir un contenu mixte avec du texte et des éléments. Il n'y a aucun contrôle sur le nombre d'occurrences de chacun des éléments et sur leur ordre d'apparition dans le contenu de l'élément ainsi déclaré. Dans une telle déclaration, le mot clé #PCDATA doit apparaître en premier avant tous les noms des éléments **[2].**

<!ELEMENT élément *(#PCDATA|element1|……..|elementN)** >

7.2.2.4. Contenu libre :

Avec ANY aucune contrainte n'est spécifiée sur le contenu de l'élément **[1]**. Tous les éléments déclarés dans la DTD sont utilisés dans n'importe quel ordre et sans limitation du nombre d'occurrences. L'élément quelconque est déclaré comme suit :

<!ELEMENT Rapport ANY>

7.2.3. Déclaration d'attribut :

Une déclaration d'attribut commence par une balise <!ATTLIST (élément) (nom(s) d'attribut(s)>. Les attributs peuvent être de différents types.

- ID, IDREF: identificateurs uniques, qui peuvent être mis en relation par référence (IDREF).

- LIST (ou ENUMERATION): choix d'attributs imposé.

- ENTITY, ENTITIES: entité(s) déclarée(s) et non analysée de la DTD.

- NMTOKEN, NMTOKENS: unité(s) lexicale nominale XML.

- CDATA: une chaîne de caractères.

8. Les espaces de noms :

L'espace de noms est une collection des noms, identifiée par une référence d'URI **[1]**. Un premier usage des espaces de noms dans un document XML consiste à utiliser simplement l'espace de noms par défaut. Ce dernier est précisé par un pseudo-attribut *xmlns* (*ns : namespace*). L'espace de noms par défaut s'applique à l'élément où se situe sa déclaration et à tout son contenu **[6]**.

Ci-dessous, l'élément *chapitre* est dans l'espace de noms http://www.monouvrage.dz.
C'est également le cas de l'élément *paragraphe*, puisqu'il est dans l'élément *chapitre*.

```
<chapitre xmlns="http://www.monouvrage.dz">
<paragraphe>
XML
</paragraphe>
</chapitre>
```

L'espace de noms par défaut peut être changé même dans les éléments enfants en utilisant une règle de priorité.

```
<chapitre xmlns=" http://www.monouvrage.dz ">
<paragraphe xmlns="http://www.autrelivre.com">
XML
</paragraphe>
</chapitre>
```

L'élément *paragraphe* n'appartient pas à l'espace de noms http://www.monouvrage.dz mais uniquement à l'espace de noms http://www.autrelivre.com.

L'espace de noms par défaut présente l'inconvénient d'être peu contrôlable sur un document de taille importante. En effet, tout ajout ou modification d'un tel espace va se répercuter sur la totalité du contenu **[6]**.

9. Schémas XML :

Les *schémas XML* permettent, comme les DTD, de définir des modèles de documents. Il est ensuite possible de vérifier qu'un document donné est valide pour un schéma, c'est-à-dire respecte les contraintes données par le schéma. Les schémas ont été introduits pour combler certaines lacunes des DTD.

À la différence des DTD dont la syntaxe est différente de XML et le seul type atomique est le type chaîne de caractères, les schémas XML introduisent le typage des données, ce qui permet la gestion de booléens, d'entiers, d'intervalles de temps, etc. Il est même possible de créer de nouveaux types à partir de types existants tout en utilisant une syntaxe XML **[9]**.

Alors que les DTD ne tiennent pas compte des espaces de noms, les schémas XML tirent parti des espaces de noms et les utilise fréquemment. Il existe deux espaces de noms dédiés aux schémas **[6]** :

• http://www.w3.org/2001/XMLSchema : espace de noms utilisé pour les éléments du schéma XML du W3C, il peut être utilisé comme espace de nom par défaut ou avec préfixe *xsd* (par exemple <xsd:element>…</xsd:element>).

• http://www.w3.org/2001/XMLSchema-instance : espace de noms utilisé pour les extensions du schéma XML du W3C employées dans des documents d'instance, il doit être défini avec un préfixe *xsi*.

D'autres espaces de noms cibles peuvent être déclarés pour correspondre aux éléments et attributs déclarés dans le document XML.

Un schéma XML contient une déclaration XML, une déclaration de l'élément racine, des déclarations des éléments, des déclarations de types simples, des déclarations de types complexes et des déclarations des attributs **[1]**.

9.1. Structure globale d'un schéma :

Un schéma XML se compose essentiellement de déclarations d'éléments et d'attributs et de définitions de types.

```
<xml version= "1.0" encoding="iso-8856-1" ?>
<xsd :shema xmln :xsd="http://www.w3.org/2001/XMLShema">
<!—Déclaration déléments,d'attributs et definitions de types-->
.....
< /xsd :shema>
```

9.2. Déclarations d'éléments :

On déclare un composant <element> dans un schéma XML comme suit :
<xsd:element name="*nom-de-l'élément*" type="type-de-*contenu-de-l'élément*">

9.2.1. Déclaration de types simples <simpleType>:

Les types simples définissent uniquement des contenus textuels. Ils peuvent être utilisés pour les éléments ou les attributs. Ils sont introduits par l'élément xsd:simpleType .

Un type simple est souvent obtenu par restriction d'un autre type défini. Il peut aussi être construit par union d'autres types simples ou par l'opérateur de listes. La déclaration d'un type simple a la forme suivante.

< xsd:simpleType >
.....
</ xsd:simpleType >

9.2.2. Déclaration de types complexes <complexeType> :

Les types complexes définissent des contenus purs (constitués uniquement d'éléments), des contenus textuels ou des contenus mixés. Tous ces contenus peuvent comprendre des attributs. Les types complexes peuvent seulement être utilisés pour les éléments. Ils sont introduits par l'élément xsd:complexType. Un type complexe peut être construit explicitement ou être dérivé d'un autre type par extension ou restriction.

La construction explicite d'un type se fait en utilisant les opérateurs de séquence xsd:sequence, de choix xsd:choice ou d'ensemble xsd:all. La construction du type se fait directement dans le contenu de l'élément xsd:complexType et prend donc la forme suivante.

< xsd:complexType >
< !—construction du type avec xsd:sequence, xsd:choice ou xsd:all >
.....
</ xsd:complexType >

9.3. Déclaration des attributs <attribute> :

La déclaration d'un attribut est très semblable à celle d'un élément **[6]**. Sa syntaxe est la suivante :

<xsd:attribute name="nom-attribut" type="type-attribut " use="utilisation-attribut" />
Pour illustrer la définition d'attributs, prenons l'exemple suivant où on a deux attributs *titre* et *auteur* :

<xsd:element name="cours" >

```
<xsd:complexType>
<xsd:attribute name="titre" type="xsd:string" />
<xsd:attribute name="auteur" type="xsd:string" use="required" />
</xsd:complexType>
</xsd:element>
```

Avec *attribute* les attributs les plus couramment utilisés sont : *name*, *form*, *type*, *ref*, *use*, *default*, et *fixed*. L'attribut *use* peut prendre les valeurs suivantes **[6]** :

- *prohibited* : l'attribut ainsi défini ne doit pas apparaître.
- *optional* : l'attribut peut apparaître au plus une fois, et c'est la valeur par défaut quand l'attribut *use* n'est pas utilisé.
- *required* : l'attribut doit obligatoirement apparaître.

10. Un document bien formé, un document valide :

Chaque document XML doit être « bien formé ». C'est simplement un document qui suit toutes les règles de notation et de structure du langage XML **[8]** :

- Il doit commencer par une déclaration XML.
- ·Les éléments non vides ont une balise ouvrante et une balise fermante.
- Les éléments vides ne comportant qu'une balise doivent la clore par />.
- Il contient un seul élément racine encapsulant tous les autres éléments et leurs attributs.
- Il contient un ou plusieurs éléments.
- Les éléments non vides sont correctement imbriqués.
- Un nom d'attribut apparaît uniquement dans la balise ouvrante et une seule fois dans cette balise.
- Les valeurs des attributs sont entre guillemets ou apostrophes **[10]**.
- Les caractères réservés sont remplacés par des références d'entités (par ex. < pour<).
- s'il n'y a pas de DTD, les seules entités utilisées sont celles réservées de XML (&amp, &lt, &gt, &apos, et &quot).

Un document est « valide » si **[2]** :

- Il est bien formé ;
- Il est conforme à ce qui est défini dans la DTD associée ou conforme au schéma qui lui est associé.

11. Conclusion :

XML dans sa capacité à exprimer les données structurées standard de l'industrie, offre de nombreux avantages aux organisations, développeurs de logiciels, de sites Web et les utilisateurs finaux. Comme la plupart des applications d'entreprise impliquant la manipulation ou le transfert de données et les enregistrements dans les bases de données, tels que les bons de commande, factures, informations clients, les rendez-vous, etc., XML va révolutionner les possibilités des utilisateurs finaux Internet, permettant la mise en œuvre d'un large éventail d'applications d'entreprise **[11].**

Suite à l'analyse des principales caractéristiques de XML, nous pouvons dire que nous entrons dans une deuxième phase du World Wide Web et, par conséquent, il y aura un développement plus intelligent application autour de cette langue. Nous devrons surmonter certains obstacles, tels que les grandes entreprises doivent se qualifier aux normes approuvées par le W3C, XML qui va devenir quelque chose de plus que le successeur de HTML. XML également être mis en œuvre dans des applications hors-monde Internet / Intranet, ce qui rend beaucoup plus facile et pratique l'échange de données entre applications. Aujourd'hui, nous pouvons dire que XML n'est pas une promesse, mais une réalité **[11].**

Chapitre II

Le clustering des données

1. Introduction

La classification est une des tâches centrales de l'étape de fouille de données dans le processus d'extraction de connaissances. Le problème de la classification est traité dans plusieurs communautés de recherche qui se découvrent et s'enrichissent mutuellement : statistiques, reconnaissances de formes, apprentissage automatique, réseaux de neurones et raisonnement à partir de cas. Le terme *classification* en français désigne à la fois les termes anglais *classification* (classification supervisée) et *clustering* (classification non supervisée). Nous nous focalisons sur la classification non supervisée.

2. Fouille de données :

Data mining, exploration de données, extraction de connaissances à partir de données, *knowledge data discovery*...etc., plusieurs termes utilisés pour désigner la Fouille de données. Pour la décrire nous employons les définitions suivantes :

La *data mining*, ou fouilles de données, est l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de grandes bases de données informatiques **[12]**.

C'est un domaine pluridisciplinaire qui regroupe des techniques d'apprentissage automatique, de la reconnaissance de forme, des statistiques, des bases de données et de la visualisation pour apporter une réponse à l'extraction d'information provenant de base de données de grande taille **[13]**.

3. Tâches de la fouille de données :

Les principales tâches du *data mining* peuvent se résumer comme suit :

3.1. La description :

C'est un moyen pour décrire des phénomènes ou des tendances gisant dans les données, les résultats du modèle doivent décrire des caractéristiques claires qui puissent amener à une interprétation et une explication intuitive. Certaines méthodes de data mining sont plus adaptées que d'autres pour une interprétation transparente. Par exemple, les arbres de décision fournissent une interprétation et une explication intuitive de leurs résultats **[14]**.

3.2. La classification :

La classification est l'opération qui permet de placer chaque individu de la population étudiée dans une classe, parmi plusieurs classes prédéfinies, en fonction des caractéristiques de l'individu indiquées comme variables explicatives. Le plus souvent, il y a deux classes prédéfinies, et le résultat de la classification est un ensemble de règles permettant d'affecter chaque individu à l'une ou l'autre des classes, sachant que la probabilité de « bon classement » dépend généralement de la règle qui s'applique à l'individu : certaines sont plus fiables que d'autres **[15]**.

Dans la classification, la variable cible est catégorielle, comme le revenu, qui peut être divisé en trois classes : revenu élevé, revenu moyen et faible revenu **[14]**. Les techniques utilisées pour la classification sont les k-plus proches voisins, les arbres de décision, et les réseaux de neurones **[14]**.

3.3. L'estimation :

L'estimation est similaire à la classification excepté la variable cible, continue au lieu d'être catégorielle **[14]**. Par exemple estimer les résultats du baccalauréat en fonction des résultats du brevet des collèges pour une population des lycéens.

Le domaine de l'analyse statistique classique fournit plusieurs méthodes d'estimation efficaces et très largement utilisées. Les réseaux de neurones peuvent aussi être utilisés pour l'estimation **[14]**.

3.4. La prévision (prédiction) :

Elle est similaire à la classification et à l'estimation mise à part que pour la prévision, les résultats portent sur le futur **[14]**. Prévoir les gagnants du championnat de football, par rapport à une comparaison des résultats des équipes est un exemple de prévision.

Les techniques les plus appropriées à la prévision sont les k-plus proches voisins, les arbres de décision, et les réseaux de neurones **[14]**.

3.5. Le clustering *:*

Le clustering porte sur le regroupement d'enregistrements, d'observations ou de cas en groupe d'objets similaires. Il est différent de la classification parce qu'ils n'y a pas de variable cible pour segmenter. Le clustering n'essaye pas de classer, d'estimer ou de prédire la valeur d'une variable cible mais cherche à segmenter l'ensemble entier des données en dessous groupes relativement homogènes, des clusters, dans lesquels la similarité des enregistrements dans le groupe est maximisée et la similarité en dehors du groupe est minimisée **[14]**.

3.6. L'association :

L'association est la tâche qui permet de trouver quelles variables vont ensemble. Elle extrait les corrélations entre les données. Très répandue dans le secteur de la distribution car leur principale application est « l'analyse du panier de la ménagère » qui consiste en la recherche d'associations entre produits sur les tickets de caisse. Trouver tous les articles achetés ensemble et ceux qui ne sont jamais achetés ensemble dans un supermarché est un exemple de la fonction d'association **[14]**. L'algorithme des règles d'association est utilisé pour générer des règles d'association.

4. Clustering :

Le clustering (ou regroupement) est un sujet de recherche en apprentissage émanant d'une problématique plus générale, à savoir la classification.

La classification non-supervisée fut, dans un premier temps, utilisée dans les domaines D'application suivants :

- la médecine **[16]**: découverte de classes de patients présentant des caractéristiques Physiologiques communes,
- le traitement de la parole **[17]**: construction de systèmes de reconnaissance de l'orateur,
- l'archéologie **[18]**: regroupement d'objets datant de l'âge de fer à partir de leur description,
- l'éducation **[19]**: structuration d'une classe d'ouvriers dans le domaine.

5. Cadre formel du clustering :

On considère un ensemble de n objets X = { $x_1,.....,x_n$ }, ainsi qu'une matrice de dissimilarité D sur cet ensemble, telle que d(xi, xj) représente la dissimilarité entre les deux objets xi et xj . La matrice D est de taille n×n et a valeurs dans [0, 1]. Lorsque les données sont uniquement décrites par une telle matrice, on parle parfois de "données relationnelles"**[20].**

Nous verrons, par la suite, que la plupart des méthodes de classification non-supervisées utilisent une description vectorielle des données. On parle alors de "données objets" et ont considéré un ensemble fini V = { $v_1,.....,v_n$ } de variables descriptives, telles que vj (xi) désigne la valeur de l'objet xi ∈ X pour la variable vj ∈ V. La tâche de clustering permet de générer un ensemble de t clusters C = { $C_1,.....,C_n$ } tel que chaque cluster Ca est un sous-ensemble de X (Ca ⊂X) et l'union des clusters couvre l'ensemble des objets de départ ($\bigcup_{a=1}^{t} C_a = X$).

5.1. Partitions, pseudo-partitions et partitions floues :

Définition 1.1. C'est une partition de X si et seulement si C vérifie les propriétés suivantes **[16]** :

i) Ca ⊂X pour tout Ca ∈ C,
ii) $u_a(x_i)$ = X,
iii) Ca ∩Cb = ϕ pour (a, b) tel que a ≠ b.

