Master 2 Recherche en Informatique
Responsable du master : Prof. Zoubir MAMMERI

Spécialité RIBD « Recherche d'Information et Base de données »
Responsable de la formation : Prof. Mohand BOUGHANEM

Équipe Systèmes d'Informations Généralisés
Composante : Documents, Données Semi-Structurées et usages

SPÉCIFICATION ET UTILISATION D'UN MODÈLE GÉNÉRIQUE DE DESCRIPTION D'ALGORITHMES D'INDEXATION MULTIMÉDIA

Dana CODREANU

Mémoire soutenu le 23 Juin 2010

Directeur de recherche	:	Prof. Florence SÈDES
Co-encadrant	:	Sébastien LABORIE

.

Remerciements

Je tiens à remercier le Professeur Claude Chrisment, responsable de l'équipe SIG, pour m'avoir accueilli dans son équipe.

J'exprime également mes profonds remerciements au Prof. Zoubir Mammeri, responsable du master Informatique et Télécommunication, ainsi qu'au Professeur Mohand Boughanem, responsable du parcours Recherche d'Information et Base de Données, qui m'ont permis d'accéder au master.

J'exprime ma vive reconnaissance à Madame le Professeur Florence Sèdes pour m'avoir encadré et pour m'avoir donné l'occasion de travailler sur ce sujet riche, actuel et passionnant. Sa grande disponibilité, son sens des relations, sa rigueur, son expérience, sa pédagogie et ses critiques constructives m'ont été précieux.

Je présente toute ma gratitude à Sébastien Laborie, pour avoir suivi et encadré cette étude et pour sa collaboration ainsi que son aide précieuse. Je voudrais lui exprimer ma reconnaissance pour les efforts inépuisables qu'il a fourni, pour sa patience et sa disponibilité à tout moment malgré ses occupations.

Je tiens également à remercier M. Romulus Grigoras pour m'avoir permis d'effectuer un stage d'été dans l'équipe SIG. Ce fût pour moi une première expérience de recherche au sein de l'IRIT.

Je remercie très sincèrement mes parents et ma sœur pour leur soutien et mes amis pour leur aide et encouragement.

Résumé

Actuellement, de nombreux contenus multimédias sont créés et stockés dans des environnements hétérogènes, éloignés géographiquement, de capacités diverses... Cette masse de données est généralement indexée par des algorithmes qui vont extraire des métadonnées permettant à un utilisateur de rechercher de l'information. De par la diversité et la multitude des algorithmes d'indexation existants, il n'est pas souhaitable d'exécuter tous les processus simultanément. En effet, cela consommerait énormément de ressources et certaines métadonnées pourrait n'être jamais exploitées par le système. De plus, il n'est pas envisageable de déterminer manuellement, avant et pendant leurs exécutions, les algorithmes d'indexation à installer. Dans ce mémoire, nous présentons un modèle de description d'algorithmes d'indexation et montrons comment exploiter ce modèle pour déterminer une liste d'algorithmes d'indexation appropriée par rapport à un ensemble de besoins, de propriétés et de contextes. Nous montrons comment nos travaux s'intègrent dans le cadre du projet ITEA2 LINDO pour indexer implicitement et explicitement des contenus multimédias.

Mots clés : Algorithme d'indexation, indexation distribuée

Abstract

Nowadays, many multimedia contents are created and stored on remote sites through heterogeneous environments having different capacities. This increasing amount of data is generally indexed by algorithms which extract metadata that will be further queried by a user for retrieving some desired information. Because of the large palette and the diversity of these indexing algorithms, it is not suitable to execute at once every indexing algorithms. Actually, it will overload the information system and will also produce metadata that may not be used inside the system. Moreover, it is cumbersome to determine manually, before and during their executions, the suitable set of indexing algorithms that have to be considered. We propose in this manuscript an indexing algorithm description model and show how to use such a model for determining, according specific user needs, properties and contexts, a relevant set of indexing algorithms. Our proposal has been integrated into ITEA2 LINDO project for indexing implicitly and explicitly multimedia contents.

Keywords : Indexing algorithm, distributed indexation

Sommaire

Chapitre 1

Introduction

1.1 Contexte et motivation

De nos jours, de nombreuses activités humaines génèrent des contenus multimédias. Ces contenus peuvent être stockés sur divers supports tels que le papier, les pistes magnétiques, les pellicules photographiques. L'évolution des techniques de stockage a conduit à la possibilité de numérisation des textes, images, sons et vidéos sous forme d'informations enregistrées et traitables par des ordinateurs. Dans le même temps, de multiples logiciels d'édition permettent la production des documents mono et multimédias encodés directement en format numérique.

Ces dernières années, la quantité de données multimédias produite et diffusée a connu une croissance très importante due à l'accès facile et peu cher aux technologies de plus en plus performantes (des PC, des dispositifs mobiles comme les téléphones portables, les assistants personnels, les ordinateurs portables) et aux connexions rapides à l'Internet.

Cette masse de données est généralement collectée en temps réel et stockée dans des serveurs hétérogènes, éloignés géographiquement, avec des capacités et des contextes différents. La question qui se pose est dès lors comment retrouver facilement des documents ou partie de ces documents ? Aider l'utilisateur à identifier, trouver, et filtrer les informations multimédias constitue un domaine de recherche aux enjeux importants. La grande quantité de données, l'hétérogénéité des sources, des formats, des logiciels de codage, des algorithmes d'indexation et techniques de gestion des métadonnées générées représentent les principaux aspects qui doivent être pris en

compte par un système d'information multimédias. Dans ce contexte, trois problèmes qui apparaissent dans ces systèmes sont adressés par les travaux de l'équipe :

- **L'architecture** : la plupart des systèmes actuels proposent des architectures spécifiques aux cas d'utilisation ;
- **Les techniques de gestion du processus et des algorithmes d'indexation** : généralement, le système d'information multimédias dispose d'un ensemble d'algorithmes fixé a priori par les développeurs ;
- **Les modèles de métadonnées** : sont utilisés pour organiser et retrouver des contenus multimédias. Il existe un grand nombre de modèles qui ne sont toujours interopérables.

Dans ce cadre, notre étude porte sur le deuxième aspect, les techniques de gestion du processus d'indexation dans un système d'information multimédias.

1.2 Problématique

Beaucoup de défis doivent être relevés pour la mise en place de mécanismes d'indexation efficaces. En effet, d'un côté nous avons de grands volumes de contenus multimédias hétérogènes distribués sur plusieurs serveurs ayant des capacités différentes (hardware, software, réseau, chargement, autonomie, etc.). D'un autre côté, nous avons une grande diversité d'algorithmes d'indexation, diversité qui concerne : les entrées et les sorties, les contraintes d'exécution (hardware, software, concernant le contenu des médias analysés, etc.), les performances, etc.

Il n'est pas souhaitable d'installer tous les algorithmes d'indexation existant sur chaque serveur contenant des contenus multimédias à analyser, ceci pour deux raisons principales :

- limitation de ressources et de contexte ;
- certaines sorties des algorithmes peuvent ne pas être exploitées dans certains contextes.

Une question se pose alors : comment trouver la liste des algorithmes la plus appropriée pour répondre aux besoins des utilisateurs face à des contextes donnés ?

Les problématiques qui vont être adressées dans ce mémoire peuvent être également exprimées par les questions suivantes :

- Comment peut-on décrire et représenter de manière générique les algorithmes d'indexation ?

– En se basant sur le modèle de description et en tenant compte d'une requête et d'une représentation du contexte comment peut-on déterminer les algorithmes les plus appropriés pour répondre à la requête ?
– Quelles sont les chaines d'application d'algorithmes qui vont permettre d'extraire les éléments désirés ?

Nous répondons à chacune de ces trois questions dans les chapitres suivants après avoir proposé un plan du mémoire dans la section suivante.

1.3 Plan

Le manuscrit est organisé en trois parties :

– Un **état de l'art** dans le domaine de l'indexation de contenus multimédias selon deux axes, les algorithmes d'indexation (§2.2) et systèmes d'information multimédias gérant plusieurs algorithmes d'indexation (§2.3). Dans la dernière section du premier chapitre, nous présentons nos deux observations principales concernant l'état de l'art : la manque d'un modèle de description des algorithmes d'indexation et d'une procédure de sélection des algorithmes d'indexation selon différents critères. Enfin, nous définissons les objectifs qui constituent aussi notre contribution par rapport à ces observations.
– **Contribution**. Notre première contribution est présentée dans le chapitre 3, dans lequel nous définissons un modèle générique de description d'algorithmes d'indexation. Nous détaillons les concepts et outils que nous avons utilisés dans notre démarche et les avantages apportés par notre approche. La deuxième contribution théorique est proposée dans le chapitre 4 à propos du processus de recherche d'algorithmes d'indexation.
– **Expérimentation**. Le prototype que nous avons développé est présenté dans le chapitre 5. Nous détaillons les trois parties principales du prototype : Création et stockage de modèles d'algorithmes (§5.3.1), Déclaration des chaines(§5.3.2) et Querry Applet (§5.3.3). Nous montrons que notre travail s'intègre dans le cadre du projet européen LINDO.
  Nous concluons par le bilan et les perspectives de notre travail.

Chapitre 2

État de l'art

2.1 Introduction

Les progrès technologiques ont engendré la création d'énormes archives qui contiennent des documents mono et multimédias ayant pour conséquence la complexité de l'accès et de l'utilisation de ces données.

Généralement, un système d'information qui gère et qui offre la possibilité de rechercher des contenus multimédias est composé de [7, 29] :

- une collection multimédias qui contient plusieurs contenus multimédias ;
- une collection de métadonnées qui contient des informations sur les caractéristiques du média (taille, nom, type de fichier) et sur son contenu. Ces métadonnées peuvent être codées dans plusieurs standards : EXIF [46], Dublin Core [22], MXF [11] ;
- un moteur d'indexation qui contient plusieurs algorithmes d'indexation qui seront exécutés sur la collection multimédias afin d'enrichir la collection de métadonnées ;
- un module de restitution et de visualisation des résultats de la recherche ;

Les principales fonctionnalités d'un tel système sont, par exemple [10] :

1. la création et la gestion de grandes bases de données contenant des documents de taille variable ;
2. la recherche des documents ou extraits de documents correspondant à un critère de recherche donné ;

3. la création dynamique et automatique d'un document « résumé » des données recherchées ;
4. l'assurance d'un niveau de qualité et de performance acceptable pour l'utilisateur.

Des travaux concernant les trois premières fonctionnalités ont été réalisés dans notre équipe, tels que [1] qui a proposé un modèle générique de description de documents multimédias et [8] qui traite de l'annotation et de l'analyse de média.

Dans ce qui suit, nous allons présenter les travaux de recherche effectués dans le domaine de l'indexation de contenus multimédias en suivant deux axes abordées dans la suite du mémoire, les algorithmes d'indexation (§2.2) et les systèmes d'information multimédias gérant plusieurs algorithmes d'indexation (§2.3).

2.2 Algorithmes d'indexation

Une définition du processus d'indexation multimédias a été spécifiée dans [10] : "*c'est la modélisation et la représentation de l'information contenue dans les documents sous la forme d'une liste ordonnée de données significatives, appelée* ***index****, par un système de gestion des contenus multimédias*". Ceci permet de réaliser les fonctionnalités 2 et 3 de la classification présentées dans la section 2.1 (la recherche des documents et la création d'un document qui résume les données recherchées).

De plus, [10] montre que le travail de recherche et d'extraction des caractéristiques, de la structure, et de la sémantique des contenus multimédias constitue le travail d'**indexation de données**. Le résultat de l'indexation d'un document est une description numérique (exploitable par une machine) dans un ou plusieurs formalismes qui permet l'accès, la recherche, le résumé, la classification et la réutilisation de tout ou partie du document.

Nous pouvons résumer les deux descriptions des systèmes d'information multimédias présentées dans l'introduction (concernant les fonctionnalités et les composantes d'un tel système) et la définition donnée par [10] dans un schéma qui définit un **algorithme d'indexation** (Figure 2.2).

Un algorithme d'indexation prend en entrée des contenus multimédias, il les analyse, il en extrait des caractéristiques (e.g., l'histogramme de couleurs, les statistiques du signal, la fréquence des mots dans un texte, etc.) et il produit des informations qui

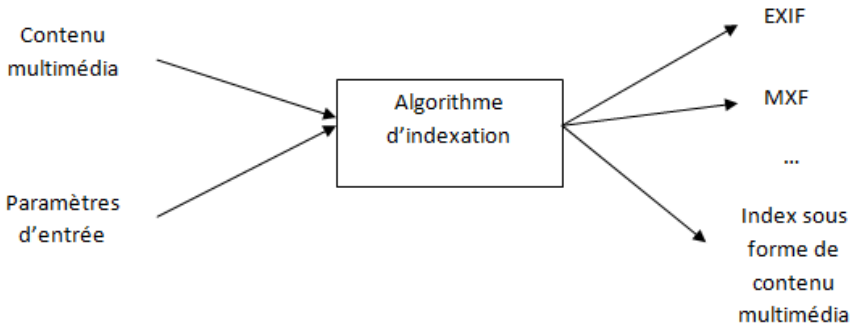


FIGURE 2.1 – Algorithme d'indexation

décrivent ces contenus. Ces informations pourront être structurées sous forme de métadonnées ou d'autres documents mono-média ou multimédias.

Les algorithmes d'indexation sont très divers et très hétérogènes. Nous allons illustrer leur diversité en présentant une partie des travaux de recherche effectués dans ce domaine en suivant une classification selon le type de média à l'entrée : **indexation monomédia** (texte, audio, image, video) (§2.2.1) et **indexation "cross-media"** (§2.2.2).

2.2.1 Indexation monomédia

Généralement, dans l'analyse de contenus multimédias, la majorité des algorithmes traite un seul media. [10] argumente cette affirmation en disant que les logiciels de recherche sur le Web n'utilisent pas l'image, notamment il indique que les systèmes de recherche d'images mélangent à peine les descriptions visuelles et textuelles. Ceci est egalement le cas pour l'analyse de vidéo qui est habituellement faite séparément pour le son et l'image [26]. Une des raisons de cela est que ces médias concernent des champs scientifiques différents et parfois très séparés. Une autre raison peut être constituée par le fait que deux ou trois algorithmes qui sont spécialisés sur le traitement d'un média peuvent être plus rapides qu'un super algorithme qui indexe tout type de contenu. La question qui s'impose est comment intégrer toutes ces informations (issues des traitements faits sur chaque média d'un document) afin de rendre l'indexation plus efficace ?