La propriété iii) exprime le fait que les clusters constitués sont disjoints, chaque objet de X ne peut donc appartenir qu'a un seul cluster de C. Une telle partition sera également appelée "partition stricte"

Définition 1.2. C'est une pseudo-partition de X si et seulement si C vérifie les propriétés suivantes :

i) Ca ⊂ X pour tout Ca ∈ C,

ii) $\bigcup_{a=1}^{t} C_a$ Ca = X,

iii) Ca ∩ Cb ssi a = b.

Dans le cas d'une pseudo-partition, les intersections entre clusters ne sont pas nécessairement vides. Cependant, la propriété iii) interdit qu'un cluster soit inclus dans un autre.

Dans les deux définitions précédentes, chaque objet xi appartient ou non à un cluster Ca donné. On peut alors formaliser le processus de construction d'une partition ou d'une pseudo-partition par la donnée de t fonctions a valeurs binaires **[20]** :

$$v_a : X \to \{0,1\}, a = 1....t \ avec \ v_a(x_i) = \begin{cases} 1 \ si \ x_i \in C_a \\ 0 \ si on \end{cases}$$

Formule 2.1 : formule de construction d'une partition **[20]**

Cette formalisation peut être généralisée au cas de fonctions à valeurs réelles. Dans ce cas, les partitions générées sont dites "floues"

Définition 1.3. Une partition floue de X, notée C = {C1,….., Ct}, est définie par la donnée de t fonctions

x_n

$u_a(x_i)$ Représente alors le degré d'appartenance de l'objet xi au cluster Ca.

5.2. Hiérarchies et pseudo-hiérarchies :

Certaines méthodes de clustering conduisent à un arbre hiérarchique aussi appelé dendrogramme. De même que l'on distingue les partitions strictes des pseudo-partitions, cette même distinction est faite entre hiérarchies et pseudo-hiérarchies **[20].**

Définition 1.4. Soit P un ensemble de parties non vides sur X, P est une hiérarchie si les propriétés suivantes sont vérifiés :

i) X ∈ P,
ii) pour tout xi ∈ X, {xi} ∈ P,
iii) pour tout h, h'∈ P, h ∩ h' ∈ { ϕ, h ,h'}
iv) pour tout h ∈ P, ∪ {h' ∈ P : h'⊂ h} ∈ {h, ϕ}

Les propriétés i) et ii) expriment le fait que la racine de l'arbre est constituée de l'ensemble X et que les feuilles de l'arbre correspondent aux singletons $\{ x_1 \}, \ldots\ldots, \{ x_n \}$.
Les propriétés iii) et iv) assurent que deux clusters ne s'intersectés que si l'un est inclus dans l'autre et que chaque cluster contient tous ses successeurs ("fils") et est contenu dans son unique cluster prédécesseur ("père").

Définition 1.5. Soit P un ensemble de parties non vides sur X, P est un pseudo hiérarchie si les propriétés suivantes sont vérifies :

i) X ∈ P,
ii) pour tout xi ∈ X, {xi} ∈ P,
iii) pour tout h, h' ∈P, h ∩ h' =ϕ ou h ∩ h0 ∈ P,
iv) il existe un ordre (total) θ sur X compatible avec P

Définition 1.6. Un ordre θ est compatible avec un ensemble P de parties de X, si tout élément de h ∈ P est connexe selon θ

Définition 1.7. Une partie h est connexe selon θ si x et y étant les bornes (i.e. le plus petit et le plus grand élément) de h selon θ, on a la condition :

$$\{z \text{ compris entre } x \text{ et } u \text{ selon } \theta\} \Leftrightarrow \{z \in h\}$$

Par ces définitions, une pseudo-hiérarchie (aussi appelée pyramide [46]) est telle que chaque cluster peut avoir plusieurs prédécesseurs. De plus, la propriété iv) concernant l'existence d'un ordre θ sur X, permet la visualisation d'une telle pyramide. La notion de hiérarchie floue n'est pas formellement définie. Ce type d'organisation est en effet très peu utilisé en apprentissage

5.3. Centroïdes et médoïdes :

De nombreux algorithmes de clustering assimilent chaque cluster a un point, ceci pour des raisons pratiques de complexité notamment. Ce "point", censé être représentatif du cluster peut être un centroïde ou un médoïde **[20]**.

Définition 1.8. Le centroide x d'un cluster Ca est le point défini dans V par :

$$\forall j = 1, \ldots\ldots, p \quad v_j\left(x^*\right) = \frac{1}{|c_a|} \sum_{x_i \in c_a} v_i\left(x_i\right)$$

Formule 2.2 : formule de calculer le centroide **[20]**

Dans le cas où toutes les variables descriptives v1,....., vp sont quantitatives (continues ou discrètes), le centroide est défini, sur chaque composante, par la valeur moyenne des objets du cluster pour cette même composante. En ce sens, le centroide correspond au centre de gravité du cluster. Le centroide d'un cluster ne fait généralement pas partie des objets constituant ce cluster

En revanche, lorsque certaines variables descriptives sont de nature qualitative (ex. couleur, forme, etc.), la définition précédente n'a plus de sens puisque les opérateurs classiques (addition, division, etc.) ne sont pas applicables sur ce type de variables. On cherche alors l'objet le plus représentatif du cluster que l'on appelle aussi parfois " prototype" ou plus formellement "médoïde".

Définition 1.9. Soient un cluster Ca d'objets définis sur V, et d'une mesure de dissimilarité sur V, le médoïde du cluster Ca est l'objet x∈ Ca tel que :

$$x^* = \arg_{x_i \in c_a} \min \frac{1}{|c_a|} \sum_{x_j \in c_a} d\left(x_i, x_j\right)$$

Formule 2.3 : formule de construction le médoïdé **[20]**

Par cette définition, le médoïde d'un cluster est l'objet du cluster tel que la dissimilarité moyenne de tous les objets du cluster avec le médoïde est minimale. Inversement, le médoïde d'un cluster est l'objet en moyenne le plus similaire aux autres.

5.4. Concavité vs. convexité d'un cluster :

On est parfois amené à s'intéresser à la "forme" des clusters obtenus par un algorithme de clustering. Notons que cette notion de "forme" est difficile à définir formellement et pourrait laisser penser que les objets doivent systématiquement être représentés dans un espace. Par exemple, dans un espace à deux dimensions, les formes convexe et concave peuvent être illustrées par la figure 2.1 **[20].**

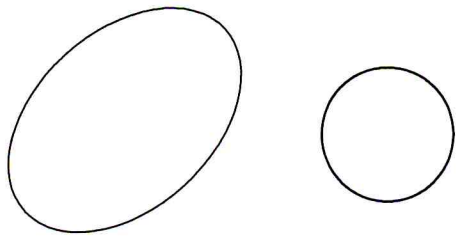


Figure 2.1 : Exemples de clusters convexes (gauche) et concaves (droite), dans R2 **[20]**

Un cluster est convexe lorsque les objets qui le composent sont organisés autour d'un centre (centroïde ou médoïde).

5.5. Autres notions utiles :

Nous définissons enfin plusieurs notions permettant de caractériser les clusters. Notons que ces critères se retrouvent dans la plupart des méthodes de clustering qui consistent à optimiser une fonction de qualité. Dans les définitions suivantes, nous

considérerons un cluster Ca composé d'objets de X ; x_a^* correspondant au centre du cluster (centroïde ou médoïde) ; et d'étant une mesure de dissimilarité (ou distance) sur X

Définition 1.10. Le rayon d'un cluster correspond à la plus grande distance du centre à un autre élément

$$rayon(C_a) = \max_{\{x_i \in c_a\}} d(x_i, x_a^*)$$

Formule 2.4 : formule de calcul le rayon **[20]**

Définition 1.11. Le diamètre d'un cluster est égal à la plus grande distance entre deux éléments de ce cluster

$$diam(C_a) = \max_{\{x_i, x_j \in c_a\}} d(x_i, x_j)$$

Formule 2.5 : formule de calcul le diamètre **[20]**

Définition 1.12. L'inertie d'un cluster (inertie intra-cluster) correspond à la somme des carrés des distances au centre

$$I_{intra}(C_a) = \sum_{x_i \in c_a} d(x_i, x_a^*)^2$$

Formule 2.6 : formule de calcul inertie intra-cluster**[20]**

Définition 1.13. étant donné un schéma de clustering C = {C1,…..,Ct}, l'inertie inter clusters correspond à la somme des carrés des distances entre les centres des clusters

$$I_{inter}(C_a) = \sum_{i=2}^{t} \sum_{j<i} d(x^*_i, x^*_j)^2$$

Formule 2.7 : formule de calcul L'inertie inter-cluster **[20]**

Définition 1.14. La variance d'un cluster est égale à la moyenne des carrés des distances au centre

$$V(C_a) = \frac{1}{|C_a|} \sum_{x \in c_a} d(x_i, x_a^*)^2$$

Formule 2.8 : formule de calcul la variance **[20]**

6. Les trois principales étapes du clustering:

Le processus de clustering se divise en trois étapes majeures : (1) la préparation des données, (2) l'algorithme de clustering et (3) l'exploitation des résultats de l'algorithme **[21], [20].**

Nous discutons ici des problématiques générales liées à chacune de ces étapes.

6.1. La préparation des données

6.1.1. Variables et sélections :

Les objets sont décrits par des variables {vj} j=1…..p, aussi appelées attributs, descripteurs ou traits. Ces variables sont de différentes natures :

- variables quantitatives : continues (ex. la taille d'une personne), discrètes (ex. le nombre de personnes) ou sous forme d'intervalles (ex. la période de vie d'une personne),
- variables qualitatives : non-ordonnées (ex. la "couleur" des cheveux) ou ordonnées (ex. la taille : "petit", "moyen", "grand", etc.),
- variables structurées : par exemple la forme d'un objet (polygone, parallélogramme, rectangle, ovale, cercle, etc.)

L'étape de préparation consiste à sélectionner et/ou pondérer ces variables, voire à créer de nouvelles variables, afin de mieux discriminer entre eux les objets à traiter. En effet les variables ne sont pas nécessairement toutes pertinentes : certaines peuvent être redondantes et d'autres non-pertinentes pour la tache ciblée. Ce problème de sélection de variables a été largement étudié en classification supervisée mais reste très ouvert pour une approche non-supervisée **[20].**

6.1.2. Distances et similarités :

La plupart des algorithmes de clustering utilisent une mesure de proximité entre les objets à traiter. Cette notion de "proximité" est formalisée à l'aide d'une mesure (ou indice) de similarité, dissimilarité ou encore par une distance. Le choix ou la construction de cette mesure est déterminant pour la suite du processus. En effet, deux mesures différentes peuvent induire deux schémas de classification différents. Finalement, chaque domaine d'application possédant ses propres données, il possédé également sa propre notion de "proximité" ; il faut concevoir alors une mesure différente pour chaque domaine d'application, permettant de retranscrire au mieux les différences (entre les objets) qui semblent importantes pour un problème donné **[20].**

6.2. Le choix de l'algorithme :

Le choix de l'algorithme de clustering doit donner lieu à une analyse globale du problème : quelle est la nature (qualitative et quantitative) des données ? Quelle est la nature des clusters attendus (nombre, forme, densité, etc.) ? L'algorithme doit alors être choisi de manière à ce que ses caractéristiques répondent convenablement à ces deux dernières questions. Les critères de décision peuvent être : la quantité de données à traiter, la nature de ces données, la forme des clusters souhaités ou encore le type de schéma attendu (pseudo-partition, partition stricte, dendrogramme, etc.) **[20]**.

6.2.1. La taille des données :

La quantité d'objets à traiter est un premier facteur de décision. En effet, pour des données de très grande taille (par exemple en traitement d'images), les algorithmes de complexité plus que linéaires (O(…:n) sur le nombre n d'objets) sont quasiment prohibés **[20]**.

6.2.2. La nature des données :

Comme nous l'avons précisé, beaucoup d'algorithmes de clustering s'appuient sur une matrice de similarité ou dissimilarité. Le plus souvent, cette matrice est obtenue à partir des descriptions des données. La nature de ces descriptions (variables qualitatives et/ou quantitatives), détermine alors le choix de la mesure de (dis) similarité utilisée **[20].**

6.2.3. La forme des clusters :

Certaines méthodes de clustering aboutissent à des clusters de formes spécifiques.

6.2.4. Le type de résultats attendus :

La sortie d'un algorithme de clustering peut être, par exemple, une partition (ou pseudo-partition), une fonction ou encore un dendrogramme (arbre hiérarchique). De même, chaque cluster obtenu peut être défini soit par l'ensemble des objets qui le composent, soit par une description relative aux variables initiales. On parle d'une définition en extension, dans le premier cas, et en intension, dans le second. Précisons que la plupart des approches proposent une classification à partir d'une définition en extension des clusters.

Une nouvelle fois, le choix de la méthode de clustering devra être fait en fonction du type de résultat souhaité et donc de l'exploitation envisagée de ce résultat.

6.3. L'exploitation des clusters

La tentation est grande, pour un non-spécialiste, de considérer comme "acquis" le résultat d'un processus de clustering. Autrement dit, les clusters obtenus ne sont généralement ni remis en cause ni évalués en terme de disposition relative,

dispersion, orientation, séparation, densité ou stabilité. Pourtant, il est sans aucun doute utile de distinguer les classes pertinentes obtenues, des autres. De même, cette étape d'analyse permet d'envisager le recours à une autre approche de clustering plus adaptée. Deux situations sont possibles : soit la tache de clustering s'inscrit dans un traitement global d'apprentissage, soit les clusters générés par clustering constituent un résultat final.

Dans le premier cas, l'analyse des clusters obtenus (mesures statistiques de qualité) peut aider à orienter le traitement suivant. Une description des clusters n'est pas nécessaire dans cette situation.

En revanche, dans le cas où le clustering constitue à lui seul un processus global de découverte de classes, l'exploitation des clusters pour une application donnée passe par une description de ces derniers. Lorsque les objets se présentent sous la forme d'une matrice de (dis)similarité, il existe peu de méthodes pour décrire les classes (médoides, k objets représentatifs et mesures de cohésion). Lorsque les objets sont décrits par un ensemble de variables, on peut avoir recours à des méthodes de description **[16].** Des classes (descriptions conceptuelles).

7. Les différentes méthodes de clustering :

On distingue classiquement deux grandes familles de méthodes en clustering**[3]** : les méthodes hiérarchiques et les méthodes par partition .dans le premier cas ,une hiérarchie de clusters est formée de telle maniéré que plus on descend dans la hiérarchie , plus les clusters sont spécifiques a un certain nombre d'objets considères comme similaires.au contraire ,dans le second cas, le résultat fournis est partition de l'espace des objets, c'est-à-dire que chaque objet est associé à un unique cluster.

- Dans le cas du clustering hiérarchique, deux grandes méthodes se distinguent **[22], [23], [24]** :
  1- **les méthodes ascendantes [24]**, qui démarrent avec autant de clusters que d'objets puis fusionnent successivement l'ensemble des clusters selon un certain critère jusqu'à ce que tous les objets soient finalement regroupés dans un unique cluster stocké a la racine de hiérarchie
  2- **les méthodes descendantes [24]**, qui démarrent avec unique cluster regroupant l'ensemble des objets, puis divisent successivement les clusters selon un certain critère jusqu'à ce que tous les objets se trouvent dans des clusters différents stockés aux feuilles de la hiérarchie.

On retrouve donc à ce niveau la différence classique en apprentissage entre généralisation et spécialisation : les méthodes ascendantes démarrent avec une solution tout à fait spécifique aux données, qui est ensuite généralisé à chaque étape, alors que les méthodes descendantes démarrent avec une solution complétement général, qui est ensuite spécialisé à chaque étape.

- Dans le cas du clustering par partition, plusieurs méthodes se distinguent fortement **[22], [23], [24]** :

1- **le clustering statistique** est basé sur l'hypothèse que les données ont été générées en suivant une certaine loi de distribution, le but étant alors de trouver les paramètres de cette distribution, ainsi que les paramètres caché déterminant l'appartenance des objets aux différents composants de cette loi, le clustering basés sur les K-moyennes ,méthodes très souvent utilisés ,en est un cas particulier
2- **le clustering stochastique** consiste à parcourir l'espace des partitions selon certaines heuristiques, et sélectionner celle qui optimise un critère donné.
3- **le clustering basé sur la densité** a pour but d'identifier dans l'espace les zones de fortes densités entourées par des zones de faible densité pour la formation des clusters.
4- **le clustering basé sur les grilles** utilise une grille pour partitionner l'espace de description des objets en différentes cellules, puis identifier les ensembles de cellules denses connectées pour former les clusters.
5- **le clustering basé sur les graphes** consiste à former le graphe connectant les objets entre eux et dont la somme des valeurs des arcs, correspondant aux distances entre les objets, est minimale, puis à supprimer les arcs de valeurs maximales pour former les clusters.
6- **le clustering spectral** consiste à projeter itérativement les objets dans des sous- espace de variance pour séparer les données.

Différentes méthodes hybrides ont également été proposées qui mélangent les caractéristiques des méthodes hiérarchiques et des méthodes par partition, cherchant ainsi à bénéficier des atouts de chaque méthode.

8. Conclusion :

Dans ce chapitre, nous avons définis c quoi la fouille de données, ces déférente taches et nous avons présenté le clustering de manière détaille. Nous allons voir dans le chapitre suivant un état de l'art sur l'analyse de concept formel.

Chapitre III

L'Analyse de Concepts Formels pour le clustering: ETAT DE L'ART

Chapitre 3 :L'Analyse de Concepts Formels pour le clustering : État de l'art

1. Introduction :

Plusieurs méthodes de clustering des documents XML publiées dans la littérature s'appuient sur des techniques différentes. Nous nous intéressons dans ce mémoire aux méthodes basées sur l'analyse de concept formel.

L'Analyse de Concepts Formels est une théorie qui est récemment utilisée pour la fouille de données. À travers ce chapitre nous voulons découvrir cette théorie en tentant surtout de soulever la question de l'applicabilité de l'analyse de concepts formels pour le clustering des documents XML.

2. Les notions de base de l'Analyse de Concepts Formels :

L'analyse de concepts formels(ACF) a été introduite par Wille en 1982 **[25].** L'ACF permet d'introduire des clusters de connaissances et se présente comme une méthode d'apprentissage et de représentation de connaissances. Elle a été utilisée dans divers domaines : psychologie, sociologie, biologie, médecine, linguistique, mathématique, informatique, etc. **[26].**

Pour bien fixer les idées nous devons connaitre c'est quoi un concept formel? D'un point de vue psychologique, un concept formel est une unité de pensé constituée de deux parties, la partie « extension » et la partie « intension». La partie extension recouvre tous les objets du concept formel et la partie intension comprend toutes les propriétés possédées par tous ces objets. En effet les objets et les propriétés jouent un rôle proéminent ensemble avec les différentes relations comme : la hiérarchie entre les concepts (sur-concepts, sous-concept) la relation d'implication entres propretés et la relation d'incidence entre les paires (objets/propriété) (un objet possède une propriété).

L'idée de Wille était d'illustrer dans un premier temps les paires (objets/propriété) ainsi que la relation d'incidence selon une représentation mathématique appelé contexte formel, ce dernier est considéré comme un outil de description des situations élémentaires sous la forme : l'objet « x » possède la propriété « a ». Dans un deuxièmes temps, il s'agira de découvrir les ensembles maximaux des objets satisfaisant un certain de propriétés.

Exemple :

Soit un ensemble d'animaux {lion, rouge-gorge, aigle, lièvre, autruche} par un certaines de leurs propriétés {prédateur, vole, ovipare, mammifère}. Le contexte formel noté C (autrement dit relation binaire) est représenté sous forme d'une table avec lignes les animaux (correspondant aux objets) et en colonnes les propriétés, telle que si l'objet « x_i » vérifie (resp.ne vérifie pas) la propriété « a_j » alors la cellule « c_{ij} » est marquée par une croix (resp.reste vide) **[27]**.

Attribut / Animal	prédateur	Vole	Ovipare	mammifère
Lion	×			×
Rouge-gorge		×	×	
Aigle	×	×	×	
Autruche			×	
Liévre				×

Tableau 3.1 : Représentation du contexte formel C **[27]**.

Pour expliquer la notion du concept formel, nous considérons toutes les propriétés du rouge-gorge {vole, ovipare} et posons-nous la question : quels sont les animaux satisfaisant ces propriétés ? Nous obtenons alors l'ensemble X={aigle, rouge-gorge}. Nous constatons que l'ensemble X est l'ensemble maximal des animaux satisfaisant toutes les propriétés de l'ensemble A={vole, ovipare}.

Il résulte que X est l'ensemble de tous les animaux vérifiant toutes les propriétés de A et A est l'ensemble de toutes les propriétés vérifiées par tous les animaux de X. La paire <X, A> est appelée concept formel. Tandis que X appelé extension et A est appelé intension.

2.1. Hiérarchie entre concepts formels :

Entre les concepts formels, il y a une relation d'ordre partiel c.à.d. la relation « Sous-concept, Sur-concept ».

Etant donné deux concepts formels <X ,A> et <Y,B> .On dit que<X ,A> est un sous-concept de <Y,B> ,(dualement <Y,B> est un sur-concept de <X ,A>) si l'extension de <X ,A> est un sous ensemble de l'extension <Y,B>.c.à.d. X⊆Y(dualement l'intension <X ,A> est un sur-ensemble de l'intension de <Y,B>.c.a.d A⊇B).

Reprenons l'exemple précédent. Etant donné les deux concepts formels suivants :<{aigle}, {prédateur, vole, ovipare}> et <{aigle, rouge-gorge}, {vole, ovipare}>. <{aigle}, {prédateur, vole ,ovipare}> est un sous-concept de <{aigle, rouge-gorge}, {vole, ovipare}>. Dualement <{aigle, rouge-gorge},{vole, ovipare}> est un sur-concept de <{aigle},{prédateur, vole ,ovipare}> .L'extension{aigle} du sous-concept est un sous-ensemble de l'extension {aigle, rouge-gorge} du sur-concept. De la même manière l'intension {prédateur, vole, ovipare} du sous-concept est un sur-ensemble de l'intension {vole, ovipare} du sur-concept. Dans la figure 1, nous présentons la hiérarchie de tous les concepts formels du contexte représenté dans le tableau 1 est à noter que le type de représentation utilisé dans la figure 1 est une représentation condensée. La règle suivante permet de lire les concepts formels.

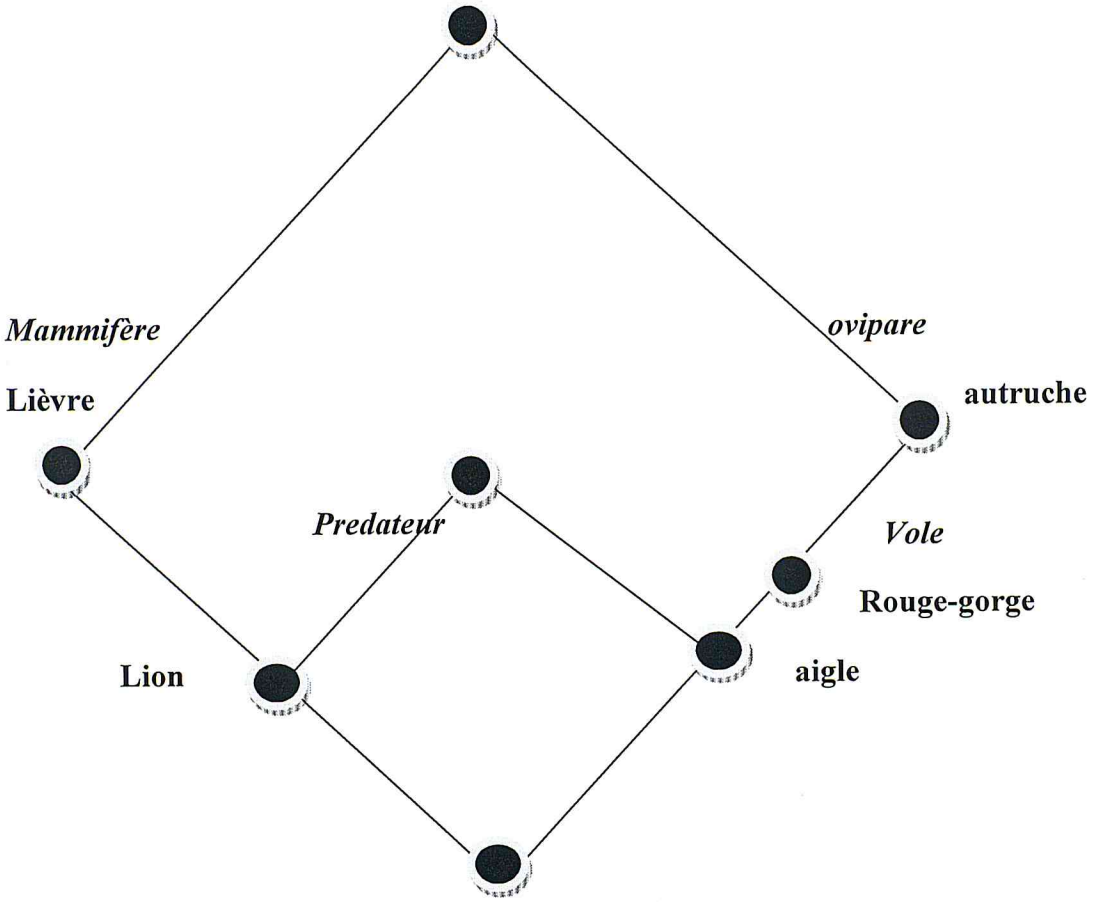


Figure 3.1 : Hiérarchie (treillis) des concepts formels **[27]**.

Cette figure est constituée de nœud et de segments .Elle comporte aussi les noms de tous les objets et toutes les propriétés du contextes .Chaque nœud correspond a un concept formel .La figure peut être lue comme suit : A chaque fois qu'un nœud « n » est étiqueté par la propriété « a », tous les objets descendantes de ce nœud « «n » est étiqueté par un objet « x », « x » est hérité vers le haut et tous les ancêtres de nœud « n » le partage. Ainsi l'extension « X » d'un concept <X,A> correspondant au nœud « n » est obtenue en considérant tous les objets qui apparaissent sur les descendants du nœud « n » dans le treillis et son intention A est obtenue en considérant toutes les propriétés qui apparaissent sur les ancêtres du nœud « n » dans le treillis.

Nous pouvons remarquer que dans cet exemple, l'intension du concept formel sommet correspond à l'ensemble vide, tandis que l'extension de concept formel sommet correspond à l'ensemble de tous les animaux. Nous pouvons trouver toute fois un concept formel sommet différent de l'ensemble vide.

2.2. Relation d'implication entre les propriétés :

L'interprétation de la figure 1 nous renseigne sur diverses relations d'implication entre propriété. On dit que a → b, si « b » un descendant de la propriété « a » or vole est descendant de ovipare alors vole → ovipare .Considérons les implications entre les propriétés déduites du contexte C suivantes :

1- Vole → ovipare

2- mammifère et vole → prédateur et ovipare,

Considérons l'univers des objets U. Nous constatons rapidement que les deux implications (1,2) sont vérifiées dans U, mais elles ne sont pas vérifiées dans U.

Nous citons un contre exemples respectivement :

Une abeille vole mais elle n'est pas ovipare, une chauve-souris est mammifère et elle vole mais elle n'est ni prédateur, ni ovipare, alors la figure 1 devient :

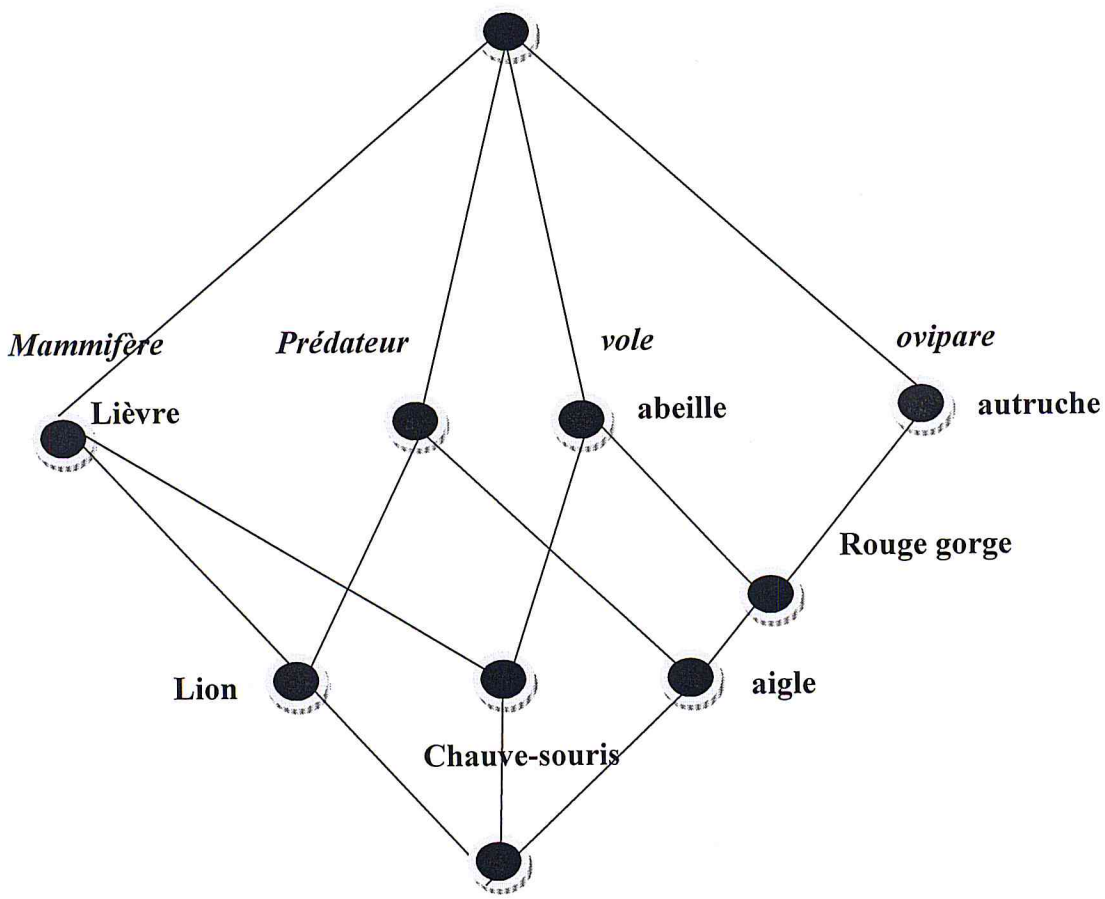


Figure 3.2 : Hiérarchie des concepts formels après rajout de l'animale abeille et chauves-souris **[27]**.

2.3. Rappels Mathématiques :

Dans cette section, nous rappelons quelques définitions nécessaires à notre travail. Ces définitions concernent les ensembles ordonnés, les treillis, les treillis complets et les opérateurs de fermetures. Ces définitions sont extraites de **[28].**

2.3.1. Théorie des treillis : Notions de base :

2.3.1.1. Ensemble ordonné :

Définition 1 : (Relation binaire) Une relation binaire R entre deux ensembles M et N est un ensemble de couples d'éléments (m, n) tels que m ∈ M et n ∈ N, i.e. un sous ensemble de M ×N. (m, n) ∈ R (aussi noté par mRn) signifie que l'élément m est en relation R avec l'élément n. Si M = N, on parle de relation binaire sur M. R^{-1} est la relation inverse de R, i.e. la relation entre N et M telle que $nR^{-1}m \Leftrightarrow mRn$ **[28].**

Définition 2 : (Relation d'ordre (partiel)) Une relation binaire R sur un ensemble E est dite relation d'ordre partiel (ou simplement relation d'ordre) sur E si elle vérifie les conditions suivantes pour tous x,y, z ∈ E **[28]** :

1. (x, x) ∈ R (R est reflexive)
2. si (x,y) ∈ R et x ≠y alors (y, x) ∉ R (R est antisymétrique)
3. si (x, y) ∈R et (y,z) ∈ R alors (x,z) ∈ R (R est transitive)

Une relation d'ordre R est souvent notée par ≤ (R^{-1} est notée par "≥") et on dit que "x est plus petit que y" lorsque x ≤y.