Les descripteurs multimédias

Tout au long de ce mémoire le terme **descripteur** ou **caractéristique** multimédia est utilisé en se référant au contenu de l'index i.e., le résultat du processus d'indexation. [10] les descripteurs comme des "*données qui décrivent elles-mêmes le contenu des données*".

Les **descripteurs** peuvent être classifiés selon leur origine [10] :

- descripteurs **culturels** (généraux) qui consistent en des données concernant le contexte dans lequel le document a été crée, (e.g., l'auteur, la date de création etc.) ;
- descripteurs **du contenu** qui sont produits après le traitement des données.

Les **descripteurs de contenu** sont généralement classifiés selon leur niveau sémantique et leur granularité [10, 12].

Selon le **niveau sémantique** on peut disposer de :

- **descripteurs de bas niveau** qui sont extraits automatiquement à partir du document par l'application des algorithmes sans tenir compte de la sémantique du contenu (e.g., caractéristiques du spectre d'un signal audio, ou de l'histogramme des couleurs d'une image ou des fréquences des mots dans un texte) ;
- **descripteurs de moyen niveau** qui sont représentés par des concepts qui expriment des propriétés ayant une signification plus importante pour l'utilisateur (e.g., la présence de l'herbe ou de la montagne ou du ciel dans une photo ou vidéo, le nombre d'instruments dans un morceau musical) ;
- **descripteurs de haut niveau** qui sont des concepts qui expriment des propriétés très significatives et qui peuvent être exprimés en termes d'autres concepts (e.g., le temps, le lieu, les relations entre objets, l'action d'une vidéo, le thème d'un texte).

Selon le **niveau de granularité** les descripteurs ont la classification suivante :

- **descripteurs de niveau local** : ils sont calculés pour une portion de petite taille de signal ;
- **descripteurs de niveau intermédiaire** : ils sont calculés pour une portion plus grande du signal, généralement de dimension variable ;
- **descripteurs de niveau global** : ils sont calculés sur l'intégralité du média, comme par exemple la couleur moyenne d'une photo.

Texte

Pour les documents textuels, quelques approches d'indexation se sont inspirées de la Recherche d'Information classique [40] ou de la Recherche de l'Information sur le Web, en exploitant les caractéristiques hypertexte, comme les liens entre les pages [5] et les tags HTML [18]. Beaucoup de travaux de recherche ont été réalisés afin d'ajouter une couche sémantique à l'indexation textuelle et passer d'une indexation basée sur les termes à une indexation basée sur les concepts grâce aux techniques d'indexation sémantique [41] ou aux modèles et méthodes de représentation des connaissances qui sont spécifiques au domaine de l'intelligence artificielle (réseaux neuronaux, réseaux sémantiques, réseaux bayesiens) [38]. Des techniques comme la désambigüisation du sens des mots [34] (par l'étiquetage des mots selon l'appartenance aux catégories lexicales ou syntaxiques) en utilisant des ressources disponibles sur le Web (comme les dictionnaires, les thésaurus ou les ontologies, e.g. la ressource lexicale la plus utilisée est WordNet [1]), l'extraction de concepts selon une ontologie [25] ou la classification des parties du texte selon les thèmes detectés [20] représentent les sujets de beaucoup de travaux récents. Dans le cadre de l'équipe il existe aussi des travaux concernant l'extraction de l'information à partir des textes [28].

Audio

Pour l'indexation du contenu audio le workflow principal est présenté dans le schéma de la Figure 2.2.

Généralement, dans un système qui indexe du contenu audio, dans un premier temps il y a une étape d'extraction des caractéristiques de bas niveau sur lesquelles vont se baser les autres algorithmes d'indexation. Après la segmentation, il est possible d'exécuter des algorithmes qui caractérisent le type de média (e.g., refrains publicitaires), effectuent une reconnaissance des locuteurs ou utilisent la transcription du discours.

Selon [10], les **descripteurs audio de bas niveau** sont classifiés en fonction du traitement appliqué pour les extraire.

Au niveau local (calculés sur un segment d'un échantillon à quelques dixièmes de seconde) :

1. `http://wordnet.princeton.edu/`

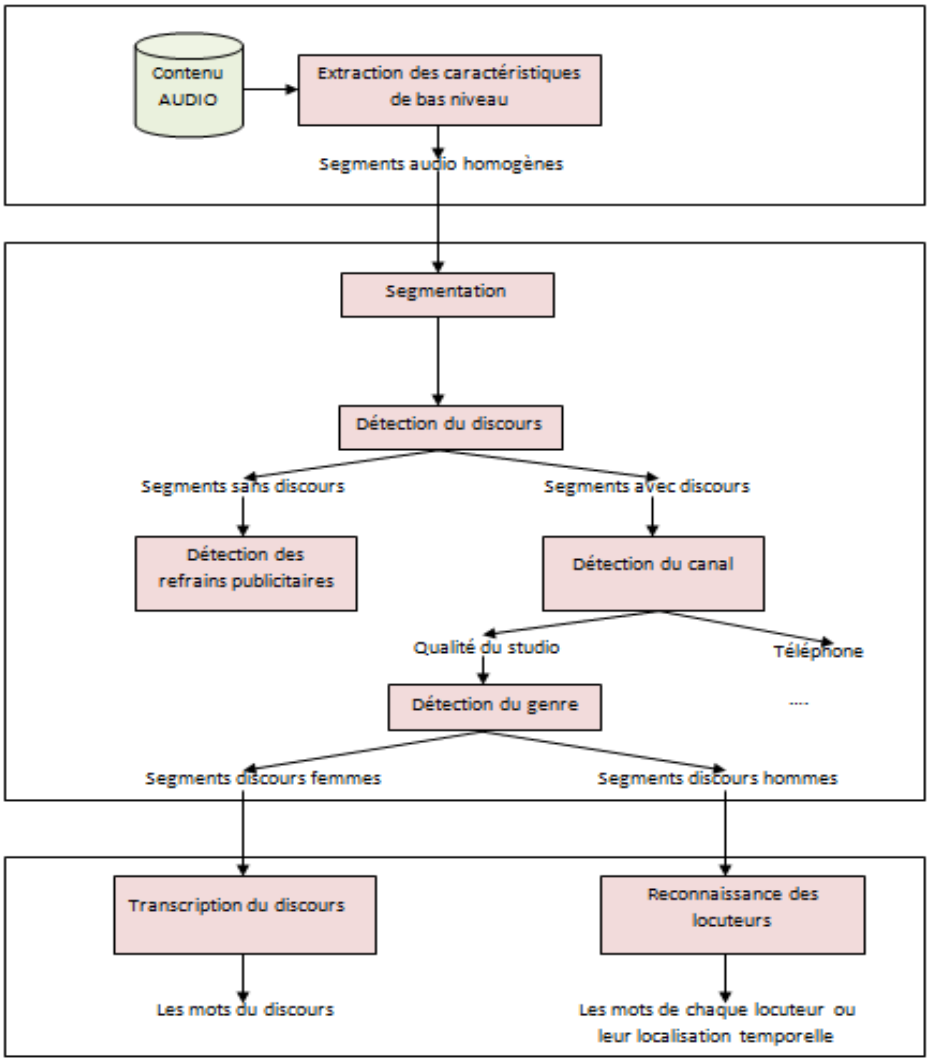


FIGURE 2.2 – Workflow principal de l'indexation audio

- Descripteurs temporels. Ils sont extraits à partir de la forme d'onde ou de la courbe d'énergie (enveloppe) : e.g., taux de silence [43], corrélation croisée, énergie à court terme [53], volume [31], écart dynamique du volume [31], modulation à 4Hz [31, 43], pulsation [43], taux de réverbération [9] ;
- Descripteurs d'énergie. Ils décrivent l'énergie de différents types de contenus : énergie globale, énergie harmonique, énergie du bruit, énergie de bandes [42] ;
- Descripteurs spectraux. Ils sont calculés à partir de représentation 2D du signal (fréquence, temps), transformée de Fourier à court terme (STFT) ou transformée en ondelettes (DWT) [45], centroïde spectral [43] , flux spectral [43], point de "roll-off" [43], bande passante [31], statistiques de bandes et ratios [47], périodicité de bande [32], inharmonicité [35], rythme et tempo [42] ;
- Descripteurs harmoniques. Ils sont calculés à partir d'un modèle harmonique du signal : Fréquence fondamentale [42, 35] , Vibrato [35], Slope [35] ;
- Descripteurs perceptifs. Ils sont déterminés à partir d'un modèle d'écoute humaine : Loudness, Sharpness [42], "Mel-Frequency Cepstral Coefficients" ou MFCC [47]. Les MFCC se présentent sous la forme d'une représentation 2D (fréquence, temps). L'échelle des fréquences est fixée afin de correspondre à la capacité cognitive de l'oreille humaine.

Au **niveau intermédiaire et global**, on peut considérer que les descripteurs sont un " résumé" de la séquence des descripteurs locaux extraits sur le segment considéré du document.

[31] propose un algorithme qui extrait et se sert des descripteurs de bas niveau pour réaliser la classification des scènes des contenus audio des émissions TV dans les catégories : publicité, basketball, football, journaux, météo. Ils utilisent un classificateur basé sur des réseaux neuronaux. Dans [20], le système d'indexation de contenus multimédias contient un algorithme de segmentation acoustique. Les segments résultant sont analysés par un algorithme qui fait de la reconnaissance automatique de la parole et sépare les segments qui contiennent des paroles des les segments contenant du silence. La transcription des paroles est fournie à un algorithme d'indexation textuelle qui fait de la classification des segments de texte selon des concepts. Un système très utilisé qui réalise une transcription du discours sous forme textuelle est le système TRANSCRIBER [2].

2. `http://trans.sourceforge.net/en/presentation.php`

Image et Vidéo

Les travaux de recherche dans le domaine de l'indexation du contenu visuel sont très nombreux. Généralement, les vidéos sont considérées comme une superposition d'une suite d'image et d'un contenu audio (l'analyse de l'audio a été présentée dans la section précédente). Les systèmes utilisent des démultiplexeurs qui séparent le contenu audio du contenu visuel et traitent les contenus séparément, e.g., [19]. Nous nous focalisons dans cette partie sur le contenu visuel.

[49, 27] ont élaboré différentes synthèses des principales techniques utilisées en analyse d'images. Une liste de travaux de recherche concernant l'indexation d'images est également disponible à cette adresse : `http://www.visionbib.com/bibliography/applicat804.html`. Dans la suite, nous présentons certaines d'entre elles pour illustrer leur diversité en terme de caractéristiques extraites, de paramètres utilisés, d'efficacité d'exécution...

Il est important de mentionner que les caractéristiques des niveaux sémantiques supérieurs sont obtenues à partir des caractéristiques de plus bas niveau et pas directement du document.

Les caractéristiques de bas niveau peuvent caractériser toute l'image (caractéristiques globales) ou des régions de l'image (caractéristiques locales).

Parmi les travaux concernant l'extraction des **caractéristiques de l'image de bas niveau globales** de nombreux systèmes utilisent de tels algorithmes : QBIC [15] propose des extracteurs et un système de recherche d'images basés sur la couleur, la texture et la forme, et PicToSeek [17] envisage l'utilisation de la combinaison des invariants des couleurs et des formes afin d'indexer et de rechercher des images. [21] propose la définition d'une nouvelle caractéristique de l'image, "the colour correlogram", qui en plus de l'histogramme, prend en considération la distribution spatiale des couleurs. Les caractéristiques visuelles incluses dans le standard MPEG7 [36] sont parmi les plus utilisés. Le standard permet la mutualisation des différentes descriptions telles que les caractéristiques de bas niveau (e.g., forme, taille, texture, couleur, mouvement, position...), des informations du niveau sémantique local (e.g., personnage, lieu, action) et des descripteurs généraux (e.g., auteur, date de création, format). Il est également possible, en utilisant ce standard, de décrire les relations de structure temporelles et spatiales entre les objets qui composent l'image ou la vidéo. Il est formalisé en utilisant le langage XML [3].

3. `http://www.w3.org/XML/`

Les caractéristiques de bas niveau locales peuvent offrir une représentation plus précise de l'image, car les objets petits peuvent être décrits séparément donc ils peuvent être retrouvés et détectés plus facilement. Par conséquent, beaucoup de techniques récentes, spécialement celles qui font de la reconnaissance des classes d'objets (e.g., personne, véhicule, animal, etc.) sont basées sur l'analyse des caractéristiques locales [14, 39]. L'inconvénient par rapport aux techniques utilisant des caractéristiques globales est la grande quantité de données produites.

Les algorithmes qui extraient des caractéristiques locales en général suivent les étapes suivantes :

- la détection des points d'intérêts et des régions de support (dont les voisinages font une différence dans le cadre de l'image, e.g., détection des changements spatiaux) [2] ;
- l'extraction des caractéristiques locales, dont le but principal est de décrire les régions de l'image en utilisant des informations géométriques ou photométriques robustes.

Comme exemples illustrant l'extraction des caractéristiques locales, ils existent des algorithmes qui utilisent l'histogramme de couleurs (e.g., reconnaissance des objets 3D [24], description des formes [51]), des algorithmes qui sont basés sur le modèle spatial de la fréquence du signal de l'image (e.g., classification des textures [33]), des algorithmes qui calculent les dérivés des variations de l'image dans le voisinage des points d'intérêt, variations qui correspondent à la luminosité, à la couleur ou aux changements des structures locales comme les formes ou les contours, etc. [37] propose un algorithme qui se base sur les coefficients DCT (Transformée en cosinus discrète) et sur les vecteurs de mouvement pour faire de la détection des changements des plans dans une vidéo compressée à l'aide du standard MPEG [4] .

Ils existent aussi des travaux concernant la **comparaison des algorithmes qui utilisent des descripteurs de bas niveau**. [4] compare plusieurs algorithmes qui détectent et classifient les frontières des plans et les types de transition des plans d'une vidéo en prenant en compte la facilité d'implémentation, la performance attendue et la présence des caractéristiques intéressantes. Dans [4], 5 algorithmes ont été testés. Ils utilisent : le premier, un histogramme pour chaque cadre de la vidéo et un seuil ; le deuxième, un histogramme pour chaque région du cadre de la vidéo et deux seuils ; le troisième, un histogramme pour chaque cadre et deux seuils ; le quatrième, la différence entre les pixels des cadres et trois seuils ; le cinquième, la différence entre les coefficients DCT (Discrete Cosines Transform) et un seuil. Tous les

4. `http://www.mpeg.org/`

algorithmes ont été implémentés dans le langage C et ils ont été exécutés sur la même plateforme en ayant les mêmes données d'apprentissage et de test. L'évaluation de la performance a été faite en termes de rappel et de précision. [16] évalue aussi plusieurs algorithmes qui font la segmentation en plans d'une vidéo. En plus de l'évaluation de la performance basée sur le nombre de détections correctes non signalées et fausses alarmes [16] teste aussi la variation de la performance en fonction de l'encodeur.