Définition 3 : (Ensemble ordonné) Un ensemble partiellement ordonné (ou simplement ensemble ordonné) est un couple (E,≤) où E est un ensemble et "≤" est une relation d'ordre sur E. Dans un ensemble ordonné (E,≤), deux éléments x et y de E sont dits comparables lorsque x ≤ y ou y ≤ x, autrement ils sont dits incomparables. Pour deux éléments comparables et différents, x ≤y et x ≠ y, on note x < y. Un sous ensemble de (E,≤) dans lequel tous les elements sont comparables est appelé chaîne. Un sous ensemble de (E,≤) dans lequel tous les éléments sont incomparables est appelé anti-chaîne **[28].**

Définition 4 : (Successeur, prédécesseur, couverture) Soient (E,≤) un ensemble ordonné et x,y ∈E. y est dit successeur de x lorsque x < y et il n'existe aucun élément z ∈ E tel que x < z < y. Dans ces cas, x est dit prédécesseur de y et on note x < y. Lorsque x est un prédécesseur de y on dit que x couvre y (et que y est couvert par x). La couverture de x est formée par tous ses successeurs.
Tout ensemble ordonné, (E,≤), peut être représenté graphiquement par un diagramme appelé "diagramme de Hasse" (ou diagramme de couverture) et obtenu comme suit **[28]** :

1. Tout élément de E est représenté par un petit cercle dans le plan
2. Si x, y ∈ E et x < y alors le cercle correspondant à y doit être au-dessus de celui correspondant à x et les deux cercles sont reliés par un segment.

À partir d'un tel diagramme on peut lire la relation d'ordre comme suit : $x < y$ si et seulement s'il existe un chemin ascendant qui relie le cercle correspondant à x à celui correspondant à y.

Définition 5 : (Principe de dualité des ensembles ordonnés) Soit $(E,\leq)$ un ensemble ordonné. La relation inverse "$\geq$" de "$\leq$" est aussi une relation d'ordre sur E. "," est appelée duale de "$\geq$" et (E;,) est appelé le dual de l'ensemble ordonné $(E,\leq)$.
Le diagramme de Hasse de $(E,\geq)$ peut être obtenu à partir de celui de $(E,\leq)$ par une simple réflexion horizontale. De plus, il est possible de dériver les propriétés duales de $(E,\geq)$ à partir des propriétés de $(E,\leq)$ **[28].**

2.3.1.2. Treillis :

Définition 6 : (Majorant, minorant, supremum, infimum) Soient $(E,\leq)$ un ensemble ordonné et S un sous ensemble de E. Un element $a \in E$ est dit majorant de S lorsque $a \geq s \; \forall \; s \in S$. De façon duale, $a \in E$ est dit minorant de S lorsque $a \leq s \; \forall \; s \in S$.

Le plus petit majorant (respectivement minorant) de S, s'il existe, est appelé supremum ou borne supérieure (respectivement infimum ou borne inférieure) de S et noté $\bigvee S$ (respectivement $\bigwedge S$). Dans le cas ou $S = \{x,y\}$, $\bigvee S$ et $\bigwedge S$ sont aussi notés par $x \vee y$ et $x \wedge y$ respectivement.
Dans tout ensemble ordonné, lorsque le supremum (respectivement l'infimum) existe, il est unique **[28].**

Définition 7 : (Treillis, treillis complet) Un treillis est un ensemble partiellement ordonné $(E,\leq)$ tel que $x \vee y$ et $x \wedge y$ existent pout tout couple d'éléments $x,y \in E$. Un treillis est dit complet si$\bigvee S$ et $\bigwedge S$ existent pour tout sous ensemble S de E. En particulier, un treillis complet admet un élément maximal (top) noté par $\top$ et un élément minimal (bottom) noté par $\bot$ Tout treillis fini est un treillis complet **[28].**

Définition 8 : (Demi-treillis) Un ensemble ordonné $(E,\leq)$ est un sup-demi-treillis (respectivement inf-demi-treillis) si tout couple d'éléments $x, y \in E$ admet un supremum $x \vee y$ (respectivement un infimum $x \wedge y$).

2.3.2. Fermeture :

Définition 9 : (Fermeture) On appelle opérateur de fermeture sur un ensemble ordonné, $(E,\leq)$, toute application $\varphi : E \rightarrow E$ qui vérifie les propriétés suivantes pour tout $x, y \in E$ **[28]** :

– $x \leq \varphi(x)$ (φ est extensive),
– si $x \leq y$ alors $\varphi(x) \leq \varphi(y)$ (φ est monotone croissante),
– $\varphi(x) = \varphi(\varphi(x))$ (φ est idempotente).
Un élément $x \in E$ est dit fermé pour φ si et seulement si $x = \varphi(x)$.

Définition 10 :(Système de fermeture) Un système de fermeture (dit aussi système de fermés) sur un ensemble E est un ensemble de parties de E contenant E et fermé

pour l'intersection. Formellement, $\mathfrak{C} \subseteq \mathfrak{B}(E)$ est un système de fermeture si les conditions suivantes sont vérifiées **[28]**:

1. $E \in \mathfrak{C}$
2. si $\mathfrak{A} \subseteq \mathfrak{C}$ alors $\bigcap \mathfrak{A} \in \mathfrak{C}$ (l'intersection de toute suite de fermés de $\mathfrak{A}$ appartient à $\mathfrak{C}$)

Les deux notions, opérateur de fermeture et système de fermeture, sont fortement liées. En effet, tout système de fermeture peut être considéré comme l'ensemble de tous les fermés d'un opérateur de fermeture. Inversement, il est possible de définir un opérateur de fermeture sur tout système de fermeture. Cette relation est exprimée dans le théorème suivant.

Théorème 1 : Si $\mathfrak{C}$ est un système fermé sur E alors l'application $\varphi\,\mathfrak{C}$ suivante $\varphi\,\mathfrak{C} : X \longmapsto \bigcap\{A \in \mathfrak{C} \mid X \subseteq A\}$ définit un opérateur de fermeture sur E ($\varphi\,\mathfrak{C} : E \to E$). Inversement, l'ensemble $\mathfrak{C}_{\varphi} = \{\varphi(x) \mid X \subseteq E\}$ de tous les fermés pour un opérateur de fermeture φ est un système fermé.

La notion de fermeture est aussi liée à la définition des treillis. En effet, l'ensemble des fermés dans un ensemble E étant donné un opérateur de fermeture sur E forme un treillis complet. Inversement, tout treillis complet est isomorphe au treillis des fermés d'un opérateur de fermeture et on a le théorème suivant.

Théorème 2 : Si $\mathfrak{C}$ est un système fermé alors $(\mathfrak{C},\subseteq)$ est un treillis complet avec $\bigwedge X = \bigcap X$ et $\bigvee X = \varphi\,\mathfrak{C}(\bigcup X)\ \forall X \subseteq \mathfrak{C}$. Inversement, tout treillis complet est isomorphe au treillis des fermés d'un opérateur de fermeture.

2.3.3. Connexion de Galois :

Définition 11 : (Connexion de Galois) Soient : $\varphi : P \to Q$ et $\psi : Q \to P$ deux applications entre deux ensembles ordonnés $(P,\leq_P)$ et $(Q,\leq_Q)$ φ et ψ: forment une connexion de Galois entre $(P,\leq_P)$ et $(Q,\leq_Q)$ si elles vérifient les conditions suivantes pour tous $p, p1, p2 \in P$ et $q, q1, q2 \in Q$ **[28]**:

1. si $p1 \leq_P p2$ alors $\varphi(p2) \leq_Q \varphi(p1)$,
2. si $q1 \leq_Q q2$ alors $\psi(q2) \leq_P \psi(q1)$,
3. $p \leq_P \psi(\varphi(p))$ et $q \leq_Q \varphi(\psi(q))$.

2.3.3.1. Connexion de Galois dans un contexte formel

Définition 12 :Soit K un contexte formel. Pour tout $A \subseteq G$ et $B \subseteq M$, on définit **[28]**:

$A' = \{m \in Mj\ o(|G|^2 |M||L|)\ g \in A\ ,\ gIm\}$

$B' = \{g \in Gj \mid m \in B, gIm\}$

Intuitivement, A' est l'ensemble des attributs communs à tous les objets de A et B0 est l'ensemble des objets possédant tous les attributs de B.

Définition 13 : (Relation de "subsomption") Soient (A1,B1) et (A2,B2) deux concepts formels de $\mathfrak{B}$ (G,M, I). (A1,B1) $\leq$ (A2,B2) si et seulement si A1 $\subseteq$ A2 (ou de façon duale B2 $\subseteq$B1). (A2,B2) est dit super-concept de (A1,B1) et (A1,B1) est dit sous-concept de (A2,B2). La relation "$\leq$" est dite relation de subsomption**[28].**

La relation "$\leq$" s'appuie sur deux inclusions duales, entre ensembles d'objets et entre ensembles d'attributs et peut ainsi être interprétée comme une relation de généralisation/spécialisation entre les concepts formels. Un concept est plus général qu'un autre concept s'il contient plus d'objets dans son extension. En contre partie, les attributs partagés par ces objets sont réduits.
De façon duale, un concept est plus spécifique qu'un autre s'il contient moins d'objets dans son extension. Ces objets ont plus d'attributs en commun.

2.3.3.2. Treillis de concepts :

Définition 14 : (Treillis de concepts) La relation "$\leq$" permet d'organiser les concepts formels en un treillis complet ($\mathfrak{B}$ (G,M, I);·) appelé treillis de concepts ou encore treillis de Galois [Birkhoff, 1967] et noté par $\mathfrak{B}$ (G,M, I) ou $\mathfrak{B}$ (K). Le supremum et l'infimum dans $\mathfrak{B}$ (K) sont donnés par :

$$\bigwedge_{j\in J}(A_j, B_j) = \left(\bigcap_{j\in J} A_j, \left(\bigcup_{j\in J} B_j\right)''\right)$$

Formule 3.1 : formule de supremum **[28].**

$$\bigvee_{j\in J}(A_j, B_j) = \left(\left(\bigcup_{j\in J} A_j\right)'', \bigcap_{j\in J} B_j\right)$$

Formule 3.2 : formule de l'infimum **[28].**

Le treillis de concepts est une représentation équivalente des données contenues dans un contexte formel qui met en avant les groupements possibles entre objets et attributs ainsi que les relations d'inclusion entre ces groupements. De plus, la représentation graphique du treillis de concepts, sous la forme d'un diagramme de Hasse, facilite la compréhension et l'interprétation de la relation entre les objets et les attributs d'une part et entre objets ou attributs d'autre part. L'avantage de cette représentation est qu'à partir d'un treillis de concepts il est toujours possible de retrouver le contexte formel correspondant et inversement **[28].**

2.3.4 .Algorithmes de construction de treillis de concepts :

La construction du treillis de concepts d'une relation binaire donnée peut être décomposée en trois parties **[29] :**

1. l'énumération des rectangles maximaux (les fermés),
2. la recherche de la relation d'ordre partiel entre ces rectangles,
3. et la représentation graphique du treillis (construction du diagramme de HASSE correspondant au treillis).

Les deux premières étapes constituent le problème de calcul des concepts d'un treillis de concepts à partir d'un contexte formel et peuvent être exécutées simultanément ou de manière séquentielle (dans l'ordre donné plus haut). Cependant, la troisième étape fait partie du problème de visualisation de graphes. De ce fait, ces deux problématiques sont souvent traitées indépendamment ce qui a donné lieu à deux axes de recherche complémentaires : le premier consiste à proposer des algorithmes de plus en plus performants (complexité, temps d'exécution, occupation mémoire, passage à l'échelle) pour le calcul de treillis de concepts à partir de contextes formels et le deuxième consiste à fournir des outils efficaces pour la visualisation de ces treillis **[29].**

On distingue trois familles d'algorithmes **[29]**: les algorithmes *batch* qui considèrent la totalité du contexte dès le départ, les algorithmes incrémentaux qui considèrent le contexte ligne par ligne et les algorithmes d'assemblage qui répartissent le contexte en deux et calculent les concepts correspondant à chaque moitié puis font l'assemblage.

2.3.4.1. Algorithmes batch :

Les algorithmes *batch* constituent la première génération des algorithmes de construction de treillis. Ils prennent en entrée le contexte formel tout entier et calculent les concepts formels et l'ordre entre ces concepts simultanément ou de manière séquentielle **[29]**.

L'un des premiers algorithmes proposés de cette catégorie est l'algorithme de Chein qui génère les concepts par niveaux. L'algorithme est itératif. Son point de départ est l'ensemble L1 de couples (A,B) représentant les lignes du contexte formel (A contient un seul élément de G et B = A'). A chaque étape i, l'algorithme part d'un ensemble Li et construit les éléments de Li+1. Un élément (A3,B3) de Li+1 est obtenu en combinant deux éléments (A1 ;B1) et (A2,B2) de Li comme suit : A3 = A1 ∪ A2 et B3 = B1∩ B2. Les éléments de Li inclues dans au moins un élément de Li¡1 ne sont pas maximaux et sont donc supprimés. L'algorithme s'arrête lorsque Li+1 contient moins de deux éléments. Les éléments non supprimés après l'arrêt de l'algorithme sont les concepts du contexte formel considéré. Pour illustrer le fonctionnement de cet algorithme, nous considérons l'exemple de contexte formel donné dans le tableau 3.1 avec un renommage des attributs et des objets pour faciliter la lisibilité des concepts lors des étapes d'exécution de l'algorithme. Le contexte considéré est donné dans le tableau 3.2 et les traces d'exécution de l'algorithme sont

données dans le tableau 3.3. Les concepts formels calculés par l'algorithme à partir du contexte formel considéré sont les éléments non barrés dans le tableau 3.3.

Exemple

	A	B	C	d	e	F	g
1	×			×			×
2	×			×			×
3	×			×		×	
4	×			×		×	
5			×		×	×	
6			×		×	×	
7		×			×	×	
8		×			×	×	
9	×				×	×	

Tableau 3.2 : Un exemple d'un contexte formel **[29]**

L1		L2		L3	L4
~~1×adg~~	**(12×adg)	12 × adg		12349×a	∅
2×adg	**(12×adg)	~~13×ad~~	**(134 ×ad)	3456789×f	
3×adf	**(34×adf)	~~134×~~ad	**(1234 × ad)		
4×adf	**(34×adf)	~~19×a~~	**(129 × a)		
5×eef	**(56 ×cef)	1234×ad			
6×eef	**(56 cef)	~~129×~~a	***(12349 ×a)		
7×bef	**(78 ×bef)	34×adf			
8×be f	**(78 ×bef)	~~35×f~~	**(356 × f)		
9×aef		~~356×~~f	**(3567 ×f)		
		~~3567×~~f	**(35678 ×f)		
		~~35678×f~~	**(345678× f)		
		~~39×ef~~	**(349 × af)		
		~~345678×f~~	***(3456789 × f)		
		~~349×ef~~			
		56×cef			
		~~57×ef~~	**(578 × ef)		
		~~578×ef~~	**(5789 × ef)		
		~~5789×ef~~	**(56789 ×ef)		
		56789×ef			
		78×bef			

Tableau 3.3 : Les ensembles de rectangles maximaux calculés par l'algorithme de Chein à partir du contexte formel donné dans le tableau 3.2 **[29]**

Les rectangles barrés correspondent à des rectangles non maximaux. Les rectangles qui les contiennent sont indiqués entre parenthèses et l'itération durant

laquelle ils ont été calculés est donnés par le nombre de *. Par exemple, "1 × adg **(12 × adg)" est interprété comme suit : le rectangle "1 ×adg" n'est pas maximal, il est remplacé par "12 × adg" retrouvé à la deuxième itération de l'algorithme.