Les limitations de l'usage des caractéristiques de bas niveau résident en le fait que les systèmes basés sur les descripteurs de bas niveau ne satisfont pas les besoins des utilisateurs qui sont intéressés par le contenu sémantique. C'est ce que les spécialistes appellent le " fossé sémantique".

Toutes les techniques présentées précédemment sont utilisées par des algorithmes qui font de la détection et de la reconnaissance des objets et des visages pour ajouter une couche sémantique aux descripteurs des images.

Pour la première catégorie d'algorithmes, deux modèles principaux sont utilisés. Le modèle du sac de mots considère l'image comme une collection de régions en ignorant leur structure spatiale. Même si cette méthode est traditionnellement utilisée pour la classification d'images, certaines approches traitent le problème de la reconnaissance d'objets. [30] et [48] utilisent la reconnaissance des textures. [50] modélise les images comme des distributions sur un dictionnaire visuel. Le modèle basé sur les parties représente la plus populaire des techniques utilisées pour la reconnaissance d'objets qui vise à décrire des relations rigoureuses entre les différentes parties de l'image. Par contre, les méthodes qui utilisent ce modèle sont beaucoup plus complexes e.g., [13, 52].

Comme les travaux de recherche dans ce domaine sont de plus en plus nombreux et par conséquence aussi les méthodes proposées, des campagnes d'évaluation des algorithmes de reconnaissance d'objets ont été mises en place. Deux exemples qui évaluent les performances des algorithmes (en termes de précision moyenne, donc le taux des reconnaissances valides) sont : PASCAL VOC 2006 [5] et IMAGEVAL Task 4 2006 [6].

Dans le cas de la catégorie d'algorithmes qui font de la détection des visages, les méthodes utilisées sont différentes pour les images statiques (utilisent les caractéristiques, la texture et la couleur de la peau et des modèles basés sur les espaces vectoriels et les réseaux neuronaux) et pour les séquences d'images (utilisent le

5. `http://www.pascal-network.org/challenges/VOC/`
6. `http://www.imageval.org/e_presentation.html`

flux optique). Les systèmes qui réalisent de la reconnaissance des visages sont nombreux : A4Vision [7], Cognitec [8], FaceSnapRecorder [9], FaceITArgus [10], VeriLookFace [11], OpenCV [12] .

2.2.2 L'analyse "cross-media" des contenus multimédias

Une amélioration de l'indexation (en termes d'efficacité) suppose une approche cross-media, dans laquelle la collaboration de analyses des différents médias est utilisée. Selon [10], les applications mono-média qui reposent sur un médium unique pour acquérir l'information des documents sont sujettes à des erreurs ; quelle que soit la qualité de l'information d'un signal, il fournit au système une vue unique de ce qui se passe. Afin d'offrir aux applications un traitement robuste, il est nécessaire de se baser sur plusieurs médias pour réunir l'ensemble des informations correctes [26]. Il est possible d'utiliser la redondance des médias pour valider les données qu'ils fournissent. La complémentarité des médias est aussi utilisée pour résoudre les ambiguïtés ou réduire les erreurs quand une perturbation environnementale affecte le système. De nombreux domaines d'analyse ont été particulièrement étudiés tels que la transcription de la parole, l'identification de personnes et la détection d'unités logiques, de dialogues et de concepts sémantique.

2.2.3 Chaînage des algorithmes

Un scénario de chaînage des algorithmes est présenté dans [19] : en partant d'un document audio-visuel, en première phase, on sépare le contenu vidéo du contenu audio. Les deux flux générés sont traités séparément. Le flux audio est transmis à un algorithme de segmentation et il est décomposé en segments de Parole/Musique/-Bruit. Un outil d'identification de langues sépare les paroles en anglais et français. Les segments résultant peuvent constituer l'entrée d'un algorithme de transcription du discours ou de détection des thèmes. Pour la partie vidéo une extraction des caractéristiques de bas niveau suivie d'une extraction des images clés est effectuée. Un

7. `http://www.a4vision.com/`
8. `http://www.cognitec-systems.de/`
9. `http://www.facesnap.de/htd/fsnapr.html`
10. `http://www.l1id.com/`
11. `http://www.neurotechnologija.com/`
12. `http://sourceforge.net/projects/opencvlibrary/`

algorithme qui extrait le texte des vidéos et un autre qui réalise la reconnaissance des visages sont exécutés en parallèle.

Pour conclure, il existe une grande diversité d'algorithmes d'indexation, diversité qui vise en premier lieu les entrées et les sorties et qui est la conséquence de la richesse du contexte, des paramètres et des performances des algorithmes. Mais la plus importante observation que nous avons constatée est qu'il n'existe pas un modèle général qui décrit des algorithmes d'indexation.

2.3 Systèmes d'information multimédias gérant plusieurs algorithmes d'indexation

La dernière décennie est riche en projets qui proposent des architectures pair à pair ou centralisées et des plateformes d'indexation qui gèrent plusieurs algorithmes d'indexation.

La grande majorité de systèmes proposent une **architecture distribuée avec un contrôle centralisé** :

Le projet K-Space [13] vise l'aire de recherche de "l'Interaction Sémantique avec les Multimédias" et est concentré sur l'inférence sémantique pour l'annotation semi-automatique et la recherche de contenus multimédias. Il intègre trois sous domaines de recherche : l'analyse multimédia basée sur le contenu (développement des outils et algorithmes pour le traitement de bas niveau du signal, segmentation des objets, traitement audio, analyse du texte, structuration et description du contenu audiovisuel), l'extraction des connaissances (construction de l'infrastructure d'une ontologie qui vise l'acquisition des connaissances des contenus multimédias) et la représentation et la gestion sémantique des données multimédias.

Le projet VITALAS [14] (Video and image Indexing and reTrievAL in the Large Scale) a comme principal objectif l'indexation et la recherche cross-media, c'est-à-dire il prend en compte une grande quantité de documents multimédias et il envisage la production des caractéristiques mono-média pour développer des techniques intelligentes d'accès à des archives multimédias professionnelles. La solution proposée est basée sur un nombre fixé d'algorithmes et ne supporte pas l'intégration distante de nouveaux algorithmes d'indexation.

13. `http://kspace.qmul.net:8080/kspace/`
14. `http://vitalas.ercim.org/`

Le projet CANDELA [15] réalise l'analyse du contenu vidéo, la distribution dans le réseau et les fonctionnalités de stockage. La gestion des outils d'indexation utilisés afin d'obtenir les métadonnées ne représente pas le but principal du projet. En effet, le projet se concentre sur la gestion des contenus multimédias.

Le réseau d'excellence MUSCLE [16] (Multimedia Understanding through Semantics, Computation and LEarning) a développé plusieurs types d'algorithmes d'indexation (reconnaissance d'objets, analyse du contenu, détection du comportement inhabituel, détection des personnes, reconnaissances de la parole). Ces algorithmes sont intégrés et gérés au niveau du serveur central, dans le cadre d'une architecture distribuée.

Le projet WebLab [17] adopte aussi une architecture centralisée qui offre la possibilité de charger de nouveaux algorithmes sur la plateforme. Les algorithmes d'indexation sont considérés comme des services Web. L'installation, le déploiement et l'exécution des algorithmes disponibles sont réalisés manuellement, par l'intermède d'une interface qui offre aussi la possibilité de définir des règles de chaînage des algorithmes.

[20] propose un système d'indexation automatique de contenus multimédias qui inclut la segmentation audio, la reconnaissance automatique de la parole, la segmentation des sujets et des caractéristiques d'indexation vidéo. Le système envisage l'indexation des journaux multimédias et des structures extraites des journaux basées sur l'information concernant le son, la parole et l'image.

Video Scout [23] est un système qui segmente et indexe des programmes TV selon leur information audio, vidéo et textuelle. L'architecture proposée est structurée en trois modules : Prétraitement video, Segmentation et indexation et Stockage et Interface avec l'utilisateur. L'indexation cross-media est réalisée par l'intermédiaire d'un réseau bayésien qui intègre les informations obtenues par l'extraction des caractéristiques de bas niveau de chaque composante de la vidéo à l'entrée (visuelle, audio et textuelle) en générant de l'information sémantique (des concepts) concernant les thématiques des programmes TV. Le nombre d'algorithmes d'indexation est figé et leur gestion est faite manuellement.

Aucune des solutions présentées ne fait la différence entre les processus d'indexation implicite (avant l'exécution du système) et d'indexation explicite (à la volée). Elles considèrent un nombre fixé a priori d'algorithmes qui seront exécutés et ne proposent aucune stratégie de sélection d'algorithmes par rapport à la requête de l'utilisateur

15. `http://www.hitech-projects.com/europrojects/candela`
16. `http://www.muscle-noe.org`
17. `http://weblab-project.org/`

ou au contexte d'exécution de l'algorithme.

[19] présente une implémentation expérimentale d'une plateforme distribuée qui permet le partage des ressources d'indexation, l'intégration sur la plateforme des algorithmes qui peuvent se trouver sur différents sites. Le chaînage automatique des algorithmes d'indexation est réalisé par une procédure qui à partir d'un index donné (recherché) identifie tous les scénarios de chaînage des algorithmes d'indexation en se basant sur l'appariement entre les entrées et les sorties des algorithmes. Afin de réaliser les deux objectifs, l'auteur propose un modèle de spécification d'un algorithme d'indexation qui contient des informations générales (URL, courte description, fournisseur) et sur l'entrée et la sortie (le nombre de paramètres et le type de données). Ce travail constitue un premier pas vers un système d'information multimédias effectuant une gestion automatique (semi-automatique) de plusieurs algorithmes d'indexation. Cependant, le modèle de spécification proposé est simple. Il est utilisé par la plateforme pour la constitution de l'invocation et du lancement de l'algorithme, et par la procédure de chainage pour comparer les entrées et les sorties. Le modèle ne prend pas en compte le besoin de recherche d'un algorithme selon la description de sa sortie, selon les préférences de l'utilisateur ou selon le contexte du système. De plus l'utilisateur doit être familiarisé avec le domaine de l'indexation multimédias, car il doit choisir l'algorithme d'indexation à appliquer ou donner un index qui représente le type des données à la sortie d'un algorithme pour lancer la méthode de construction d'un chaînage.

Comme exemple **d'architecture pair à pair** nous mentionnons le projet SAPIR [18] qui traite de la recherche à grande échelle dans des contenus audio-visuels. Dans l'architecture proposée, des caractéristiques sont extraites sur un pair et elles sont envoyées vers un autre pair pour effectuer l'indexation. Le moteur de recherche utilise trois moteurs d'indexation différents, un pour images fixes, un pour du texte et un pour des contenus audio et vidéo.

Comme synthèse nous illustrons dans la table 2.1 les principales caractéristiques des systèmes d'informations multimédias qui concernent notre étude §2.1. Celles-ci sont : L'objectif principal du projet, s'il est ciblé sur un domaine, l'architecture (elle peut être centralisée, pair à pair ou orientée services web), le type de processus d'indexation utilisé (monomedia et cross media i.e.) et la flexibilité du nombre d'algorithmes.

18. `http://www.sapir.eu`

TABLE 2.1 – Comparaison des systèmes d'information multimédias.

Système	Objectif principal	Domaine ciblé	Architecture			Indexation		Nombre d'algos	
			Centr.	P2P	SOA	Mono	Cross	Fixé	Variable
[20]	indexation automatique du contenu MM	+	+	−	−	+	−	+	−
Weblab	plateforme pour l'integration des composantes qui font le traitement du contenu MM	−	+	−	+	−	+	−	+
Vitalas	indexation et recherche crossmédia	−	+	−	+	−	+	−	+
K-Space	l'inférence sémantique pour l'annotation semi-automatique et la recherche des contenus MM	−	+	−	+	+	−	−	+
Candela	analyse du contenu vidéo distribution dans le réseau fonctionnalités de stockage	+	+	−	−	+	−	−	+
Muscle	utilisation de l'apprentissage statistique pour la générationu (semi) automatique des métadonnées sémantiques	−	+	−	−	+	−	−	+
[19]	intégration sur une plateforme des algorithmes d'indexation et chainage automatique	−	+	−	+	+	−	−	+
Video Scout	segmentation et indexation des programmes TV	+	+	−	−	−	+	+	−
Lindo	indexation distribuée à grande echelle	−	+	−	−	+	−	−	+
Sapir	recherche à grande échelle dans du contenu audio-visuel	−	−	+	−	+	−	+	−

Conclusion et objectifs

Dans ce chapitre, nous avons présenté un état de l'art dans le domaine des systèmes de gestion et d'indexation de contenus multimédias. En guise de synthèse, nous allons tout d'abord reprendre les principales observations faites dans les sections précédentes concernant les deux aspects considérés, les algorithmes d'indexation et les systèmes d'information multimédias qui gèrent plusieurs algorithmes d'indexation.

– il y a une grande diversité d'algorithmes, en ce qui concerne leurs sorties, leurs entrées, leur contexte d'exécution, leurs paramètres et leurs performances. De plus, il n'existe pas un modèle de description générique des algorithmes ;

– même s'il existe beaucoup de systèmes qui gèrent plusieurs algorithmes d'indexation, aucun d'entre eux ne prend en compte la différence entre les processus d'indexation implicite et explicite, et ne définit de procédure de recherche des algorithmes appropriés par rapport à une requête d'utilisateur et à un contexte d'utilisation.

Les objectifs que nous nous proposons d'aborder dans les chapitres suivants et qui seront notre contribution sont :

– la définition d'un modèle générique de description d'algorithmes d'indexation, qui peut constituer un guide pour les développeurs et qui permet de retrouver des algorithmes en fonction de plusieurs critères ;
– l'élaboration d'une méthode de recherche d'algorithmes d'indexation tenant en compte la requête et les préférences de l'utilisateur ainsi que le contexte d'exécution des algorithmes.

Chapitre 3

Modèle générique de description d'algorithmes d'indexation de contenus multimédias

3.1 Introduction

Dans le domaine de l'indexation de contenus multimédias, de nombreuses équipes développent leurs propres algorithmes d'indexation. Pour diffuser et partager leurs algorithmes, ces équipes collaborent dans le cadre de projets, tels que CANDELA, LINDO, WebLab, ou plus généralement utilisent le web. Par exemple, les développeurs d'OpenCV diffusent des algorithmes d'indexation de contenus multimédias via le lien suivant : $http://sourceforge.net/projects/opencvlibrary/$.