2.3.4.2. Algorithmes incrémentaux :

Les algorithmes incrémentaux considèrent le contexte formel ligne par ligne (ou colonne par colonne) et construisent le treillis de concepts par ajouts successifs de ligne ou de colonne tout en conservant sa structure. À une étape k, les concepts formels correspondants au k premières lignes du contexte formel sont calculés. L'ajout de la (k + 1)ème ligne entraîne la modification d'une partie des concepts calculés à l'étape k et l'ajout d'éventuels nouveaux concepts.Conçu ainsi, les algorithmes incrémentaux permettent de gérer les contextes dynamiques, où le nombre d'objets et/ou d'attributs peut évoluer, sans avoir à recalculer le treillis à partir de zéro suite à une modification du contexte **[29]**.

2.3.4.3. Algorithmes d'assemblage :

Les algorithmes d'assemblage constituent une évolution des algorithmes incrémentaux qui généralise le caractère incrémental à des ensembles d'objets/attributs. Ils permettent diviser un contexte formel en deux parties verticalement ou horizontalement puis de calculer le treillis de concepts correspondant à chaque partie et enfin d'assembler les treillis obtenus en un seul. Le seul algorithme connu de cette famille est l'algorithme Divide&Conquer **[29].**

3. L'ACF et le clustering :

L'ACF peut être appliquée pour accomplir les différentes tâches de fouille de données telles que l'association, la classification supervisée et la classification non supervisée.

Il n'existe pas de travaux de clustering des documents XML basé sur l'ACF et nous allons étudier les travaux des documents web parce qu'ils sont proche des documents XML puisque ce sont des documents semi-structurés.

3.1. Les travaux de Yimin et al [30] :

Cette approche propose un nouvel algorithme de clustering K-MeansBCC (K-Means Algorithme Baser sur le treillis de concept)

Les étapes de cette approche sont :

1- la représentation et le prétraitement du texte : dans cette étapes
   - élimine les mots vides (le la ,….)
   - utilise TF×Idf pour calculer le poids de terme dans chaque document

2- La construction du contexte formel d'une collection de texte prétraitée.

3- La construction du concept formel :

- utilisent CopExp pour construire des treillis de concepts (CopExp c'est un logiciel qui prend en entrée un contexte sous forme d'un tableau et il généré les concepts sous forme d'un treillis)
- Calcule le poids de chaque concept

4- Clustering : utilisent l'algorithme kmeansBBC pour extraire les clusters

3.2. Les travaux de Yun et al : [31]

Dans cette approche , proposent une nouvelle méthode basée sur l'analyse de concepts formels (ACF) pour construire les concepts pour facilite la navigation sur la collecte et la localisation des résultats d'intérêt. Après un nouvel algorithme est proposé d'extraire des concepts les plus pertinents pour regrouper les concepts en clusters.

Les étapes de cette approche :

1- la représentation et le prétraitement du texte

2- La construction du contexte formel d'une collection de texte prétraité.

3- La génération du concept formel : **[31]** utilisent le logiciel CopExp pour générer les treillis de concepts.

4- Clustering : Dans cette méthode, les documents sont regroupés selon les concepts extraits du treillis de concepts, Pour sélectionner les concepts les plus pertinents, plusieurs propriétés sont définies pour chaque concept tout d'abord :

- **L'importance d'un concept** : Cette propriété est utilisée pour indiquer l'importance d'un concept.
- **Similarité de Concept :** cette propriété indique la similarité entre deux concepts.
- **Couverture ConceptSet :** La couverture d'un ensemble de concept est utilisée pour contrôler le nombre de nœuds enfants d'un certain nœud.

3.3. Les travaux de Zhiming et al : [32]

Dans cette approche, proposent une méthode de clustering des documents en utilisant la technologie ACF.
[32] définissent les tâche de clustering des documents web en utilisons l'ACF comme suit :

(1) **Choisir mots caractéristiques**: il choisit que les mots caractéristiques pour décrire directement les documents web.

(2) **Construction du contexte formel** : ce contexte décrit la relation entre les documents et les tags

(3) **Traitement de cluster**: Selon le contexte formel construit dans la tâche 2, cette tâche sera divisée en trois étapes, treillis de concept, le diagramme de Hasse optimisation et l'analyse de cluster.

3.4. Les travaux de Nyeint et al : [33]

Proposent une approche de clustering des documents web basé sur Analyse de concept formel .cette approche permet de découvrir l'information utiles et Réorganiser les informations.

Les étapes de cette approche :

1- Le prétraitement des documents :
   - Eliminer les mots vides
   - Compter l'apparition de chaque terme dans chaque document pour mesurer l'importance d'un terme dans un document.

2- Génération d'une matrice de documents et de terme (contexte formel)

3- A Partir de cette matrice on extrait les concepts formels

4- Utilisation de la technologie TF×IDF pour éliminer les concepts les plus similaires.

5- Clustering

4. L'utilisation de l'Analyse de Concepts Formels pour les documents XML :

A partir de l'étude effectuée dans la partie précédente sur les différentes approches de clustering basée sur l'ACF, nous constatons que ces approches sont appliquées sur les documents web, nous allons essayer de discuter son applicabilité sur les documents XML .Nous pensons à appliquer l'ACF pour la fouille des documents XML pour plusieurs raisons. Le tableau ci-dessous illustre les points forts et les points faibles de l'applicabilité de l'ACF pour la fouille des documents semi-structurés **[34]**.

Les points forts	Les points faibles
Une théorie mathématique.	La complexité du treillis de concepts pour les grands contextes : - Taille très grande. - Nombre de concepts élevé. - Treillis illisibles et difficilement manipulables.
Processus d'extraction de connaissance : + Association. + Classification	
Représentation graphique : + Visualisation. + Compréhension et interprétation des relations. + Interaction facile avec l'expert.	
Conceptualisation : + Représenter l'ensemble des concepts. + Ordonner les concepts. + Réduire les concepts.	

Tableau 3.4 : Les points forts et les points faibles de l'applicabilité de l'ACF pour la fouille des documents XML **[34]**.

5. Conclusion :

Les techniques d'ACF permettent d'extraire et de classer des regroupements pertinents objets/attributs à partir de données tabulaires. Ces concepts ordonnés peuvent être vus comme une représentation de la structure des données d'origine.

Dans ce chapitre nous avons d'abord présenté l'analyse de concepts formels de maniéré détaillé afin d'en faciliter la compréhension .Nous avons présenté par la suite les notions mathématiques sous-jacentes a la théorie de l'analyse de concepts formels.

En fin nous avons vue quelques approches existent et l'évolution des travaux pour le clustering des documents web basé sur de l'analyse de concepts formels et son applicabilité sur les documents XML.

Dans le chapitre suivant, nous allons présenter notre approche de fouille dans les documents XML. Cette approche basée sur l'analyse de concepts formels.

Chapitre IV

Une nouvelle approche de clustering des documents XML basé sur ACF

Chapitre 4 : Une nouvelle approche de clustering des documents XML basé sur l'ACF

1. Introduction :

Dans le chapitre précédent, nous avons fait une analyse de l'état de l'art du domaine de clustering des documents XML en nous basant sur l'analyse de concept formel. En prenant comme base cette analyse, nous allons proposer dans ce chapitre une nouvelle approche de clustering des documents XML. Celle-ci prend en considération la structure et/ou le contenu. Elle est basée sur la construction des concepts formels.

2. Présentation générale de l'approche :

Un document XML est composée de structure (tags ou balises) et de contenu, notre approche est appliquée soit sur le contenu seul, ou la structure seule ou le contenu et la structure

Notre approche est constituée de quatre grandes phases dont chacune est composée de différentes étapes. La **figure 4.1** présente l'architecture globale de l'approche.
Les quatre phases sont ;

Phase A : Présentation et prétraitements des documents XML

Phase B : construire le contexte formel
Phase C : Génération des concepts formels.
– générer des concepts formels à partir du contexte formel
Phase D : Clustering
- Application d'un algorithme de *clustering* sur les concepts générer en se basent sur les travaux de Yimin et al **[30]**. .

Dans ce qui suit, nous allons détailler les déférentes phases et étapes :

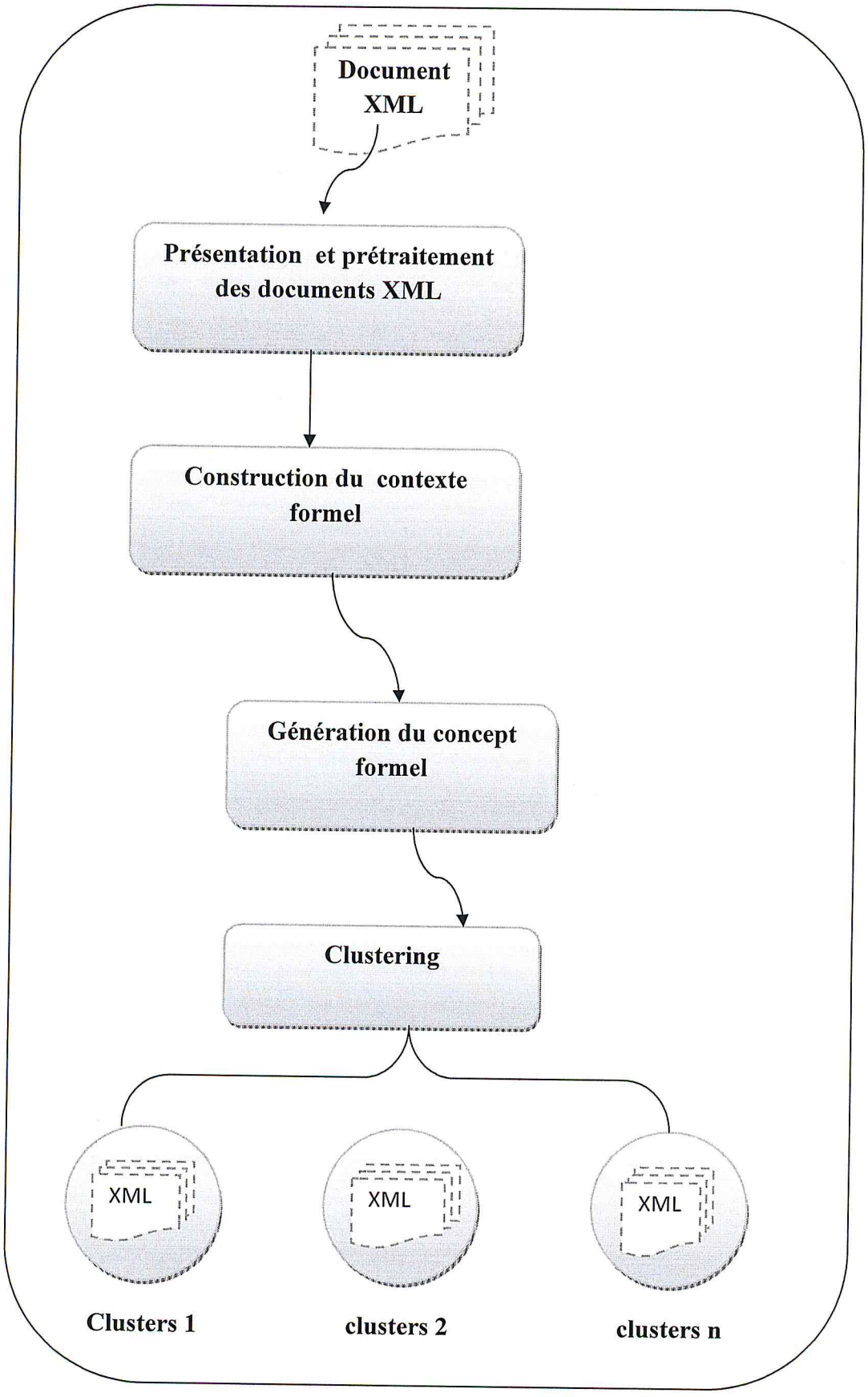

Document XML
Présentation et prétraitement des documents XML
Construction du contexte formel
Génération du concept formel
Clustering
XML
XML
XML
Clusters 1
clusters 2
clusters n

Figure 4.1 : les phases de notre approche

2.1. Présentation et prétraitements des documents XML :

Cette étape permet de présenter un document XML sous forme d'un vecteur à l'aide d'un parseur SAX (simple API pour XML), ensuite on traite ce vecteur de présentation.

2.1.1. Le contenu:

Nous extrayons le contenu des documents XML et nous le présentons par un vecteur. Par exemple doc1.XML= terme1, terme2, terme3..... comme l'indique la **Figure 4.3**.

Le prétraitement du contenu d'un document XML consiste principalement à appliquer une analyse linguistique sur le contenu (Voir figure 4.4) :

- nous éliminons les mots vides par exemple : le, la, un, en.....
- nous éliminons les mots alphanumériques :123sport,.....

2.1.2. La structure :

Nous extrayons la structure des documents XML en la présentant par un vecteur de chemin (un chemin est une suite de tags visités à partir de tags initiaux) . Par exemple doc1.XML=terme1, terme1.terme2, terme1.terme3..... comme l'indique la **Figure 4.3**

Le prétraitement de la structure d'un document XML consiste principalement à réduire le nombre de chemins générés, en essayant de diminuer successivement le nombre de balises. Le filtrage sur les balises utilise une connaissance sémantique de ces balises. Nous pouvons, optionnellement (Voir figure 4.4) :

- éliminer les mots alphanumériques :123 sport,...
- éliminer les doublons
- éliminer les chemins dupliqués.

2.1.3. Le contenu et la structure :

Nous extrayons toutes les tags et le contenu des tags d'un document XML et nous le présentons comme l'indique la **Figure 4.3.** Nous appliquons un prétraitement linguistique (Voir **figure 4.4).**

Voici un exemple de documents XML

```
<?xml version="1.0" encoding="ISO-8859-15"?>

<personne>
    <nom>Bousmaha</nom>
    <prénom>Sara</prénom>
    <adresse>
        <numéro>3</numéro>
        <rue>cité Moumen citué à boufarik la willaya de blida</rue>
        <ville>blida</ville>
        <codePostal>3200</codePostal>
    </adresse>
```

Figure 4.2 : Un exemple d'un document XML « personne »

Présentation du document XML :

Le contenu	La structure
Doc1.xml=Bousmaha,Sara,3,cité,Moumen,citué,a,boufarik,la,willaya,de,blida,blida,32 00	Doc1.XML=personne.nom,personee.prenom,personne .adresse,personne .aderesse.num, personne .aderesse.rue, personne .aderesse.ville, personne .aderesse.CodePostal

Contenu et structure
Doc1.xml=personne,nom,Bousmaha,prenom,Sara,adresse, numero,rue,cité,Moumen,a,boufarik,la,willaya,de,blida,ville,blida,CodePostal,3200

Figure 4.3 : Le document XML « «personne » avant le prétraitement

Le prétraitement du document XML :

Le contenu	La structure
Doc1.XML=Bousmaha, Sara,cité,Moumen,citué,boufarik ,willaya,blida	Doc1.XML=personne.nom,per sonee.prenom,personne .adres se,personne .aderesse.num, personne .aderesse.rue, personne .aderesse.ville, personne .aderesse.CodePostal

Contenu et la structure
Doc1.xml=personne,nom, Bousmaha,prenom,Sara,a dresse,numero,rue,cité,M oumen,boufarik,willaya,b lida,ville,CodePostal

Figure 4.4 : Le document XML « «personne » après le prétraitement

2.2. Construction du contexte formel :

À la fin de la phase de présentation et de prétraitements des documents XML, l'ensemble des termes (ou chemins) de chaque document est mappé dans une matrice booléenne. La matrice de termes (ou chemins) est définie pour calculer la fréquence d'apparition des termes (ou chemins) dans chaque document XML. Cette matrice s'appelle le contexte formel.

Le contexte formel est un triplet (D,N,R), dans D={ *d1, d2 ,d3 ,..., dj*} l'ensemble des documents XML de toute la collection ,N={N1,N2,N3,……Nj} est l'ensemble des termes de toute la collection, R la relation binaire entre D et N.

Les éléments de D sont les objets, N sont les attributs et R est l'incident du contexte (D, N, R).

Le contexte formel est une matrice des relations, elle est définie pour calculer la fréquence d'apparition des termes (ou chemins) dans chaque document XML tel que les m lignes sont les documents et les n colonnes sont les termes (ou chemins) de toute la collection XML.

2.2.1. Le contenu :

	Bousmaha	mouhamed	cité	Moumen	citué	boufarik	El bayedh	blida
Doc1.xml	1	0	1	1	0	1	0	1
Doc2.xml	0	1	1	1	0	0	0	1
Doc3.xml	0	0	1	0	1	0	1	0
Doc4.xml	0	0	0	0	1	0	1	0

Tableau 4.1 : Le contexte formel pour le contenu d'un document XML

2.2.2. La structure :

	Personne	Personne.Nom	Personne.prenom	Personne.adresse
Doc1.xml	1	0	1	1
Doc2.xml	0	1	0	0
Doc3.xml	0	1	1	0
Doc4.xml	1	0	0	0

Tableau 4.2 : Le contexte formel pour la structure d'un document XML

2.2.3. Le contenu et la structure :

	personne	nom	prenom	Bousmaha	Sara
Doc1.xml	1	1	1	1	1
Doc2.xml	0	1	1	0	0
Doc3.xml	0	0	0	1	0
Doc4.xml	0	0	0	1	0

Tableau 4.3 : Le contexte formel pour le contenu et la structure d'un document XML

La prochaine phase de l'approche consiste à extraire les concepts formels à partir du contexte formel afin d'identifier des concepts des documents similaires.

2.3. Construction du treillis de concepts :

On traite le contexte formel pour arriver au treillis de concepts, et ses concepts sont extraits pour former la liste des concepts.

2.3.1. Le contenu :

Concept	Intent	Extent
C0	{}	{Doc1.xml,Doc2.xml,....}
C1	{bousmaha,sara,cuité,....}	{}
C2	{bousmaha,cité,Moumen,boufarik,blida}	{Doc1.xml}
C3	{el bayedh,citué}	{Doc3.xml,Doc4.xml}
C4	{mouhamed,cité ;moumen,blida}	{Doc1.xml,doc2.xml ,Doc3 . xml}

Tableau 4.4 : le tableau du concept formel pour le contexte textuel

2.3.2. La structure :

Concept	Intent	Extent
C0	{}	{doc1.xml,doc2.xml,doc3.xml,doc4.xml}
C1	{personne,personne.nom,personne.prenom,personne.adresse}	{}
C3	{personne. prénom}	{doc1.xml,doc3.xml}
C4	{personne.adresse}	{doc2.xml,doc2.xml}
C5	personne	{doc1.xml,doc4.xml}

Tableau 4.5 : le tableau du concept formel pour le contexte structurel

2.3.3. Le contenu et La structure :

Concept	intent	Extent
C0	{}	{doc1.xml,doc2.xml,doc3.xml,doc4.xml}
C1	{personne ,nom,prenom,Bousmaha,Sara}	{}
C2	{nom ,prenom}	{doc1.xml, doc2.xml}
C4	{Bousmaha}	{doc1.xml,doc4.xml}

Tableau 4.6 : le tableau du concept formel pour le contexte formel textuel et Structurel

L'intention de concept C0 est Φ qui n'a pas la capacité de distinguer les documents, l'extension de concept de C1 est Φ qui ne peut pas contenir l'objet document, Le concept C2 (du **Tableau 4.4**) ne contient qu'un objet donc leur capacité est relativement faible. En conséquence les concepts C3 et C4 (du **Tableau 4.4**) sont retenus pour représenter les documents.

Le treillis de concepts représente la hiérarchisé des documents. Si on a la relation d'ordre entre deux concepts et le concept représente certains documents, cela signifie qu'un document comprend l'autre document. À savoir tous les termes inclus dans le document sont également inclus dans les autres documents. En d'autres termes, un document comprend bien les autres documents.

2.4. Clustering:

Cette étape est composée de 3 phases:

1- Calculer le poids de chaque document par rapport à chaque concept, en se basant sur les travaux de Yimin et al **[30]**.
2- Calculer la similarité des documents XML, en se basant sur les travaux de Yimin et al **[30]**.
3- Appliquer l'algorithme K-MeansBCC .

2.4.1. Calcul le poids de chaque concept par rapport à chaque document :

Après la construction du contexte formel, on utilise la mesure TF×IDF (fréquence d'un terme× fréquence inverse de document) **[30]** pour calculer le poids d'un terme dans un document XML :

$$w_{i,j} = TF \times IDF$$

Formule 4.1 : calcule de poids d'un terme dans un document

$$IDF = \log \frac{D}{J}$$

Formule 4.2 : fréquence inverse de document

Avec :

- $w_{i,j}$ représente le poids d'un terme dans un document
- **TF** représente le nombre d'occurrences de terme dans un document .cette valeur représente l'importance locale du terme.
- **IDF** représente l'importance globale du terme, cette valeur est calculée à l'aide :
  - **D** représente le nombre de documents dans la collection
  - **J** représente le nombre de documents dans la collection contenant le terme

Ensuite on calcule le poids d'un concept dans document XML, on utilise la formule **[30]** suivante :

$$CV_{i,j} = \frac{1}{n_j} \sum_{k=1}^{m} p_{i,k} . w_{i,k}$$

Formule 4.3 : calcule le poids d'un concept dans un document

$$p_{i,k} = \begin{cases} 1, t_k \text{ appartient à l'intention de } C_i \\ 0, t_k \text{ n'appartient pas à l'intention de } C_j \end{cases}$$

Avec :

- $CV_{i,j}$ représente le poids d'un concept dans un document XML.
- $w_{i,k}$ représente le poids de terme dans un document XML.

2.4.2. Calcul de la similarité des documents XML :

On calcule la similarité entre deux documents XML par la formule suivante **[30].**

$$sim(d_1, d_2) = \sum_{j=1}^{s} \sum_{i=1}^{s} v_{1,i} v_{2,j} sim(C_i, C_j)$$

Formule 4.4 : la similarité entre deux documents

Avec :

- $sim(d_1, d_2)$ représente la similarité entre le document d_1 et le document d_2.
- $v_{1,i}$ représente le poids de concept C_i dans le document d_1.
- $sim(C_i, C_j)$ représente la similarité entre deux concepts, elle est définie par la formule suivante :

$$sim(C_i, C_j) = \left| \frac{I_{C_i} \cap I_{C_J}}{I_{C_i} \cup I_{C_j}} \right|$$

Formule 4.5 : la similarité entre deux concepts **[30].**

Avec :

- $sim(C_i, C_j)$ représente la similarité entre deux concepts.
- I_{C_i} représente intension d'un concept C_i

2.4.3. Appliquer l'algorithme K-MeansBCC :

Nous appliquons l'algorithme K-MeansBCC (K-Means Algorithm basé sur le treillis de concepts) pour extraire des clusters **[30]**.

A – **Paramètre** : le nombre K de clusters

B – **Entres** : un corpus de documents XML d_1, d_2,.....d_n

1- Le choix des K centre initiaux $C_1, C_2, \ldots C_k$

2- Pour chacun des documents affecter le premier groupe i dont le centre C_i le plus similaire (la similarité maximale)

3- Si aucun document ne change de groupe donc arrêt et sortir les groupes

4- Calculer les nouveaux centres pour tous i, C_i est la moyenne des éléments de groupes.

5- Aller a 2

Figure 4.5 : l'algorithme K-meansBBC

L'algorithme de la **Figure 4.5** nous a aidés à générer les clusters :

Tous d'abord nous initialisons le nombre de clusters k, ensuite le choix aléatoirement des centres initiaux à partir de la collection des documents XML prétraités. Après nous calculons la similarité entre les centres et les autres documents et nous affectons le document au centre le plus similaire.

Nous calculons les nouveaux centres. Si les groupes sont les mêmes et ne changent pas on arrête sinon l'opération est répétée dès le début.

3. Conclusion

Dans ce chapitre nous avons exposé les différentes phases constituant notre nouvelle approche de fouille dans les documents XML et plus précisément la fouille dans les instances des documents en prenant en considération la structure et en se basant sur l'ACF.

L'approche nécessite une phase de validation sur des corpus de documents XML. Pour ce faire, nous avons développé un prototype d'évaluation. Dans le chapitre suivant, nous allons présenter le prototype développé ainsi que la validation sur les corpus XML.

Chapitre V

Expérimentation et test

1. Introduction :

Dans ce chapitre, nous allons valider notre approche proposée sur le clustering des documents XML .Nous présentons les déférents outils de développements choisis pour valider notre approche et les collections de tests.

2. Environnement de développement Eclipse IDE:

Eclipse IDE est un environnement de développement libre permettant de créer des programmes dans de nombreux langages de programmation (Java, C++, PHP...). C'est l'outil que nous avons utilisé pour programmer.

2.1. Le langage de programmation JAVA

Pour la réalisation de notre projet, nous avons utilisé le langage JAVA qui est un langage de programmation orienté objet développé par SUN. Le choix de ce langage est dû aux avantages suivants : **[35]**

- ✓ Java est un langage orienté objet
- ✓ Java est extensible à l'infini
- ✓ Java est un langage à haute sécurité
- ✓ Java est un langage simple
- ✓ Java est portable

3. Implémentation :

3.1. L'API SAX :

3.1.1. Présentation :

SAX (Simple API for XML) est basé sur un modèle événementiel, ce qui signifie que l'analyseur appelle automatiquement une méthode lorsqu'un événement est détecté dans le fichier XML **[36]**.

Un événement signifie, par exemple, détection d'une balise ouvrante, de la fin du document etc…

Voici les 5 évènements détectés par SAX ainsi que les méthodes qui sont appelées :

- Détection d'une balise de début → ***startElement()***
- Détection d'une balise de fin → ***endElement()***
- Détection de données entre deux balises → ***characters()***
- Début du traitement du document XML → ***startDocument()***
- Fin du traitement du document XML → ***endDocument()***

3.1.2. Quand utiliser SAX :

- Document XML volumineux
- Lecture rapide et efficace
- Economie de mémoire car SAX ne construit pas une représentation en mémoire de la structure en arborescence des fichiers XML lus.

SAX gère les fichiers XML comme un flux de données, on accède aux données quand elles arrivent, mais on ne peut pas revenir en arrière ou sauter à une position dans le fichier. En général, SAX est efficace uniquement pour lire des données pour que l'application travaille avec.

4. Les collections de tests :

Nous allons évaluer notre approche sur deux collections XML disponibles gratuitement sur internet. Les collections sont «*Wikipédia XML Corpus*2».et «*DBLP Computer Science Bibliography* ».

4.1. Wikipédia XML Corpus :

La collection est le corpus Wikipédia qui est composé essentiellement de 8 collections qui correspondent à 8 différentes langues **[37]:** Anglais, Français, Allemand, Hollandais, Espagnol, Chinois, Arabe et Japonais.

Nous allons utiliser le corpus textuel Français **[37]** avec les cinq types : *départements*, *foot*, *peintres*, *références* et *rivières*.

4.2. DBLP Computer Science Bibliographie :

La deuxième collection des documents XML est générée à partir la base de données DBLP contenant les références bibliographiques des publications scientifiques. DBLP est constituée de plus de 1200 000 publications en Anglais de différents types **[38]** : *article*, *inproceedings*, *proceedings*, *book*, *incollection*, *phdthesis*, *mastersthesis* et *www*.

Pour cette collection de documents XML, les cinq types utilisés sont : *articles*, *inproceedings*, *phdthesis*, *proceedings* et *www*.

5 . Test :

La fenêtre principale du prototype contient deux onglets : « Opération » et « Outil »

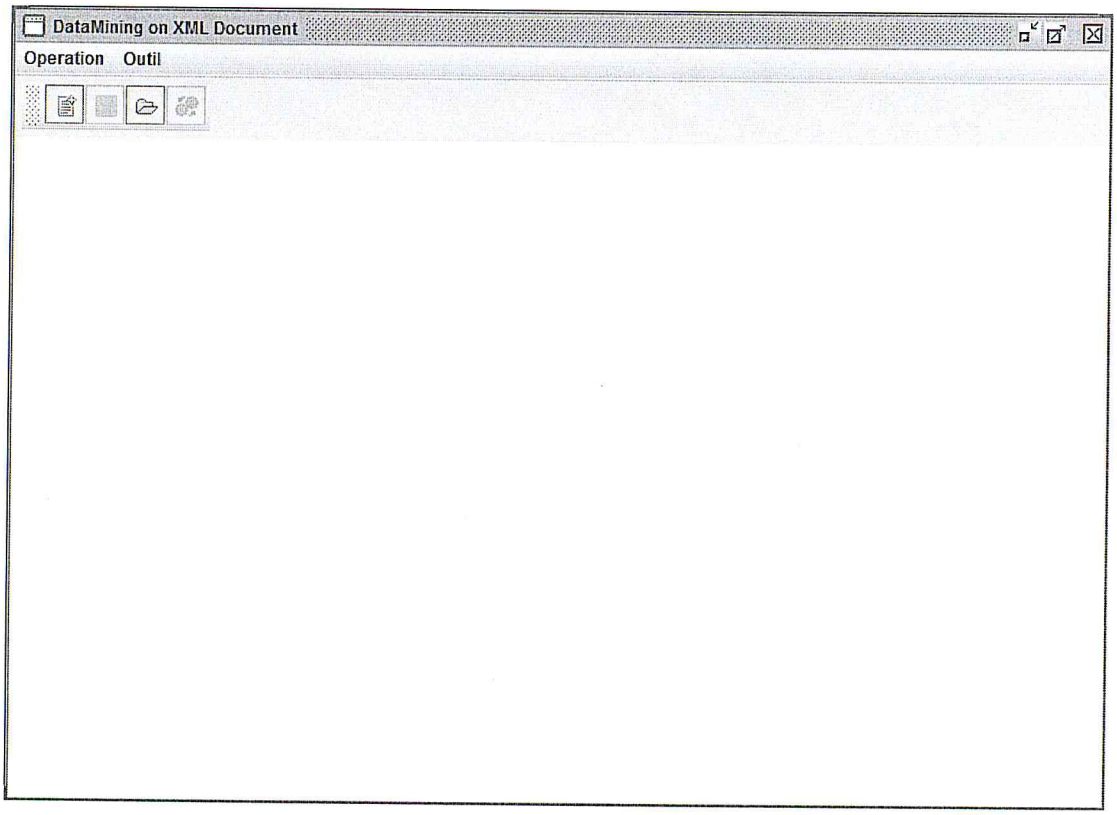


Figure 5.1 La fenêtre principale

La fenêtre « Opération » permet de :

- Importer les documents XML
- Générer un contexte formel
- Calculer TF×IDF
- Générer le concept formel
- Calculer les clusters

La fenêtre « Outil » permet de :

- Choisir si on veut faire le clustering par contenu et/ou structure

Les figures suivantes représentent les différentes fenêtres de notre système :