Malheureusement, ce mode de diffusion a l'inconvénient de ne pas regrouper au sein d'un même modèle les descriptions de ces algorithmes. Par conséquent, cela ne fait que complexifier la recherche d'un algorithme selon différents besoins.

Afin de retrouver efficacement des algorithmes d'indexation qui répondent à des besoins utilisateurs, des contextes d'exécution et des propriétés des utilisateurs, il apparait nécessaire de spécifier une description générique d'un algorithme d'indexation. Grâce à notre description générique, des annuaires de descriptions d'algorithmes d'indexation en termes de paramètres, de performances, de contexte d'utilisation, de contraintes d'exécution pourront être construits.

Un de nos principaux objectifs consiste donc à offrir un guide de description d'algorithmes pour faciliter leur recherche, leur consultation, la comparaison de leurs résultats...

Au moment de la construction d'un nouveau système d'information ces descriptions seront utilisées pour rechercher et comparer des algorithmes d'indexation selon des critères discriminatoires (e.g., la description des objets en sortie, des paramètres d'entrée, etc.) afin de sélectionner les plus appropriés à exécuter dans le système d'indexation multimédia.

Dans ce chapitre, nous présentons notre proposition de modèle générique de description d'algorithmes d'indexation qui regroupe dans une structure générale leurs principales caractéristiques (§3.2).

3.2 Notre modèle de description d'algorithmes d'indexation

Dans la première partie, nous allons présenter notre démarche de construction du modèle de description d'algorithmes d'indexation (§3.2.1). Ensuite, nous montrons différentes représentations de notre modèle dans plusieurs formalismes : UML (§3.2.2), XML (§3.2.3) et RDF (§3.2.4).

Les lecteurs qui ne sont pas familiarisés avec UML, XML et RDF peuvent trouver dans l'annexe A, une brève présentation de ces technologies.

3.2.1 Caractéristiques générales et discriminatoires des algorithmes d'indexation

Pour construire notre modèle de description d'algorithmes d'indexation, nous avons effectué une étude sur une vingtaine d'outils d'indexation utilisés notamment dans le cadre de projets de recherche, tels que KLIMT [1], LINDO [2], [20]... En particulier, nous avons considéré des outils d'analyse et de segmentation de contenus audio, comme par exemple un outil de segmentation de contenu audio en segments de parole, de musique, de bruit et de silence ; un outil d'identification automatique des langues ;

1. `http://www.klimt-project.org`
2. `http://www.lindo-itea.eu/`

un outil de transcription automatique de la parole en texte (français ou anglais) ; un outil qui reconnait les locuteurs d'un discours, etc. De plus nous avons étudié des outils d'analyse et de segmentation de contenus vidéo, comme par exemple des outils de segmentation en plans ; un outil de détection de personnes ; un outil d'extraction d'images clés de vidéo... En ce qui concerne les images, nous avons examiné un outil de détection de visages ; un outil qui extrait les caractéristiques globales de l'image, etc. Pour les documents textuels, nous avons considéré un outil de segmentation selon le thème ; un outil de reconnaissance des entités nommées...

En observant les caractéristiques communes mais discriminatoires de tous ces algorithmes d'indexation, nous avons identifié six groupes de caractéristiques suivantes :

- **les caractéristiques générales**, comme par exemple le nom, l'auteur (personne ou entreprise), le type de média en entrée, le langage de programmation dans lequel l'algorithme est codé, l'URL où l'algorithme peut être localisé, la complexité de l'algorithme... ;
- **les paramètres d'entrée** que l'on connaît a priori (concernant les données d'apprentissage et de test, et les caractéristiques spécifiques pour chaque type d'entrée) et dont la modification peut engendrer des résultats différents ;
- **les caractéristiques de sortie** : le type et la description des objets qui sont produits par l'algorithme ;
- **les mesures de performance** : pour chaque ensemble de données de test cette catégorie indique quels sont les résultats obtenus par l'algorithme en terme, par exemple, de rappel et de précision ;
- **les contraintes d'exécution** : des conditions qui doivent être satisfaites pour que l'algorithme s'exécute dans les meilleures conditions ;
- **les chaînes d'algorithmes** : des algorithmes peuvent être composés de séquences d'exécution de différents algorithmes.

3.2.2 Représentation du modèle avec le formalisme UML

La représentation UML [3] du modèle que nous proposons est présentée dans la figure 2.3. Le modèle contient les caractéristiques qui ont été mentionnées dans la sous-section antérieure, structurées dans cinq classes principales (notons que les chaînes d'algorithmes ne sont pas identifiées dans une classe supplémentaire) : AlgorithmModel, InputParam, OutputObj, Performance et ExecutionConstraints.

3. `http://www.uml.org/`

La classe **AlgorithmModel** contient des caractéristiques générales d'un algorithme d'indexation (voir la section antérieure). Le modèle d'algorithme peut avoir :

- 0 ou 1 objet **InputParam** qui est composé de 0 ou 1 objet DataTest (les localisations et les descriptions des ensembles de données d'apprentissage et de tests) et d'un objet contenant des paramètres spécifiques pour chaque type de média en entrée (les classes AudioParam, Video/ImageParam, TextParam peuvent contenir : les caractéristiques multimédia analysées, les seuils qui peuvent être fixés avant le lancement de l'algorithme, etc.) ;
- un seul objet **OutputObj** qui peut consister d'un ou plusieurs objets multimédia et/ou un ou plusieurs objets métadonnées qui ont un type et une description en langage naturel ;
- 0 ou 1 objet **Performances** qui contient des mesures de résultats de l'algorithme pour chaque ensemble de données de tests ou en général en terme de précision, rappel, F mesure ou d'autres mesures qui peuvent être ajoutées à la volée ;
- 0 ou 1 objet **ExecutionConstraints** qui contient les contraintes de traitement (les hypothèses qui sont faites ou les conditions qui sont supposées d'être accomplies pour atteindre les résultats désirés), les contraintes concernant le média à l'entrée (dimension maximale, types acceptés, etc.) et les contraintes concernant la plateforme d'exécution de l'algorithme (le système d'exploitation, la puissance de calcul nécessaire, etc.).

Les chaînages sont représentés par la relation isChained autour de la classe AlgorithmModel. En effet, un algorithme global (GlobalAlgo) peut être composé d'une séquence d'AlgorithmModel.

3.2.3 Représentation des instances en XML

Notre modèle peut être instancié en utilisant XML [4]. Nous présentons un exemple d'instance XML du modèle pour un algorithme qui fait de la reconnaissance des piétons dans une image :

```
<?xml version="1.0"?>
 <AlgorithmModel AlgoAuthor="LINDO" AlgoComplexity="O(n)" AlgoName=
    "Pedestrian Recognition" AlgoScript="Java" AlgoURL="www.irit.fr
    /PedestrianRecognition" MediaType="Image"/>
     <InputParameters InputParamFileURL="www.irit.fr"
        InputParamFileFormat="Xml">
       <DataSet>
          <DataSetUrl>www.irit.fr</DataSetUrl>
```

4. http://www.w3.org/XML/

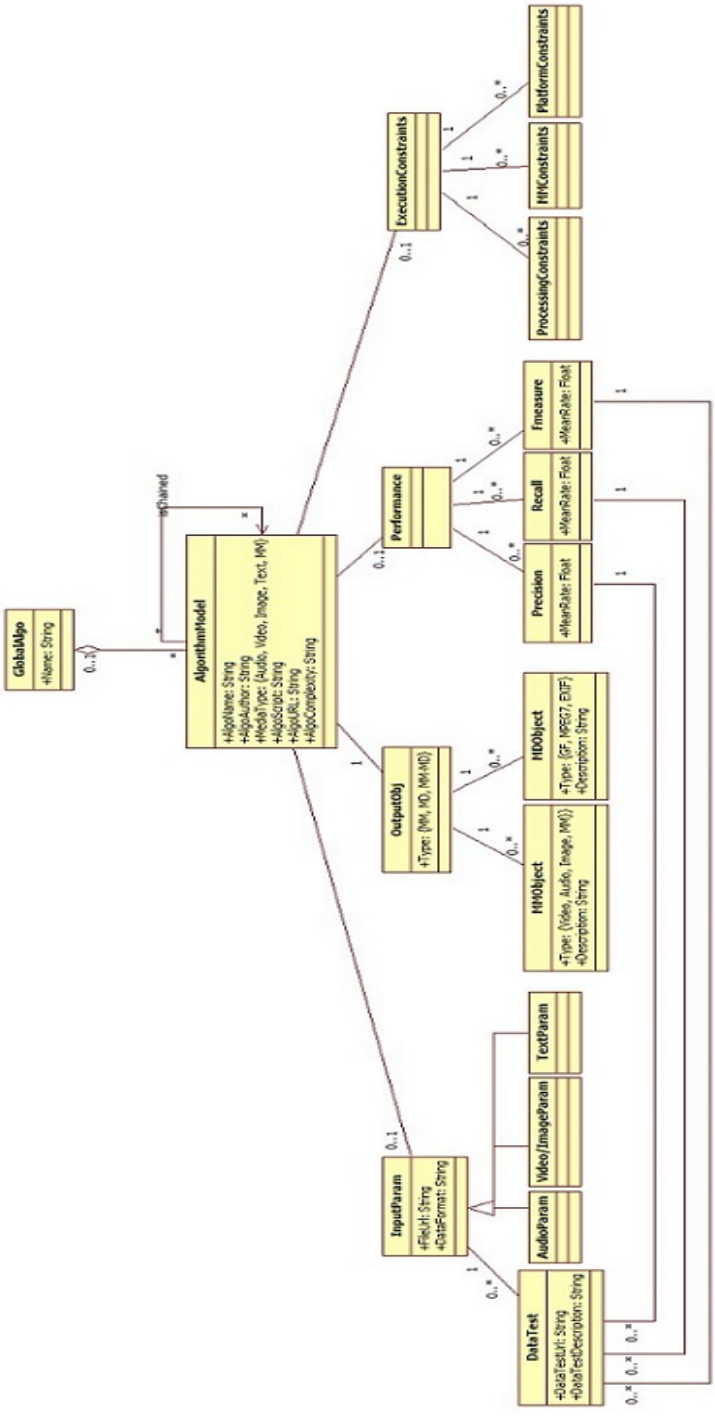


FIGURE 3.1 – Modèle générique de description d'algorithmes d'indexation

```
            <DataSetDescription>www.irit.fr</DataDescription>
        </DataSet>
        <Features Type="Primitive Features">
            <PrimitiveFeature>Color Histogram</PrimitiveFeature>
            <PrimitiveFeature>Spatial Location</PrimitiveFeature>
        </Features>
        <Threshold>0</Threshold>
        <CompressionStandard>uncompressed</CompressionStandard>
      <InputParameters>
      <OutputObject Type="Metadata">
        <MetadataObject Type="GenericFormat" Description="detected
            pedestrians with their position">
      </OutputObject>
      <Performances>
        <Precision>90</Precision>
      </Performances>
      <ExecutionConstraints>
        <ProcessingConstraint>the algorithm detects only standing
            pedestrians</ProcessingConstraint>
        <ProcessingConstraint>pedestrians must be vertical in the
            image</ProcessingConstraint>
        <MMConstraints>
            <DataFormat>jpeg,png</DataFormat>
            <MaxSize>5G</MaxSize>
        </MMConstraints>
        <PlatformConstraints>
            <OS>Linux</OS>
            <CPU>2GHz</CPU>
            <RAM>2G</RAM>
        </PlatformConstraints>
      </ExecutionConstraints>
  </AlgorithmModel>
```

L'exemple présenté représente une description d'un algorithme développé dans le cadre du projet LINDO (AlgoAuthor). L'algorithme s'appelle "Pedestrian Recognition" (AlgoName), prend des images en entrée (MediaType), il extrait des caractéristiques comme l'histogramme de couleurs et la localisation spatiale (PrimitiveFeatures) afin de détecter la position des piétons dans les images. Il va produire des métadonnées dans un format générique, développé dans notre équipe qui va contenir les informations extraites [6]. Notons que si nous ne disposons pas de plusieurs ensembles de données de tests un seul ensemble de mesures de performances sera considéré (comme des valeurs moyennes de performance des algorithmes). .

Plusieurs descriptions d'algorithmes sont disponibles à l'adresse : *http : //www.irit.fr/~Sebastien.Laborie/LINDO/listingAlgos.php.*

Interrogation des instances du modèle avec XQuery

XQuery [5] est un langage qui permet de faire des requêtes sur des documents XML selon leur structure ou bien leurs contenus en se basant sur des expressions XPath [6]. Il offre la possibilité non seulement d'extraire des informations d'un document XML, ou d'une collection de documents XML, mais également d'effectuer des calculs complexes à partir des informations extraites et de reconstruire de nouveaux documents ou fragments XML.

Par conséquent, nous pouvons interroger la collection des instances XML de notre modèle en utilisant XQuery. Par exemple, si nous voulons rechercher le nom de l'algorithme d'une instance de notre modele (instance.xml) nous utiliserons la requête suivante :

```
for $x in doc("instance.xml") return $x//@AlgoName/text()
```

3.2.4 Représentation des instances en RDF

Afin d'utiliser les technologies issues du Web Sémantique, nous proposons également une représentation possible de notre modèle sous forme de description RDF [7].