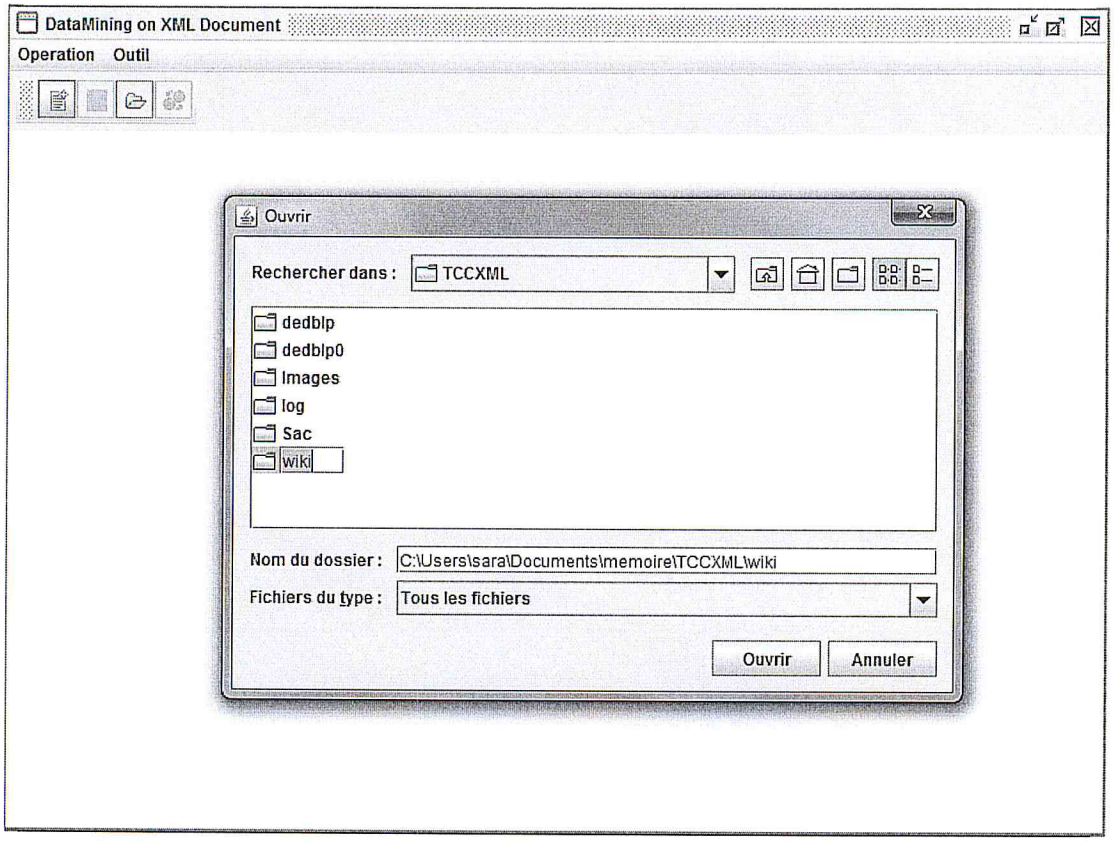


Figure 5.2 : La fenêtre d'importe les documents XML

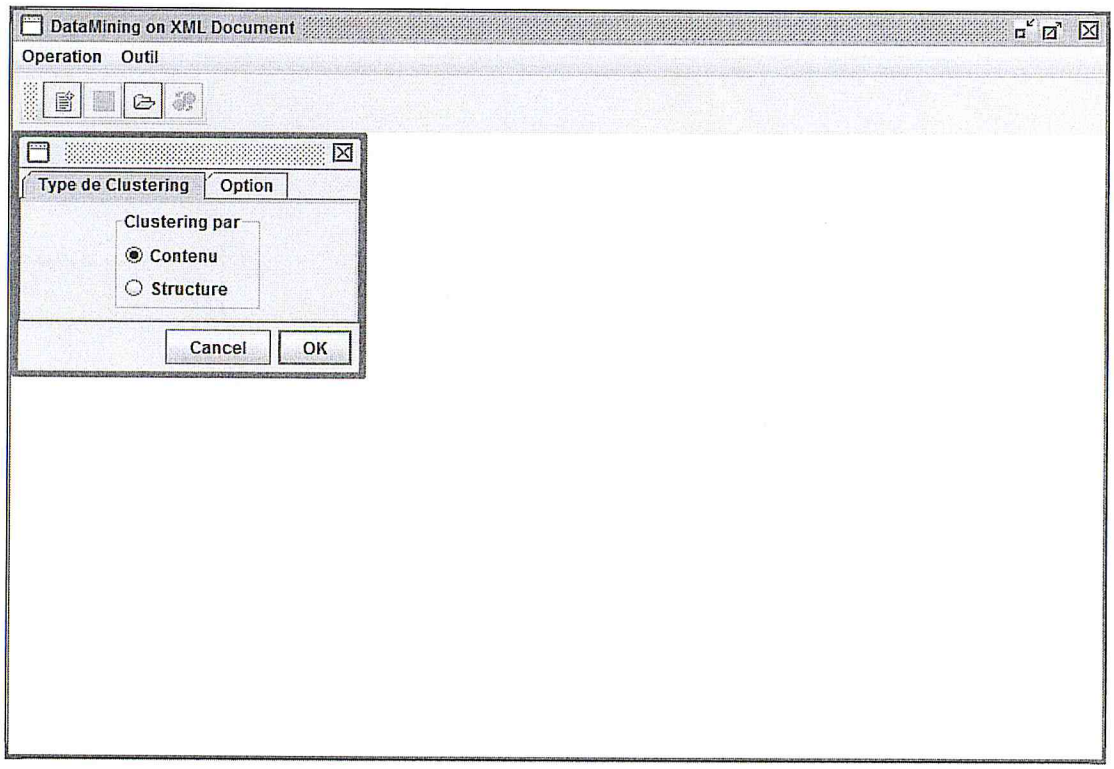
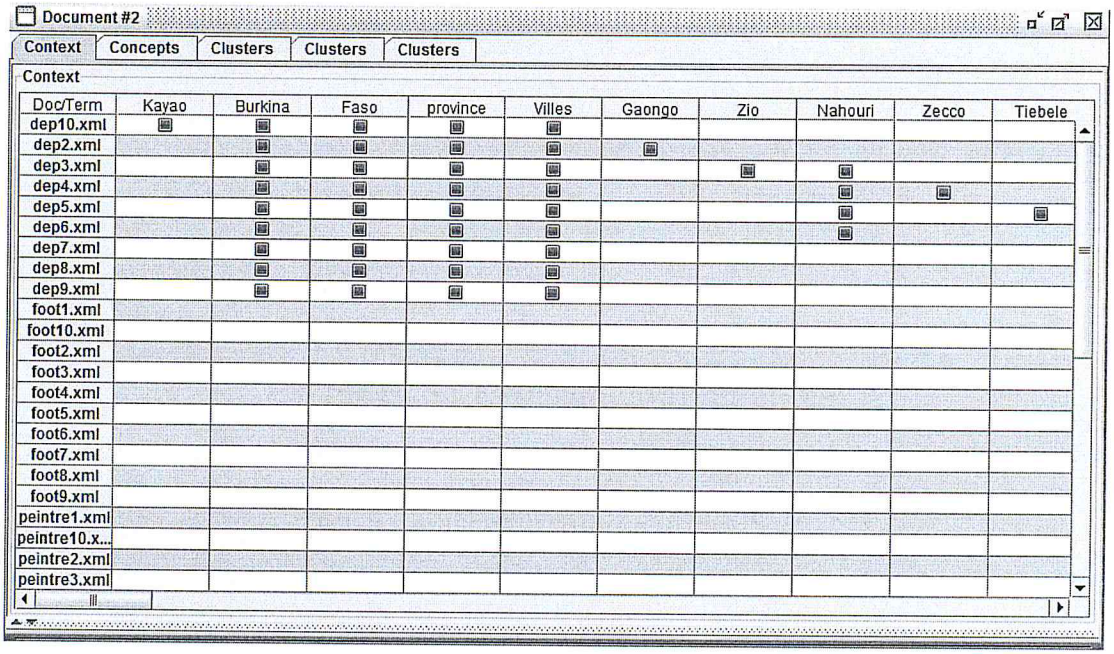


Figure 5.3 : La fenêtre de choix

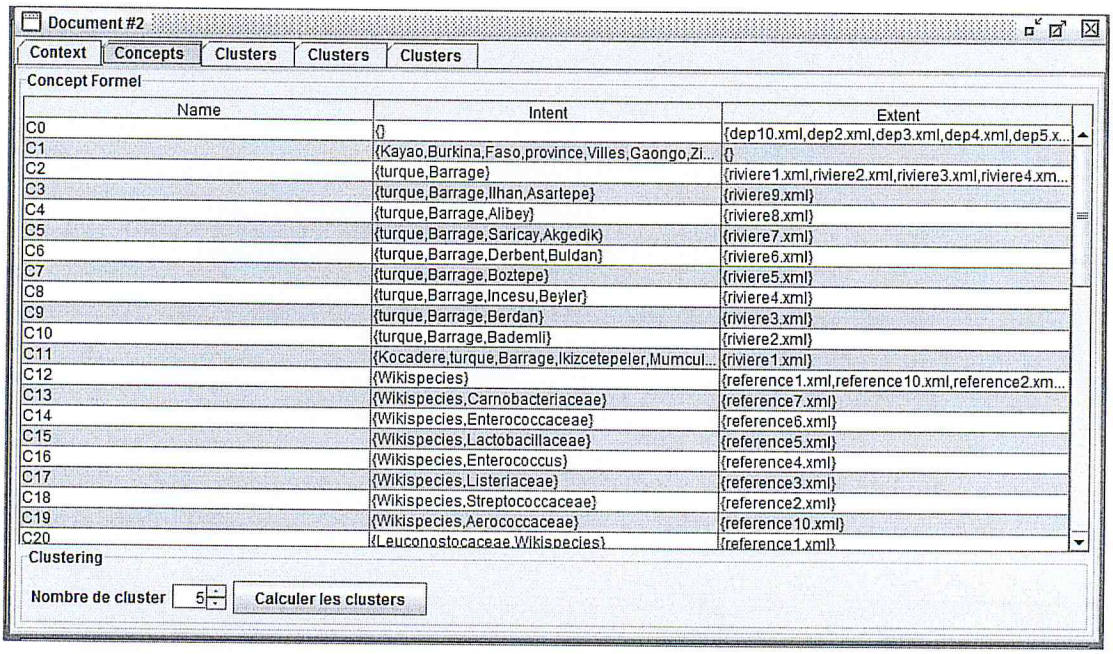
La fenêtre ci-dessous permet d'afficher le contexte formel.



Doc/Term	Kayao	Burkina	Faso	province	Villes	Gaongo	Zio	Nahouri	Zecco	Tiebele
dep10.xml	×	×	×	×	×					
dep2.xml		×	×	×	×	×				
dep3.xml		×	×	×	×		×	×		
dep4.xml		×	×	×	×			×	×	
dep5.xml		×	×	×	×			×		×
dep6.xml		×	×	×	×			×		
dep7.xml		×	×	×	×					
dep8.xml		×	×	×	×					
dep9.xml		×	×	×	×					
foot1.xml										
foot10.xml										
foot2.xml										
foot3.xml										
foot4.xml										
foot5.xml										
foot6.xml										
foot7.xml										
foot8.xml										
foot9.xml										
peintre1.xml										
peintre10.x...										
peintre2.xml										
peintre3.xml										

Figure 5.4 : La fenêtre d'affichage du contexte formel pour le contenu

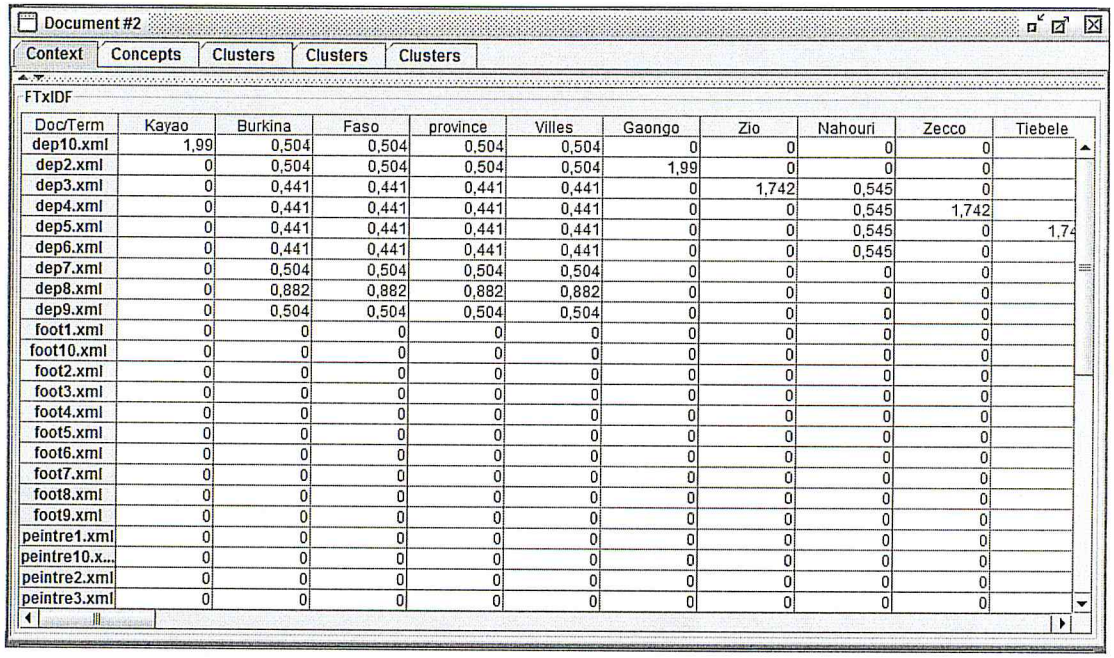
La fenêtre ci-dessous permet de générer les concepts formels et le choix de k (le nombre de cluster).



Name	Intent	Extent
C0	{}	{dep10.xml,dep2.xml,dep3.xml,dep4.xml,dep5.x...
C1	{Kayao,Burkina,Faso,province,Villes,Gaongo,Zi...	{}
C2	{turque,Barrage}	{riviere1.xml,riviere2.xml,riviere3.xml,riviere4.xm...
C3	{turque,Barrage,Ilhan,Asartepe}	{riviere9.xml}
C4	{turque,Barrage,Alibey}	{riviere8.xml}
C5	{turque,Barrage,Saricay,Akgedik}	{riviere7.xml}
C6	{turque,Barrage,Derbent,Buldan}	{riviere6.xml}
C7	{turque,Barrage,Boztepe}	{riviere5.xml}
C8	{turque,Barrage,Incesu,Beyler}	{riviere4.xml}
C9	{turque,Barrage,Berdan}	{riviere3.xml}
C10	{turque,Barrage,Bademli}	{riviere2.xml}
C11	{Kocadere,turque,Barrage,Ikizcetepeler,Mumcul...	{riviere1.xml}
C12	{Wikispecies}	{reference1.xml,reference10.xml,reference2.xm...
C13	{Wikispecies,Carnobacteriaceae}	{reference7.xml}
C14	{Wikispecies,Enterococcaceae}	{reference6.xml}
C15	{Wikispecies,Lactobacillaceae}	{reference5.xml}
C16	{Wikispecies,Enterococcus}	{reference4.xml}
C17	{Wikispecies,Listeriaceae}	{reference3.xml}
C18	{Wikispecies,Streptococcaceae}	{reference2.xml}
C19	{Wikispecies,Aerococcaceae}	{reference10.xml}
C20	{Leuconostocaceae,Wikispecies}	{reference1.xml}

Figure 5.5 : La fenêtre d'affichage du concept formel

La fenêtre ci-dessous permet d'afficher le poids de chaque terme.