Une partie d'une instance RDF/XML de notre modèle est présentée dans la suite :

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ex="http://example.org/">
<rdf:Description rdf:about="http://example.org/AlgorithmModel">
<ex:author>LINDO</ex:author>
<ex:complexity>O(n)</ex:complexity>
<ex:name>Pedestrian Recognition</ex:name>
<ex:script>Java</ex:script>
<ex:url>www.irit.fr/PedestrianRecognition</ex:url>
<ex:mediaType>Image</ex:mediaType>
<ex:hasAnObject>
<rdf:Description rdf:about="http://example.org/InputParameters">
```

5. http://www.w3.org/TR/xquery/
6. http://www.w3.org/TR/xpath/
7. http://www.w3.org/TR/REC-rdf-syntax/

```
<ex:fileUrl>www.irit.fr</ex:fileUrl>
<ex:fileFormat>Xml</ex:fileFormat>
<ex:isComposedOf>
<rdf:Description rdf:about="http://example.org/DataSet">
<ex:isDescribedOf>
<rdf:Description rdf:about="http://example.org/DataSetUrl">
<ex:value>www.irit.fr</ex:value>
</rdf:Description>
</ex:isDescribedOf>
<ex:isDescribedOf>
<rdf:Description rdf:about="http://example.org/DataSetDescription">
<ex:value>news broadcast</ex:value>
</rdf:Description>
</ex:isDescribedOf>
</rdf:Description>
</ex:isComposedOf>
<ex:isComposedOf>
<rdf:Description rdf:about="http://example.org/Feature">
<ex:Type>PrimitiveFeatures</ex:Type>
<ex:isComposedOf>
<rdf:Description rdf:about="http://example.org/PrimitiveFeature">
<ex:value>ColorHistogram</ex:value>
</rdf:Description>
</ex:isComposedOf>
<ex:isComposedOf>
<rdf:Description rdf:about="http://example.org/PrimitiveFeature">
<ex:value>SpatialLocation</ex:value>
</rdf:Description>
</ex:isComposedOf>
</rdf:Description>
</ex:isComposedOf>
</rdf:Description>
</ex:hasAnObject>
</rdf:Description>
</rdf:RDF>
```

Nous avons repris l'exemple de la section 3.2.3. Quelques exemples de triplets issues de ce graphe sont :

– {AlgorithmModel author LINDO} ;
– {AlgorithmModel name Pedestrian Recognition} ;
– {AlgorithmModel hasAnObject InputParameters } ;
– {InputParameters fileUrl www.irit.fr } ;

Interrogation des instances du modèle avec SPARQL

SPARQL [8] est un langage qui permet d'interroger des graphes RDF. SPARQL permet de trouver un ensemble de solutions, s'il existe une solution et construire de nouveaux graphes basés sur des résultats.

La requête SPARQL qui extrait le nom de l'algorithme d'une instance de notre modèle (instance.rdf) en RDF est la suivante :

```
SELECT ?n
FROM <instance.rdf>
WHERE {
ex :AlgorithmModel rdf :nom ?n.
}
```

3.3 Conclusion

Dans ce chapitre, nous avons présenté notre proposition d'un modèle de description d'algorithme d'indexation. Notre modèle est générique, car il permet de mutualiser différentes caractéristiques communes et discriminatoires des algorithmes d'indexation. Également, il peut être formalisé en utilisant plusieurs langages de modélisation (UML, XML, RDF, etc.) et il est dynamique, parce que celui-ci est extensible.

Les principaux avantages d'avoir un tel modèle consistent dans le fait qu'il constitue un guide de description d'algorithmes d'indexation qui n'existe pas actuellement. Il permet également de retrouver et de comparer des algorithmes d'indexation en fonction de plusieurs critères (e.g., performances, contraintes d'exécution, complexité) qui sont organisés dans une structure générale.

8. `http://www.w3.org/TR/rdf-sparql-query/`

Chapitre 4

Processus de recherche d'algorithmes d'indexation

4.1 Introduction

Nous avons déjà fait l'observation dans le chapitre 2 que même s'il existe de nombreux systèmes gérant de multiples algorithmes d'indexation, aucun d'entre eux ne définit une méthode automatique déterminant des algorithmes d'indexation appropriés par rapport aux besoins des utilisateurs et des contextes d'exécution.

Un processus automatique de recherche d'algorithmes d'indexation consiste à déterminer une liste d'algorithmes d'indexation qui permet d'extraire des éléments désirés par l'utilisateur (Figure 4.1) et qui sont appropriés à ses préférences exprimées par l'intermédiaire de propriétés et de contextes.

Par rapport à la figure 4.1 nous pouvons faire les remarques suivantes :
- L peut être vide=> il n'existe aucun algorithme qui extrait les éléments désirés ;
- L n'identifie pas tous les t_i=> il existe des éléments qui ne sont pas extraits par la collection d'algorithmes ;
- L peut être composé de listes d'algorithmes => plusieurs candidats ;
- chaque algorithme de L peut être équivalent avec une chaine d'algorithmes =>liste des listes de chaines.

Dans la suite de ce chapitre, nous allons présenter la proposition de procédure de recherche d'algorithmes d'indexation qui exploite notre modèle, défini dans le chapitre précédent, afin de sélectionner des algorithmes d'indexation à partir des besoins

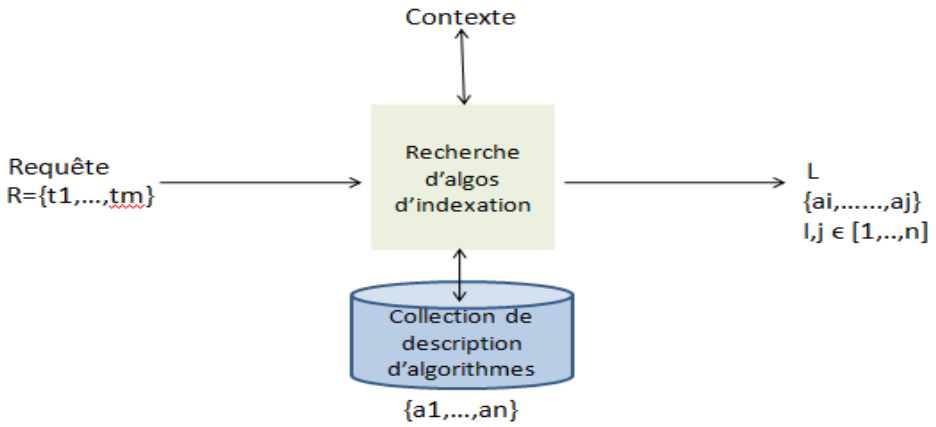


FIGURE 4.1 – Processus de recherche d'algorithmes d'indexation

des utilisateurs, de leur préférences exprimées par des propriétés qui rendent notre algorithme flexible et des éléments du contexte. Dans un premier temps nous allons expliquer les étapes principales de notre procédure (§4.2). Ensuite, nous détaillerons l'algorithme que nous avons développé (§4.3). Enfin, dans la section 4.4 nous allons présenter un exemple de déroulement de l'algorithme.

4.2 Description de la procédure de sélection d'algorithmes d'indexation

Les éléments qui vont être considérés à l'entrée de notre procédure de sélection sont :
- la requête (sous forme de termes) R=t_1, t_2,...,t_m ;
- les descriptions d'algorithmes d'indexation correspondant au modèle que nous avons présenté dans la figure 3.1 ;
- le contexte de chaque serveur qui fait partie du système d'information multimédia dans le cadre duquel l'algorithme est executé. Le contexte contient des éléments comme le système d'exploitation, le type et la fréquence du processeur, les caractéristiques du dispositif (s'il est capable de traiter du texte, des images, des vidéos, etc.), les caractéristiques du réseaux, etc. Nous allons considérer trois éléments pour le contexte : C = {OS, Processor, RAM} ;
- des propriétés et des contraintes selon les préférences de l'utilisateur comme le type de média recherché (des émissions TV, des journaux TV, des discours audio,

etc.), le format de média recherché, le nombre maximum d'algorithmes solutions, des préférences pour les algorithmes qui extraient beaucoup d'éléments ou pour des algorithmes qui extraient moins d'éléments, des préférences pour des chaînes d'algorithmes longues ou plus courtes, des préférences sur la redondance de caractéristiques extraites... Nous allons prendre en considération les propriétés et la contrainte suivantes :

- C1 : le format de média recherché : Audio, Video, Image, Texte, AudioVisuel, ImageText ;
- P1 : les algorithmes qui extraient beaucoup d'éléments ou les algorithmes qui extraient peu d'éléments ;
- P2 : la redondance des caractéristiques ou non ;
- P3 : les chaines longues ou les chaines courtes ;
- P4 : le nombre maximum d'algorithmes ;

Notre procédure de sélection d'algorithme d'indexation va suivre les étapes suivantes :

Étape 1 : Calcul des listes d'algorithmes qui extraient les éléments de la requête

a) Pour chaque mot de la requête, une liste d'algorithmes d'indexation qui l'extrait est construite (chaque mot est cherché dans le champ Description de l'OutputObject du modèle) ;

b) Les listes seront triées selon les éléments du contexte. Les algorithmes qui ne respectent pas les conditions du contexte, comme par exemple l'OS et le processeur (ces valeurs seront comparées avec les champs de PlatformConstraints du modèle) sont éliminés ; Une liste des algorithmes distincts qui extraient les éléments de la requête est créée ;

Étape 2 : Calcul des listes inverses

a) Pour chaque algorithme sélectionné à la fin de l'étape 1, on calcule la liste des éléments de la requête qu'il extrait à partir de la liste construite dans l'étape précédente ;

b) Un score est associé à chaque algorithme. Ce score dépend du nombre d'éléments de la requête qu'il extrait ;

Étape 3 Calcul des listes du résultat

a) Tri de la liste des algorithmes selon les scores. Si l'on souhaite les algorithmes qui extraient le plus d'éléments, le tri sera décroissant. Sinon, le tri sera croissant ;

b) Une procédure de recherche qui prend en entrée la liste triée des algorithmes et des scores et qui va produire la liste des listes d'algorithmes du résultat ; En considérant

à chaque pas les algorithmes ayant le score maximum ou minimum (en fonction de P1), la procédure vérifie si les algorithmes qui ont été déjà considérés extraient tous les éléments de la requête. Dans le cas contraire, on cherche d'autres algorithmes complémentaires. À chaque pas les scores sont recalculés en fonction de l'algorithme qui est traité. La procédure prend également en considération les propriétés P2 et P4.

À la fin de cette étape, la liste des résultats est parcourue et les doublons sont éliminés.

Étape 4 Calcul des chaînes d'algorithmes

a) Pour chaque algorithme du résultat (calculé à l'étape 3), on extrait les chaînes qui contiennent l'algorithme traité (si elles existent). Pour chaque chaîne identifiée, en partant de la position de l'algorithme du résultat, on commence à parcourir la chaîne en arrière. Pour chaque algorithme considéré, la procédure d'extraction de chainages vérifie que le type de média accepté en entrée (champ InputMediaType du modèle) correspond à la contrainte C1 définie par l'utilisateur. Dès qu'une solution est trouvée la chaine est gardée dans le résultat et le parcours de la chaine est arrêté. Si pour un algorithme il existe plusieurs chaines qui accomplissent les conditions, une seule chaine est sélectionnée (en fonction de la propriété P3) ; Si aucune chaine ne respecte cette condition, la liste qui contient l'algorithme est éliminée du résultat.

À la fin de cette étape, on dispose de listes contenant des chaînes d'algorithmes qui répond à la requête, e.g., des résultats de la forme [[A1→A2→A3] et A5] et [[A0→A6→A7→A3] et A5] ;

b) Pour les listes résultantes, on vérifie la propriété du nombre maximum d'algorithmes. Les listes qui contiennent un nombre d'algorithmes plus grand que le nombre précisé par l'utilisateur (P4) est éliminée ;

c) Les listes résultats sont ordonnées de façon ascendante selon les tailles des résultats (le nombre d'algorithmes qui font partie d'une liste du résultat) ;

4.3 Algorithme de recherche proposé

Algorithm 1: Premières trois étapes de l'algorithme

Input: Une liste de mots clés qui constitue la requête R, une liste avec les valeurs des éléments du contexte C et les valeurs des propriétés $P1$, $P2$, et $P4$.

Output: $\mathcal{L}$ est la liste des listes des algorithmes qui correspond à la requête, au contexte et aux propriétés données.

```
1  for chaque élément R_i de R do
2  |  for chaque algorithme A_j de la base de données do
3  |  |  if OutputObjetDescription(A_j) contains R_i then
4  |  |  |  if OS(C) = PlatformConstraintsOS(A_j) et
   |  |  |  CPU(C) >= PlatformConstraintsCPU(A_j) then
5  |  |  |  |  AlgoList_i ← AlgoList_j ∪ {A_j};
6  |  |  |  |  AlgoSolution ← AlgoSolution ∪ {A_j};
7  |  |  |  end
8  |  |  end
9  |  end
10 end
11 for chaque algorithme A_i de AlgoSolution do
12 |  ListeInverse_i = les elements de la requete qui sont extraits par A_i;
13 end
14 for chaque algorithme A_i de AlgoSolution do
15 |  Score_i = nombre d'elements de la requete qui sont extraits par A_i;
16 end
17 construction de la liste des listes AlgoInfo={{A_i, Score_i, ListeInverse_i}};
18 if P2=algorithmes qui extraient peu d'elements then
19 |  Trier(AlgoInfo,ascendant);
20 end
21 else
22 |  Trier(AlgoInfo,descendant);
23 end
24 Rechercher(AlgoInfo,∅,0) ;
25 éliminer les doublons de L ;
```

L'algorithme 4 correspond aux étapes 1,2,3 présentées dans la section 4.2. Il considère en entrée une collection de descriptions d'algorithmes d'indexation, une requête mots clés et il produit un résultat qui contient des listes des algorithmes qui peuvent extraire les éléments de la requête, en tenant compte des éléments du contexte et des propriétés.

Entre les lignes 1 et 10, l'étape 1 est réalisée. Pour chaque mot de la requête, une liste d'algorithmes qui l'extraient est construite (*AlgoList*). La liste des algorithmes différents qui extraient des éléments de la requête est *AlgoSolution*.

L'étape 2 est réalisée de la ligne 11 à la ligne 17, pendant que la troisième est effectuée dans les dernières lignes.

Dans la suite, nous allons détailler l'algorithme appliqué par la procédure principale de notre mécanisme, *Rechercher*, qui correspond à l'étape 3.b) présentée dans la section 4.2 :

Algorithm 2: Les entrées, les sorties et les données utilisée par la procédure Rechercher

Input: Les paramètres de la fonction sont :
- S1 : un sous ensemble de AlgoInfo, donc une liste d'algorithmes qui extraient des éléments de la requête ;
- S2 : l'ensemble d'algorithmes qui sont des bons candidats ;
- NombreElementsExtraits : nombre d'éléments de la requête qui peuvent être extraits avec S2 ;

Data:
- $\mathcal{L} \leftarrow vide$;
- $\mathcal{L}$incomplet $\leftarrow vide$;
- $ListeElementsExtraits \leftarrow vide$;
- NombreElementsExtraits=0 ;
- à chaque pas, la liste S1 va être triée selon les scores des algorithmes ;

Output: Une liste des listes de bons candidats

La procédure de recherche prend en entrée une liste triée selon les scores (en fonction de P1). Si les algorithmes considérés (S2) extraient tous les éléments de la requête S2 est ajoutée dans le résultat. Le cas où certains éléments de la requête peuvent être extraits avec les algorithmes dont on dispose est traité dans les lignes 4-6. Sinon, pour tous les algorithmes qui ont le score minimum ou maximum (lignes 10-15), on l'ajoute dans la liste des algorithmes traités (S2) et on calcule pour tous les autres des nouveaux scores (le nombre d'éléments qu'ils extraient en plus par rapport aux éléments qui sont extraits déjà par les algorithme de S2). La propriété P2 (redondance des caractéristiques) est traitée dans les lignes 22-28.

Algorithm 3: Rechercher

```
1  if NombreElementsExtraits=cardinalite(R) then
2  |  L.add(S2);
3  end
4  else if S1=Vide then
5  |  Lincomplet.add(S2);          /* certains élements de R sont extraits */
6  end
7  else
8  |  copyS1=clone(S1);                          /* Duplication de S1 et S2 */
9  |  copyS2=clone(S2);
10 |  if P1=algorithmes qui extraient beaucoup d'éléments then
11 |  |  listBestAlgos=les algorithmes A_i pour lesquels Score_i=Score_max;
12 |  end
13 |  else if P1=algorithmes qui extraient moins d'éléments then
14 |  |  listBestAlgos=les algorithmes A_i pour lesquels Score_i=Score_min;
15 |  end
16 |  copyS1=clone(S1);       /* CopyS1 va être amené à la valeur initiale */
17 |  for chaque algorithme A_i de listBestAlgos do
18 |  |  copyS1.remove(A_i);
19 |  |  copyS2.add(A_i);
20 |  |  for chaque algorithme A_j de copyS1 do
21 |  |  |  Score(A_j)=Cardinalite(ListeInverse(A_j)-ListeElementsExtraits);
22 |  |  |  if Score(A_j)=0 then
23 |  |  |  |  if P2=true then
24 |  |  |  |  |  copyS2.add(A_j); copyS1.eliminer(A_j);
25 |  |  |  |  end
26 |  |  |  |  else if P2=false then
27 |  |  |  |  |  copyS1.eliminer(A_j);
28 |  |  |  |  end
29 |  |  |  end
30 |  |  end
31 |  |  ListeElementsExtraits ← ListeElementsExtraits ∪ ListeInverse(A_i);
32 |  |  if copyS2.size()<=P4 then
33 |  |  |  Rechercher(copyS1,copyS2, NombreElementsExtraits+Score(A_i));
34 |  |  end
35 |  |  copyS1=clone(S1);
36 |  |  copyS2=clone(S2);
37 |  end
38 end
```

Algorithm 4: La fonction qui extrait la meilleure chaine

Input: Le nom de l'algorithme algoName, algorithmsChains=la liste des chaines définies a priori et le format de média désiré par l'utilisateur mediaFormat (définit par l'intermède de la contrainte C1)

Output: La sous chaine la plus courte qui finit avec algoName et commence avec un algorithme qui prend en entrée le mediaFormat