Doc/Term	Kayao	Burkina	Faso	province	Villes	Gaongo	Zio	Nahouri	Zecco	Tiebele
dep10.xml	1,99	0,504	0,504	0,504	0,504	0	0	0	0	
dep2.xml	0	0,504	0,504	0,504	0,504	1,99	0	0	0	
dep3.xml	0	0,441	0,441	0,441	0,441	0	1,742	0,545	0	
dep4.xml	0	0,441	0,441	0,441	0,441	0	0	0,545	1,742	
dep5.xml	0	0,441	0,441	0,441	0,441	0	0	0,545	0	1,74
dep6.xml	0	0,441	0,441	0,441	0,441	0	0	0,545	0	
dep7.xml	0	0,504	0,504	0,504	0,504	0	0	0	0	
dep8.xml	0	0,882	0,882	0,882	0,882	0	0	0	0	
dep9.xml	0	0,504	0,504	0,504	0,504	0	0	0	0	
foot1.xml	0	0	0	0	0	0	0	0	0	
foot10.xml	0	0	0	0	0	0	0	0	0	
foot2.xml	0	0	0	0	0	0	0	0	0	
foot3.xml	0	0	0	0	0	0	0	0	0	
foot4.xml	0	0	0	0	0	0	0	0	0	
foot5.xml	0	0	0	0	0	0	0	0	0	
foot6.xml	0	0	0	0	0	0	0	0	0	
foot7.xml	0	0	0	0	0	0	0	0	0	
foot8.xml	0	0	0	0	0	0	0	0	0	
foot9.xml	0	0	0	0	0	0	0	0	0	
peintre1.xml	0	0	0	0	0	0	0	0	0	
peintre10.x...	0	0	0	0	0	0	0	0	0	
peintre2.xml	0	0	0	0	0	0	0	0	0	
peintre3.xml	0	0	0	0	0	0	0	0	0	

Figure 5.6 : La fenêtre d'affichage de TF×IDF

La fenêtre ci-dessous permet d'afficher les différents clusters :

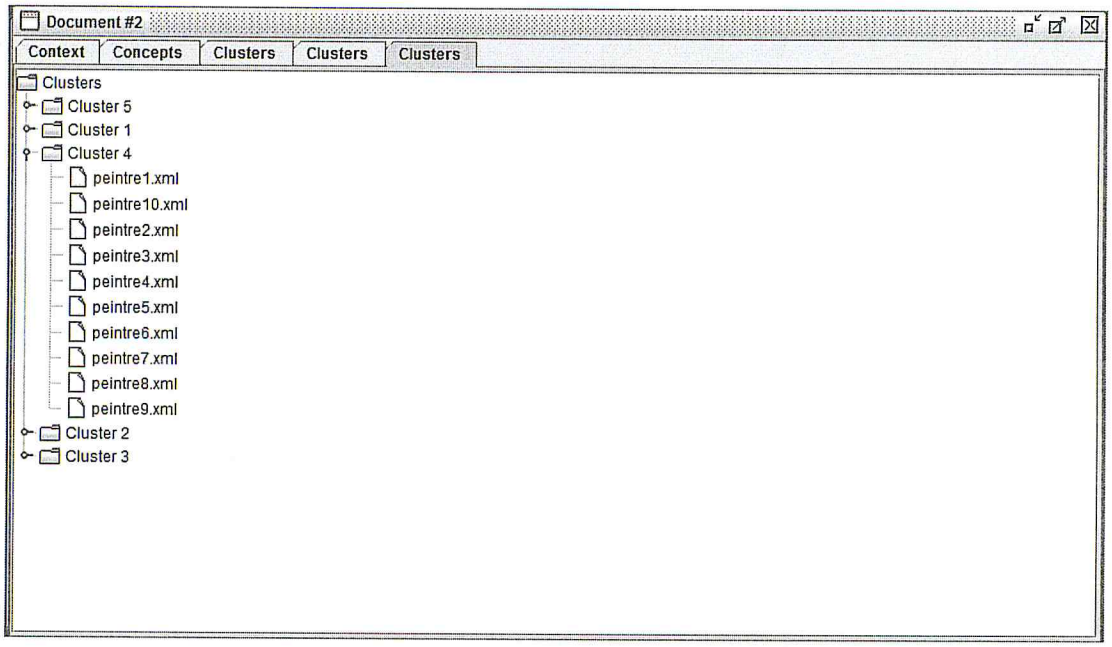


Figure 5.7 : La fenêtre d'affichage des clusters

6. Les mesures d'évaluation :

Le rappel (*Recall*), la précision (*Precision*) et la *F-mesure* (*F-measure*) sont utilisés pour évaluer les résultats obtenus. Ces mesures sont fréquemment utilisées pour évaluer la qualité de la classification en fouille de documents. Le rappel R et la précision P sont définis par les formules suivantes **[39]** :

$$p = \frac{\sum_i a_i}{\sum_i a_i + \sum_i b_i} \quad , \quad R = \frac{\sum_i a_i}{\sum_i a_i + \sum_i c_i}$$

Formule 5.1 : Les mesures d'évaluations

Pour un cluster construit C_i **[39]**, **[40]** :

- a_i (TP : *True Positive*) est le nombre de documents XML qui sont correctement attribués au cluster C_i.
- b_i (FP : *False Positive*) est le nombre de documents XML qui sont dans le cluster C_i pourtant ils ne doivent pas être membres.
- c_i (FN : *False Negative*) est le nombre de documents XML qui doivent être membre du cluster C_i bien qu'ils ne se trouvent pas.

Une précision élevée signifie une justesse élevée du *clustering* alors qu'un rappel faible signifie qu'il y a plusieurs documents XML qui ne sont pas dans le cluster approprié **[39]**. Une précision et un rappel élevés indiquent que la qualité du *clustering* est excellente **[39]**.

F-mesure (*F1*) combine les mesures de précision et de rappel pour essayer d'assigner chaque classe calculée à une classe prédéfinie de telle sorte que deux classes calculées ne peuvent pas être assignées à la même classe prédéfinie **[40]**. La *F-mesure* mesure un équilibre entre la précision et le rappel. Elle calcule la moyenne harmonique des deux précédentes valeurs **[42]** :

$$F1 = \frac{2 \times p \times R}{p + R}$$

Formule 5.2 : La formule de F-Mesure

7. Résultat :

Le tableau ci-dessous représente les résultats de deux collections (DELP et Wikipédia, qui contiennent 50 documents XML, avec K=5

Vecteur de caractéristiques	Collection	Précision	Rappel	F-measure
Le contenue seul	DBLP	0,83	0,78	0,80
La structure seule		0,83	0,83	0,83
Le contenue et la structure		0,63	0,63	0,63
Le contenue seul	Wikipédia	1	1	1
La structure seule		0,83	0,78	0,80
Le contenue et la structure		1	1	1

Tableau 5.1 : Résultat de clustering des documents XML

Comme l'indique le **tableau 5.1**, selon les mesures d'évaluation nous pouvons dire qu'avec notre approche les résultats de *clustering* obtenus sont de bonne qualité. Le rappel et la précision ont atteint de bonnes valeurs pour les deux collections.

8. Conclusion :

Dans ce chapitre nous avons met en œuvre notre système, nous avons montré les différentes étapes qui représentent le déroulement de processus de l'approche, et nous avons évalué notre approche en utilisant les deux collections : DBLP en Anglais et *Wikipédia* en Français.

Conclusion générale

Conclusion générale :

Nous procédons dans les lignes qui suivent à un récapitulatif du travail effectué. Rappelons que notre travail consistait à propose une approche qui permet d'utiliser l'Analyse de Concepts Formels (ACF) pour le clustering des documents XML.

Nous nous sommes attelés, dans un premier temps, à définir les notions essentielles du langage de balisage extensible XML qui est un langage de présentation et de stockage. Ensuite, nous présentons les notions de fouille de donnés, et nous nous somme intéressés aux clustering et son applicabilité sur les documents XML.

Dans un second temps, nous avons procédé à un état de l'art sur l'Analyse de Concepts Formels pour le clustering des documents XML. Nous avons cité les notions de base de l'Analyse de Concepts Formels, ensuite nous avons parcouru les différents travaux basés sur l'ACF, et son applicabilité sur les documents XML .

Pour évaluer notre approche, nous l'avons développé avec *Java*. Nous avons évalué notre approche en utilisant deux collections différentes : DBLP en Anglais et Wikipédia en Français.

Néanmoins, nous estimons avoir atteint les objectifs que nous nous sommes fixés, en présentant une nouvelle approche de clustering basé sur l'analyse de concepts formels

Les perspectives envisageables à nos travaux portent sur plusieurs points :

- Améliorer le temps d'exécution de calcul des concepts.
- Améliorer le temps de génération des clusters.
- Trouver une manière pour réduire le nombre de concepts générés les plus similaires.
- Evaluer notre approche pour des collections de tests comportant un nombre plus élevé de documents.

Bibliographie

[1] World Wide Web Consortium, « Extensible Markup Language (XML) », http://www.w3c.org/xml/.

[2] Michard, A., « XML langage et applications », Editions Eyrolles, ISBN : 2-212-09206-7, 2001

[3] Olivier Carton « L'essentiel de XML » http://www.liafa.univ parisdiderot.fr/~carton/Enseignement/XML/Cours/Annexes/presentation.html,(12/06/2013).

[4] St-Laurent, S., « Introduction au XML», Editions Osman Eyrolles Multimedia, ISBN : 2-7464-0094-4, 2000.

[5] Popescu-Belis, A., « cours XML L'impact d'XML pour les techno multilingues »,Université deGenève http://www.issco.unige.ch/staff/andrei/xml, 2005.

[6] Brillant « XML Cours et exercices », Editions Eyrolles, ISBN : 978-2-212-12151-3, 2007

[7] Ludovic ROLAND « les base de XML » http://www.siteduzero.com/informatique/tutoriels/structurez-vos-donnees-avec-xml/les-entites-3,13.06.2013

[8] Harold, E.R., « XML Le guide de l'utilisateur », Editions Osman Eyrolles Multimedia, ISBN : 2-7464-0070-7, 2000

[9] Chagnon, G., «Cours de XML», dismonible en Internet en http://www.gchagnon.fr/cours/xml/index.html mars 2008

[10] Dong, G., Li, J., « Efficient mining of emerging patterns: Discovering trends and differences», In: ACM SIGKDD International Conference on Knowledge

[11] Dalamagas, T., Cheng, T., Winkel, K.J., Sellis, T.K., «Clustering XML Documents using Structural Summaries», In Proc. of ClustWeb - International Workshop on Clustering Information over the Web in conjunction with EDBT 04, Crete, Greece, 2004.

[12] Tufféry, S., «Data mining et Scoring», Editions Dunod, ISBN : 2 10 008184 5,2002

[13] Cabena, P. Hadjinian, P. Stadler, R. Verhees, J. et A. Zanasi , « Discovering Data Mining : From Concept to Implementation », Prentice Hall Upper Saddle River, NJ, 1998

[14] Larose, D.T., "Des données à la connaissance : Une introduction au data mining", Traduction et adaptation de Thierry Vallaud, Vuiber, ISBN 2-7117-4855-3, Septembre 2005

[15] Tufféry, S., "Datamining & statistique décisionnelle", http://data.mining.free.fr/cours/Presentation.pdf, 2006.

[16] Siegel (J. H.), Goldwyn (R. M.) et Friedman (H.P.). - Pattern and process of the evolution of human septic shock. Surgery, vol.70, pp. 232-245. 1971.

[17] Rabiner (L. R.), Levinson (S. E.), Rosenberg (A. E.) et Wilpon (J. G.). « Speaker independent recognition of isolated words using clustering techniques » . IEEE Trans. Accoust. Speech Signal Process., vol.27, pp. 336-349, 1979.

[18] Hodson (F. R.), Sneath (P. H. A.) et Doran (J. E.). « Some experiments in the numerical analysis of archaeological data ». Biometrika, vol.53, pp. 311-324, 1966.

[19] Kettenring (J. R.), Rogers (W. H.), Smith (M. E.) et Warner (J. L.). « Cluster analysis applied to the validation of course objectives». J. Educ. Statist., vol.1, , pp. 39-57. 1976.

[20] Guillaume (C). « Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information » . Thèse de doctorat Université d'Orléans VI, décembre 2004.

[21] Ng (T. R.) et Han (J.). « efficient and Effective Clustering Methods for Spatial Data Mining ». In : Proceedings of 20th International Conference on Very Large Data Bases VLDB'94, Fed. par Bocca (J. B.), Jarke (M.) et Zaniolo (C.). pp. 144-155. - Santiago de Chile, Chile, 1994

[22] Boley (D.). « Principal Direction Divisive Partitioning. Data Mining and Knowledge Discovery », vol.2, No 4, pp. 325-344, 1998.

[23] Candillier, L., « Contextualisation, visualisation et évaluation en apprentissage non supervisé », Thèse, Université Charles de Gaulle - Lille 3, 2006.

[24] Gennari (J. H.). « An experimental study of concept formation ». Technical Report N No 90-06, Irvine : University of California, Department of Information and Computer Science, 1990.

[25] Wille, R.. Restructuring lattice theory : «an approach based on hierarchies of concepts». Ordered sets, pages 445–470. 1982.

[26] Nizar, M., « Analyse de concepts formels guidée par des connaissances de domaine : Application à la découverte de ressources génomiques sur le Web», Thèse, l'université Henri Poincaré - Nancy 1, Mars 2009

[27] Seddoud, F., « Construction du treillis de concepts formels pour des contextes a intervalle de vérité à partir d'implication résiduées», Mémoire, Université de Boumerdès , 2012.

[28] N paskier : « Algorithme d'extraction et des règles d'association dans les base de donnes ».Université Clermont –Ferrand école Doctorale, science pour l'ingénierie pour Clermont –Ferrand, 2000.

[29] Guenoche and Mechelen, Guenoche, A. and Mechelen, I. V. Galois approach to the induction of concepts. In Mechelen, I. V., Hampton, J., Michalski, R., and Theuns, P., editors,«Categories and Concepts. Theoretical Views and Inductive Data Analysis », pages 287– 308. Academic Press, London, 1993.

[30] Yimin et al : « Web Text Clustering Based on Concept Lattice» is supported by the Fundamental Research Funds for the Central Universities(No.2009QN031), National Science Foundation of China(No.60972090) and Fundamental Research Funds for the Central Universities(No.2009QN030). 2010.

[31] Yun et al: «A New Search Results Clustering Algorithm based on Formal Concept Analysis» Fifth International Conference on Fuzzy Systems and Knowledge Discovery ,2008.

[32] Zhiming et al : « Formal Concept Analysis Support for Web Document Clustering Based on Social Tagging» International Conference on Uncertainty Reasoning and Knowledge Engineering, 2012

[33] Nyeint : « A combining approach of formal concept analysis and text mining for concept based document» Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05) 0-7695-2415-X/05 $20.00 © 2005 IEEE

[34] Madani Amina, D.Eddine Zegour « L'Analyse de Concepts Formels pour la fouille des documents semi-structurés»

[35] Jerom Bougeaut , « java la maitrise » ,Edition 2008

[36] Stefan Arnault :« Le XML en java SAX et DOM » ,Version n°1.1-28 juin 2005

[37] Wikipédia XML Corpus ,disponibe sur l'internet , http://www-connex.lip6.fr/~denoyer/wikipediaXML/,2007

[38] *DBLP Computer Science Bibliography,* dispinible sur internet DBLP XML records, http://dblp.uni trier.de/xml/ ,2011

[39] Bourret R : «XML Database Products ». California Disponible en Internet en http://www.rpbourret.com/xmldbms/index.htmLeermás: http://www.monografias.com/trabajos7/xml/xml.shtml.Citado el 1 de mayo del 2001.

[40] Xing, G., Xia, Z., « Classifying XML documents based on structure/content similarity», In: Workshop of the INitiative for the Evaluation of XML Retrieval, 2006.

[41] Vercoustre, A.M., Fegas, M., Lechevallier, Y., and Despeyroux, T., « Classification de documents XML à partir d'une représentation linéaire des arbres de ces documents », In Actes des 6ème journées Extraction et Gestion des Connaissances (EGC 2006), Revue des Nouvelles Technologies de l'Information (RNTI-E-6), pages 433–444, Lille, France, January 2006.

[42] Knijf, J.De., « Studies in Frequent Tree Mining », UU Universiteit Utrecht (169 pag.) (Utrecht: Utrecht University). Prom./coprom.: prof. dr. A.P.J.M. Siebes & dr. A.J. Feelders. 2008