```
1  if dans algorithmsChains il y a au moins une chaine qui contient algoName then
2      for chaque chaine C_i qui contient algoName do
3          indexAlgo=la position de algoName dans la chaine;
4          for chaque algorithme A_j de la chaine C_i pour lequel j>indexAlgo do
5              if mediaType(A_j)=mediaFormat then
6                  extraction(subchain);
7                  relevantChains.add(subchain);
8              end
9          end
10     end
11     bestChain=minSize(relevantChains);
12 end
```

L'algorithme 4 correspond à l'étape 4 présentée dans la section 4.2. Il considère en entrée les listes qui constitue le résultat, issues de la troisième étape.

4.4 Exemple de déroulement de l'algorithme

Nous proposons d'illustrer notre procédure de recherche d'algorithmes d'indexation via un exemple.

Tout d'abord, nous énumérerons une partie de la collection de descriptions d'algorithmes. Pour chaque algorithme, nous présentons son nom, son type de média en entrée et ses contraintes de plateforme :

- A1 : Acoustic Segmentation : Audio, OS="Windows", Processor="2.2GHz" ;
- A2 : Car detection in a parking : Video, OS="Windows", Processor="2GHz" ;
- A3 : Cars color recognition : Video, OS="Windows", Processor="2GHz" ;
- A4 : Color recognition : Image, OS="Windows", Processor="2GHz" ;
- A5 : Demultiplexing : AudioVisual, OS="Windows", Processor="2GHz" ;
- A6 : Detecting Persons : Video, OS="Linux",Processor=" 2.7GHz" ;
- A7 : Face Recognition : Image, OS="Linux", Processor="2GHz" ;

- A8 : Feature Extractor : Video, OS="Windows", Processor="2.5GHz" ;
- A9 : Key Images Extractor : Video, OS="Windows", Processor="2GHz" ;
- A10 : Language Detection : Text, OS="Windows", Processor="2GHz" ;
- A11 : Named Entity Recognition : Text, OS="Windows", Processor="2GHz" ;
- A12 : Pedestrian Recognition : Image, OS="Windows", Processor="2GHz" ;
- A13 : Person detection : Image, OS="Windows", Processor="SGHz" ;
- A14 : Person detection in moving cars : Video, OS="Windows", Processor="2GHz" ;
- A15 : Scenes Resuming Images Extraction : Video, OS="Windows", Processor="2GHz" ;
- A16 : Shot change detection : Video, OS="Windows", Processor="2.5GHz" ;
- A17 : Speech Diarization : Audio, OS="Linux", Processor="2.5GHz" ;
- A18 : Topic Segmentation : Text, OS="Windows", Processor="2GHz" ;n

Le graphe suivant résume les chaines que nous avons définit :

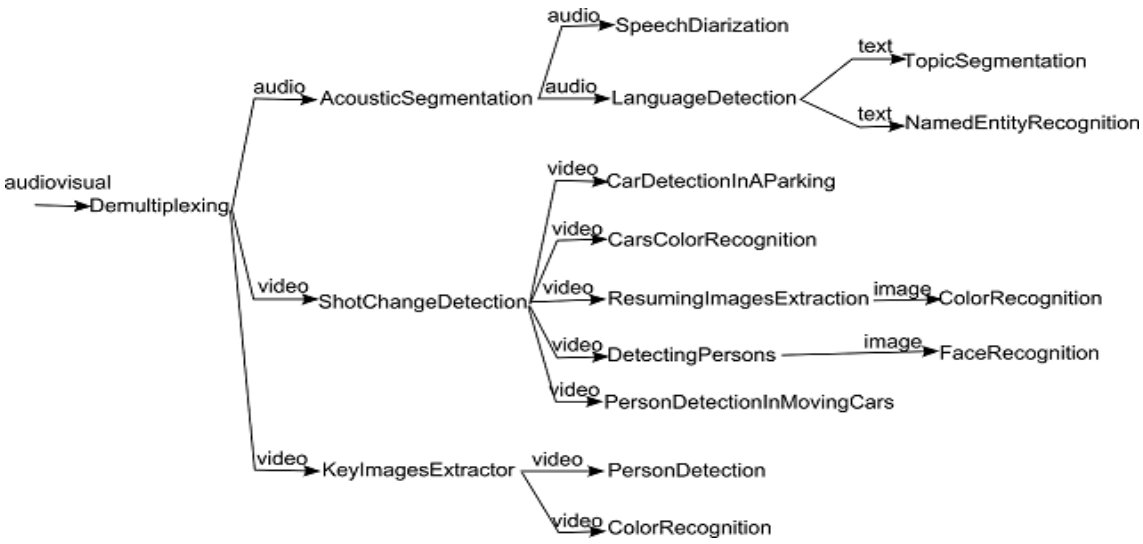


FIGURE 4.2 – Graphe de chaines d'algorithmes

Nous considérons la requête suivante : R={person,car,color,parking} ;

Nous allons considérer les préférences suivantes :

- nous cherchons des algorithmes qui prennent en entrée des vidéos ;
- nous voulons sortir en premier les algorithmes qui extraient beaucoup d'éléments ;
- nous voulons les chaines les plus courtes ;

– nous ne désirons pas des algorithmes qui extraient des éléments qui ont été déjà extraits ;

Étape 1 :

– $ListeAlgos_1$={A6, A13, A14} ;
– $ListeAlgos_2$={A2, A3, A4, A14} ;
– $ListeAlgos_3$={A3,A4, A8} ;
– $ListeALgos_4$={A2, A3, A4, A14} ;
– $ListeAlgosDifferents$={A2, A3, A4, A6, A8, A14} ;

Étape 2 :

– $ListeInverse_{A2}$={car,parking} ;
– $ListeInverse_{A3}$={car, color, parking} ;
– $ListeInverse_{A4}$={color, parking,car} ;
– $ListeInverse_{A6}$={person} ;
– $ListeInverse_{A8}$={color} ;
– $ListeInverse_{A13}$={person} ;
– $ListeInverse_{A14}$={person, car, parking} ;
– $Score_{A2}$=2 ;
– $Score_{A3}$=3 ;
– $Score_{A4}$=3 ;
– $Score_{A6}$=1 ;
– $Score_{A8}$=1 ;
– $Score_{A13}$=1 ;
– $Score_{A14}$=3 ;
– S={{A3,$ListeInverse_{A3}$,3}, {A4,$ListeInverse_{A4}$,3}, {A14,$ListeInverse_{A14}$,3}, {A2,$ListeInverse_{A2}$, 2}, {A6,$ListeInverse_{A6}$,1}, {A8,$ListeInverse_{A8}$,1}, {A13,$ListeInverse_{A13}$,1} } ;

Le schéma de la figure 4.3 présente le déroulement de la fonction Rechercher :

Après l'exécution de la fonction Rechercher nous allons avoir la liste suivante de solutions :

{{{A3, A6}, {A3, A13}, {A3, A14}}, {{A4, A6}, {A4, A13}, {A4, A14}}, {{A14, A3}, {A14, A4}, {A14, A8}}} ;

Après élimination des doublons (e.g., {A4, A14} et {A14, A4}), l'extraction des chaines et l'ordonnancement croissant par rapport aux nombres d'algorithmes composant la solution, nous aurons les résultats suivants :

1. {Person Detection in moving cars}, {Feature Extractor} ;
2. {Person Detection in moving cars}, {Color recognition} ;

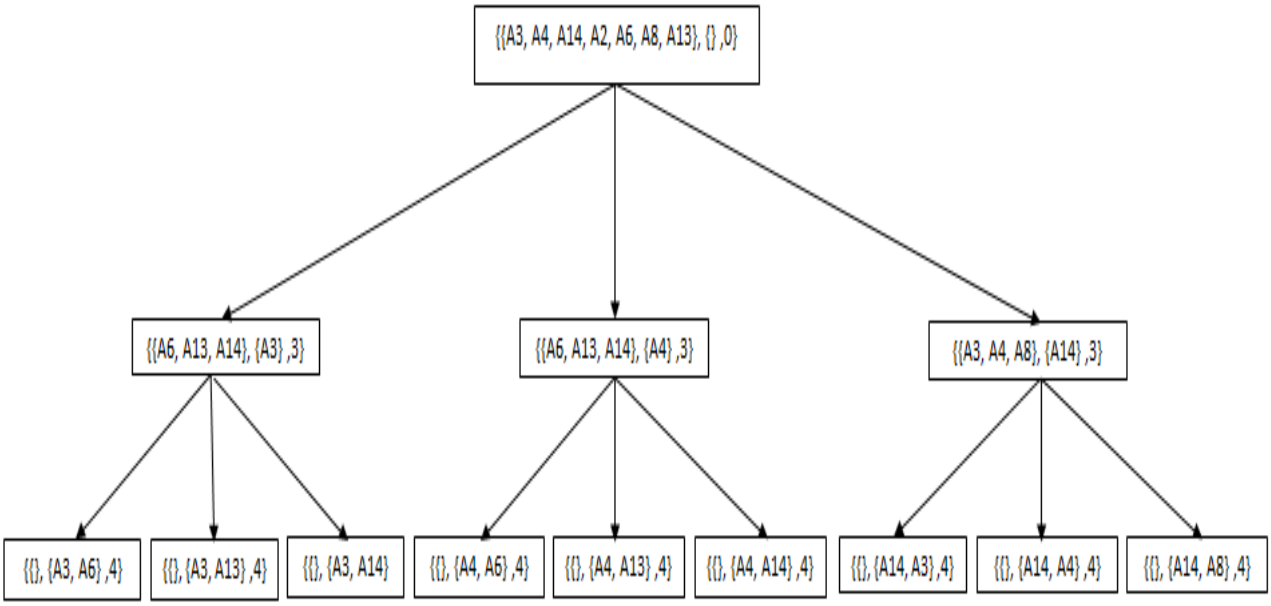


FIGURE 4.3 – Graphe des solutions

3. {Person Detection in moving cars}, {Cars color recognition} ;
4. {Color recognition}, {Detecting Persons } ;
5. {Cars color recognition}, {Detecting Persons } ;
6. {Color recognition}, {Key Images Extractor - > Person Detection} ;
7. {Cars color recognition}, {Key Images Extractor -> Person Detection}.

4.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode de recherche d'algorithmes d'indexation qui permet de répondre aux besoins et aux préférences de l'utilisateur, et à un contexte donné.

Les avantages de notre proposition sont les suivants :

– la flexibilité de recherche : notre mécanisme adapte le résultat en fonction des préférences de l'utilisateur définies lors de l'introduction de la requête. Il peut

opter pour des algorithmes qui extraient beaucoup ou moins de caractéristiques. Également, il peut choisir d'avoir des chaînes d'algorithmes longues ou des chaines plus courtes. Nous avons choisi de prendre en compte ce genre de propriétés sachant que plusieurs algorithmes qui extraient moins de caractéristiques peuvent être plus performants qu'un super algorithme qui extrait tout, mais la première solution peut être plus consommatrice de ressources ;
D'autres propriétés sont également prises en compte comme la redondance des caractéristiques (si pendant la procédure de recherche on trouve un algorithme qui n'extrait aucun élément nouveau, est ce que l'utilisateur le veut dans la solution ou non) et le nombre maximum d'algorithmes de la solution ;

- l'adaptation du résultat selon le contexte du serveur sur lequel l'algorithme devrait être exécuté ;
- notre algorithme réalise un filtrage (par rapport aux propriétés) et un ordonnancement des résultats (en fonction de la taille des résultats) ;
- le mécanisme que nous avons proposé interagit avec un modèle qui offre une description riche des algorithmes d'indexation ce qui permet d'envisager son utilisation dans d'autres types d'applications (qui supposent par exemple la comparaison des algorithmes selon les performances ou la complexité) ;
- l'algorithme n'est pas consommateur de ressources sachant qu'il interroge la base de données une fois lors de la première étape présentée dans la section 4.2 ;
- on peut trouver avant l'exécution de l'algorithme les mots de la requête pour lesquels on a pas d'algorithmes qui les extraient ;

Chapitre 5

Application

5.1 Introduction

Nous avons développé un prototype qui permet de créer et de stocker des descriptions d'algorithmes en se basant sur le modèle que nous avons présenté dans le Chapitre 2. En se basant sur les descriptions d'algorithmes, il permet également définir des chaines d'algorithmes et de rechercher des algorithmes d'indexation dans la collection par rapport à une requête mots clés, à des contraintes et à des propriétés définies par l'utilisateur et à des éléments du contexte.

Notre travail s'intègre dans le cadre du projet européen ITEA2 LINDO (Large Scale Distributed INDexation of multimedia Objects) qui propose une architecture distribuée et générique et qui permet l'indexation de contenus multimédias. Au lieu de déplacer des contenus volumineux vers un serveur central pour traitement, le projet propose d'indexer les contenus multimédias sur les sites distants et de déployer différents algorithmes d'indexation sur ces serveurs distants.

Dans ce chapitre nous allons expliquer l'application de notre prototype dans le cadre de l'architecture LINDO. Dans un premier temps (§5.2), nous allons présenter le projet en suivant trois aspects : contexte et présentation générale (§5.2.1), intégration de nos travaux dans l'architecture (§5.2.2) et les technologies utilisées dans le cadre du projet qui nous servirons aussi pour notre implémentation (§5.2.3). Ensuite, dans la section 5.3 nous allons présenter notre prototype.

5.2 Projet LINDO

5.2.1 Présentation générale du projet

Plusieurs domaines, comme la collecte des journaux, la TV, les collections de ressources pour les applications commerciales et dédiées aux consommateurs, le travail collaboratif, la vidéo surveillance, ont été inondées dans ces dernières années avec des sources multimédia qui génèrent du contenu volumineux et hétérogène. Tous ces domaines cherchent les meilleurs systèmes d'information multimédia répondant aux besoins de gérer leur contenu.

Dans ce contexte, le projet LINDO définit une solution architecturale distribuée avec un contrôle centralisé qui pourrait être un guide pour la conception des systèmes d'information.

L'architecture LINDO est divisée en deux composantes principales :

– les **serveurs distants** qui acquièrent, indexent et stockent des contenus multimédia ;
– le **serveur central** qui a une vision globale sur tout le système.

Le processus d'indexation adopté dans le cadre de LINDO est géré au niveau du serveur central et il est réalisé au niveau des serveurs distants. Le serveur central n'indexe pas de contenus multimédias, il déploie simplement des algorithmes d'indexation à la demande et répond à des requêtes.

Il existe deux processus d'indexation :
– une **indexation implicite** qui est exécutée a priori sur les contenus multimédia sur chaque serveur distant ;
– une **indexation explicite** qui pourrait être exécutée à la demande sur un serveur spécifique ;

5.2.2 Architecture

La solution architecturale proposée dans le cadre de LINDO offre les avantages suivants :

– chaque serveur local est indépendant, i.e., il a ses propres spécificités dans le cadre du système distribué, dépendant des différents contextes, comme la localisation et la capacité. Par exemple, certains serveurs distants pourraient indexer en temps

réel les contenus multimédia acquis, pendant que d'autres pourraient faire une indexation hors ligne.
– le serveur central peut déployer des algorithmes d'indexation ou des requêtes vers les serveurs distants, pendant que le système fonctionne.

L'entière architecture LINDO est présentée dans la Figure 5.1. La partie supérieure de la figure concerne le serveur central, pendant que la partie inférieure détaille le schéma d'un serveur distant.

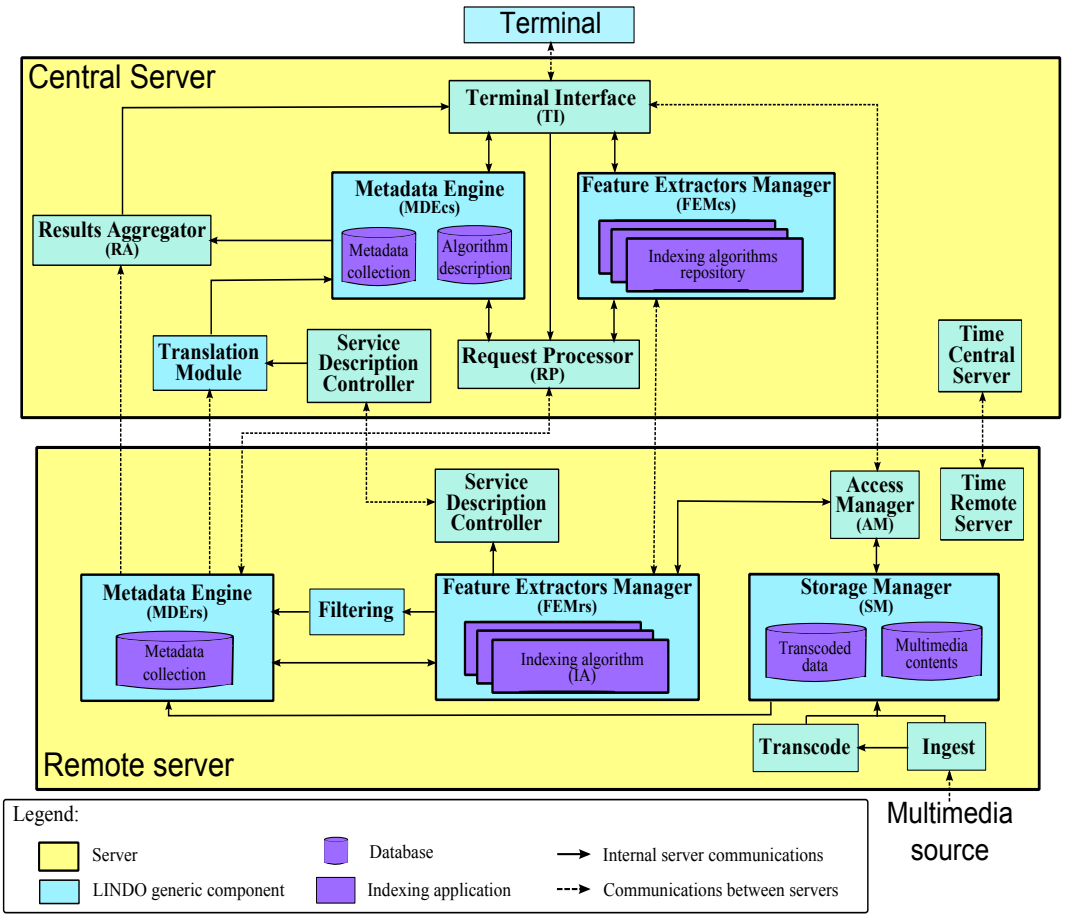


FIGURE 5.1 – Architecture LINDO

Dans la suite, nous allons décrire les fonctionnalités des modules des serveurs distants et du serveur central qui concernent notre contribution.

Les composants des serveurs distants

- **Feature Extractors Manager (FEM)** : est en charge avec la gestion et l'exécution d'un ensemble d'algorithmes d'indexation sur les contenus multimédia acquis. À chaque moment, de nouveaux algorithmes peuvent être installés dans ce module. Également, il est possible de supprimer des algorithmes ;
- **Metadata Engine (MDE)** : collecte et agrège toutes les métadonnées extraites sur les contenus multimédia. Les métadonnées stockées dans ce module peuvent être interrogées afin de retrouver des information désirées ;

Les composants du serveur central

- **Terminal Interface (TI)** : dans laquelle l'utilisateur peut spécifier des requêtes et où les résultats seront affichés ; Dans le cadre de l'interface graphique, plusieurs fonctions ont été développées afin de visualiser des collection de métadonnées et d'installer, de déployer et d'exécuter à distance des algorithmes ;
- **Metadata Engine (MDE)** : contient des informations sur les serveurs distants. Également, ce module peut contenir des résumés de métadonnées sur les contenus multimédia, des informations contextuelles sur le système, sur les serveurs distants (leur localisation, leur capacité, etc.) ;
- **Feature Extractors Manager (FEM)** : gère l'ensemble total d'algorithmes d'indexation utilisés dans le système ;
- **Request Processor (RP)** : traite des requêtes sur le MDE du serveur central ou il les redirige vers certains MDE des serveurs distants.

Dans la suite, nous illustrons les technologies utilisées dans le cadre du projet et desquelles nous nous sommes servi afin de réaliser notre prototype.

5.2.3 Technologies utilisées

Oracle Berkeley DB XML

Oracle Berkeley DB XML [1] est une base de données open source, qui offre l'accès basé sur XQuery aux documents stockés dans des containers et indexés selon leur contenu. Oracle Berkeley DB XML est construit ayant à la base Oracle Berkeley DB et hérite de la richesse de ses caractéristiques et attributs. Comme Oracle Berkeley

1. `http://www.oracle.com/database/berkeley-db/xml/index.html`

DB, l'outil dédié aux fichiers XML s'exécute dans un processus avec l'application sans le besoin d'une administration humaine. Oracle Berkeley DB XML ajoute un parseur de document, un indexeur XML et un moteur XQuery sur Oracle Berkeley DB afin de faciliter la recherche de données. Nous allons utiliser Oracle Berkeley DB XML pour stocker les instances de notre modèle et nous transformerons les requêtes mots clés des utilisateurs dans des requêtes XQuery.

XQuery

XQuery [2] est un langage de "programmation" puissant permettant d'extraire des données XML. Les types de données qui peuvent être interrogées par des requêtes XQuery sont : un seul "document" ou encore des collections sous forme de fichier, bases de données XML ou des arbres DOM. XQuery est un langage qui permet de faire des requêtes selon la structure ou encore les contenus en se basant sur des expressions XPath (version 2.0). Il offre la possibilité non seulement d'extraire des informations d'un document XML, ou d'une collection de documents XML, mais également d'effectuer des calculs complexes à partir des informations extraites et de reconstruire de nouveaux documents ou fragments XML. XQuery joue par rapport aux données XML un rôle similaire à celui du langage SQL vis-à-vis des données relationnelles, et l'on peut trouver des analogies entre ces deux langages. Il existe deux syntaxes distinctes pour XQuery :

- la syntaxe "naturelle" non-XML dite aussi FLWOR (prononcer flower), dont le nom vient des cinq clauses principales qui la composent (for, let, where, order by et return) ;
- la syntaxe XQueryX (pour « XML Syntax for XQuery »), dans laquelle une requête est un document XML. De ce fait, elle est beaucoup plus verbeuse et moins lisible que la précédente et est destinée à des manipulations formelles par des programmes (éventuellement eux-mêmes écrits en XQuery).

5.3 Prototype développé

Notre prototype comprend trois grandes parties :

- Une interface graphique (Java Applet) qui constitue un formulaire qui permet de créer des instances de notre modèle (voir Chapitre 3). Les instances issues sont des

2. `http://www.w3.org/TR/xquery/`

documents XML qui vont être stockées dans une base de données Oracle Berkeley DB XML. . Le formulaire peut être trouvé aussi en ligne à l'adresse suivante : $http://www.irit.fr/\tilde{}Sebastien.Laborie/LINDO/launch.html$. De cette façon, il est disponible pour les membres du projet qui peuvent le compléter et ajouter des descriptions d'algorithmes. Également, les instances sont aussi stockées sur le web à l'adresse : $http://www.irit.fr/\tilde{}Sebastien.Laborie/LINDO/listingAlgos.php$ pour une meilleure visualisation ;
- Une Applet qui permet de déclarer des chaines composées d'algorithmes qui se trouvent dans la collection de descriptions ;
- Une Applet qui permet de spécifier une requête, donner des valeurs aux propriétés et choisir un contexte ;

5.3.1 Création et stockage de modèles d'algorithmes

La Figure 5.2 représente une copie écran de l'applet qui produit des instances de notre modèle.

Le formulaire a six onglets. Chaque onglet correspond à une classe du modèle (sachant que même si on a plusieurs onglets qui concerne les paramètres d'entrée, seulement un va être actif en fonction du type de média en entrée choisi dans le premier onglet).

L'utilisateur remplit les suivants champs :
- Dans l'onglet AlgorithmDescription (qui correspond à la classe AlgorithmModel) : le type de média en entrée, le nom de l'algorithme, le langage de programmation dans lequel l'algorithme est implémenté, l'URL où l'on peut trouver l'algorithme, la complexité et le nom de l'auteur (personne ou entreprise) ; Le nom de l'algorithme représente un champ obligatoire, car il représente aussi le nom du fichier dans lequel la description va être enregistrée (lors d'un enregistrement d'une description, si le nom existe déjà dans la base de données, un message d'erreur est affiché) ;
- Dans l'onglet des paramètres d'entrée correspondant au type de média coché : si l'algorithme reçoit en entrée un fichier avec les paramètres d'input, l'utilisateur peut indiquer le format du fichier ; Également, il spécifie les caractéristiques multimédia qui sont extraites par l'algorithme (le formulaire contient des boites à cocher correspondant aux caractéristiques les plus utilisés, mais il offre aussi la possibilité d'introduire de nouveaux descripteurs), et les URLs des ensembles de données d'apprentissage et de test ;
- Dans l'onglet correspondant à la classe OutputObject, l'utilisateur peut spécifier le type et la description des objets produits par l'algorithme ;

– Dans l'onglet correspondant à la classe Performances, pour chaque ensemble de données de test, l'utilisateur peut introduire les résultats de l'algorithme en terme de rappel, précision et f mesure (ou il a la possibilité d'indiquer une autre mesure de performance) ;
– Dans l'onglet ExecutionConstraints, on a trois parties : Processing Constraints, dans laquelle on peut introduire des descriptions en langage naturel des contraintes liées à la bonne exécution de l'algorithme ; Input Media File Constraints : les formats acceptés en entrée par l'algorithme, la dimension maximale du fichier en entrée, ou des autres contraintes qui peuvent être introduites par l'utilisateur ; Platform Constraints : les caractéristiques des plateformes sur lesquelles l'algorithme peut être exécuté (système d'exploitation, fréquence minimum du processeur, RAM minimum, ou autres contraintes introduites par l'utilisateur).

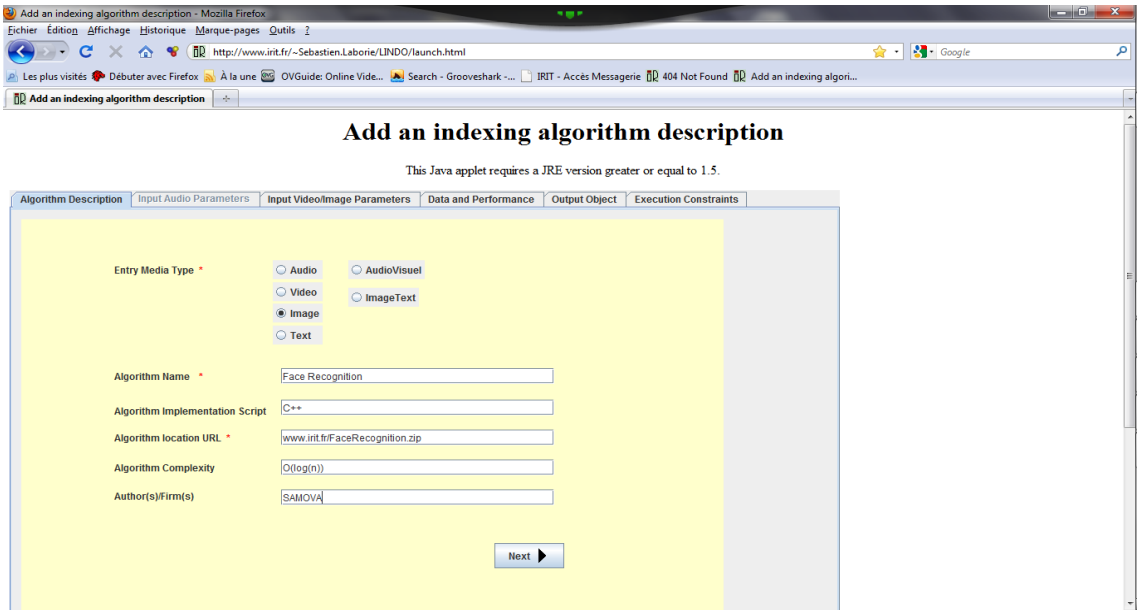


FIGURE 5.2 – Le formulaire qui permet d'instancier le modèle

5.3.2 Déclaration des chaines

Comme nous pouvons observer dans la Figure 5.3, l'applet qui permet de déclarer des chaines d'algorithmes contient trois zones d'affichage. Dans la partie gauche, nous avons une liste rangée par ordre alphabétique avec tous les algorithmes qui se trouvent dans la base de données. Cette liste peut être mise à jour en temps réel.

L'utilisateur définit une chaine qui va être affiché dans la partie droite de l'interface. Il peut enregistrer ou effacer la liste qui est en train de définir. En bas de l'applet nous avons une liste avec toutes les chaines qui existent dans notre système, y compris les chaines qui viennent d'être ajoutées. L'applet offre la possibilité d'effacer des chaines d'algorithmes.

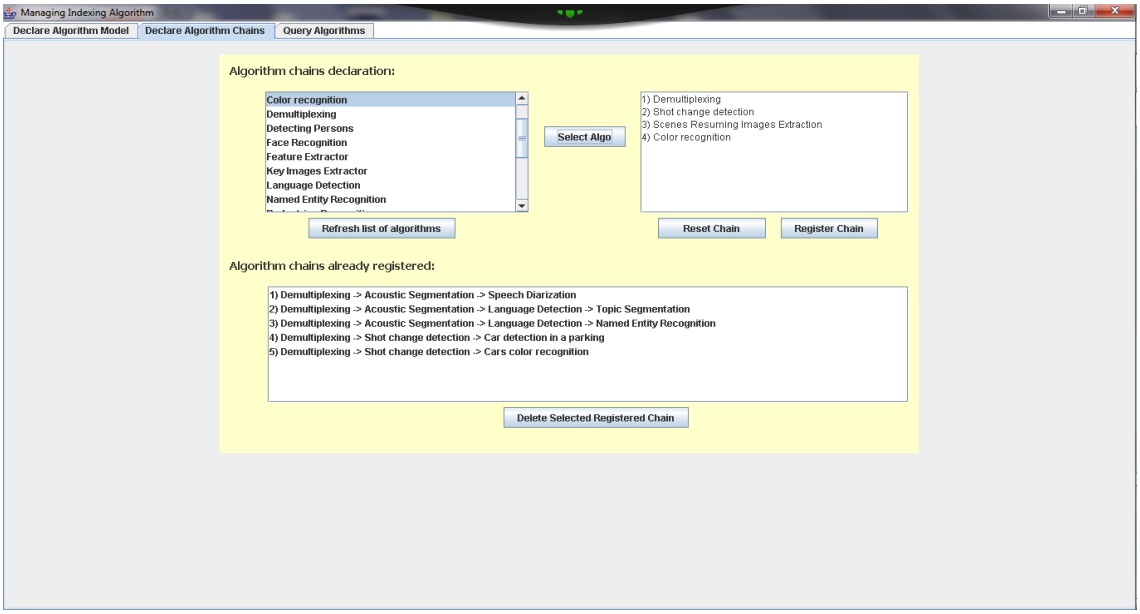


FIGURE 5.3 – L'Applet qui permet la définition des chaines d'algorithmes

5.3.3 Query Applet

L'applet qui est affichée dans la Figure 5.4 permet à l'utilisateur de spécifier une requête mots clés et d'exprimer ses préférences par l'intermède des propriétés dont nous avons présenté dans la section 4.2.

Également, l'utilisateur peut sélectionner un contexte particulier d'exécution des algorithmes d'indexation, en choisissant un élément de la liste qui est affichée. Chaque contexte de cette liste correspond à une description des capacités d'un serveur distant (e.g., système d'exploitation, fréquence du CPU, RAM). Le formalisme de description que nous avons utilisé pour décrire ces contextes est celui a été spécifié dans [44].

En bas de l'applet, nous pouvons observer la zone d'affichage des listes des résultats.

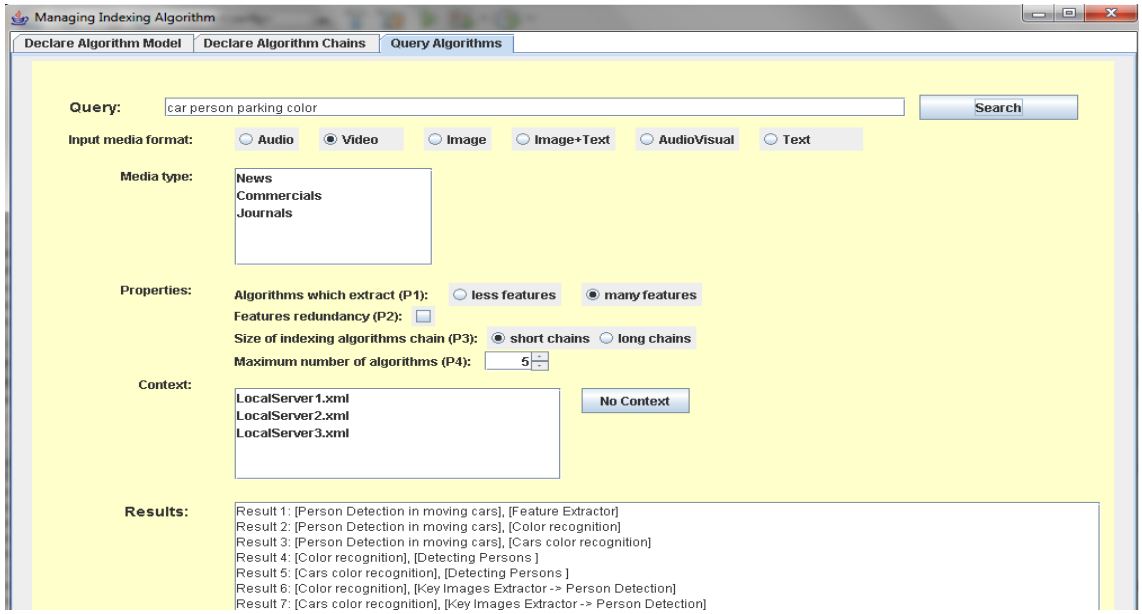


FIGURE 5.4 – L'Applet qui permet de spécifier une requête et d'instancier des contraintes et des propriétés

Un exemple d'un cas d'utilisation de notre prototype est illustré en Annexe B.

5.3.4 Application dans le cadre de LINDO

Dans le cadre de l'architecture LINDO presentée dans la Figure 5.1, notre prototype est intégré pour réaliser les processus d'indexation implicite et explicite. L'algorithme va être exécutée dans le cadre du Request Processor.

Les applets se trouvent dans le module Terminal Interface au niveau du serveur central. Chaque fois qu'un nouveau algorithme est ajouté dans le système, sa description va être enregistrée dans le Metadata Engine toujours sur le serveur central.

Pour trouver les algorithmes à exécuter lors de l'indexation implicite, notre procédure va être exécutée avec un ensemble de requêtes en entrée. Par exemple, nous pouvons prendre une statistique des requêtes faites dans le système dans la dernière période (peut être un mois, un trimestre, etc.), et exécuter notre algorithme sur une requête étant composée de tous les mots des requêtes.

Si pour une requête de l'utilisateur le système ne retourne aucun résultat, peut

être que les bons algorithmes d'indexation n'ont pas été exécutés. Par conséquent, la requête est transformée et elle est transmise à notre procédure, qui va rechercher dans la collection des descriptions les algorithmes appropriés par rapport à la requête.

5.3.5 Conclusion

Dans ce chapitre, nous avons fait une présentation du prototype que nous avons développé. Nous avons implémenté trois interfaces en Java qui servent à spécifier des descriptions d'algorithmes d'indexation, à définir des chaines d'algorithmes et à spécifier des requêtes et des préférences d'utilisateurs. Les instances des descriptions sont stockées dans une base de données Oracle Berkeley DB XML, et notre procédure se base sur la collection de descriptions afin de sélectionner des algorithmes appropriés par rapport à la requête, aux préférences de l'utilisateur et au contexte.

Nous avons montré également que que nos travaux s'intègrent dans le cadre du projet ITEA2 LINDO pour indexer implicitement et explicitement des contenus multimédia.

Les principaux avantages offerts par notre prototype sont :

– les interfaces développées sont faciles à utiliser (la durée moyenne de temps requise par le remplissage complet du formulaire de description d'algorithmes est d'approximativement 5 minutes) ;
– le fait que l'applet de description d'algorithmes est disponible en ligne le rend accessible aux développeurs qui veulent introduire des descriptions ;
– le fait d'implémenter dans le cadre d'un système d'information multimédia qui gère des contenus volumineux et distribués les processus d'indexation implicite et explicite rend l'indexation beaucoup plus efficace.

Chapitre 6

Conclusion et perspectives

Dans ce mémoire, nous avons proposé un modèle générique de descriptions d'algorithmes d'indexation et un processus automatique de sélection d'algorithmes d'indexation qui correspond aux besoins et aux préférences des utilisateurs liées à différents contextes donnés.

Nous avons montré l'utilité d'avoir un modèle de description d'algorithmes d'indexation qui réunisse des caractéristiques communes mais discriminatoires des algorithmes d'indexation et qui peut constituer une base pour des applications qui supposent la sélection et la comparaison des algorithmes selon différents critères.

La méthode proposée est utile afin de mettre en place les processus d'indexation implicite et explicite dans le cadre d'un système d'indexation distribué en temps réel de contenus multimédias.

Nous envisageons comme perspectives, d'utiliser notre modèle de description d'algorithmes d'indexation pour d'autres types d'applications, comme par exemple engendrer à partir de notre modèle des descriptions de type WSMO ou AWSDL. Également, nous souhaiterions déterminer implicitement des chaines d'applications d'algorithmes d'indexation par rapport aux entrées et aux sorties des algorithmes. En effet, actuellement l'utilisateur définit manuellement tous les chaînages possibles ce qui peut être très fastidieux s'il existe beaucoup d'algorithmes et de possibilités de combinaison.

Enfin, nous souhaiterions évaluer à plus large échelle notre prototype et mesurer la pertinence des résultats produits, la rapidité de temps de réponse ...

Nous proposons une synthèse de notre contribution et de nos perspectives dans la figure 6.1.

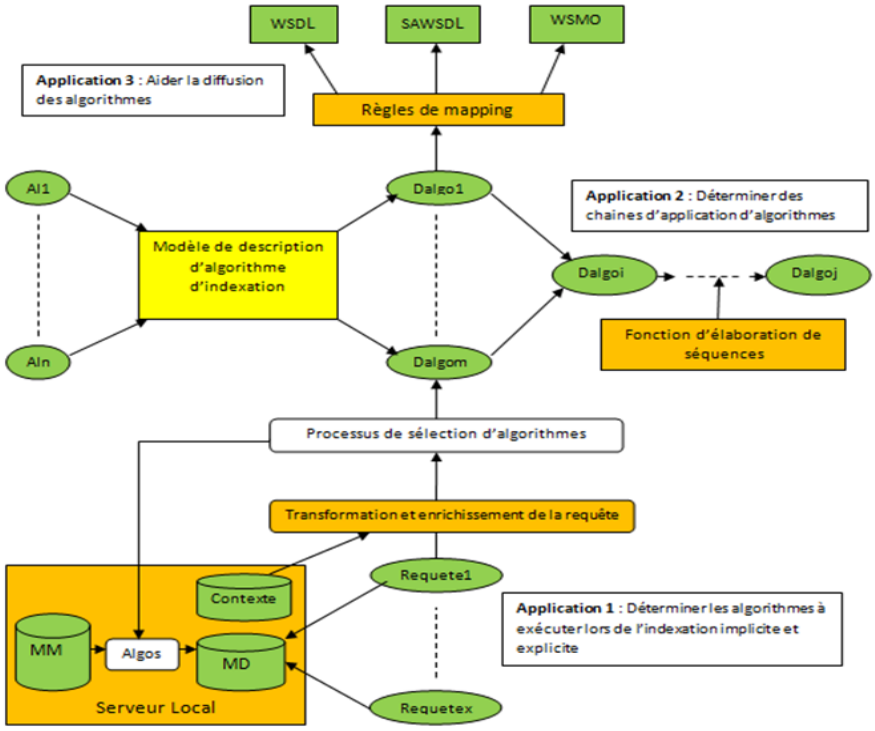


FIGURE 6.1 – Contribution et perspectives

Bibliographie

[1] Ikram Amous. *Méthodologies de conception d'applications hypermédia - Extension pour la réingénierie des sites Web.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 2002.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5) :34–43, 2001.

[4] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. *Storage and Retrieval for Still Image and Video Databases IV*, pages 170–179, 1996.

[5] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. Technical report, Computer Science Department, Stanford University, 1998.

[6] Mihaela Brut, Sébastien Laborie, Ana-Maria Manzat, and Florence Sèdes. A generic metadata framework for the indexation and the management of distributed multimedia contents. In *Proceedings of the Third International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE Computer Society, 2009.

[7] Michael K. Buckland and Christian Plaunt. On the construction of selection systems. *Library Hi Tech*, 12 :15–28, 1994.

[8] Claude Chrisment and Florence Sèdes. Annotations de média – Vers une représentation multidimensionnelle . *Ingénierie des Systèmes d'Information (ISI)*, 7(5-6) :45–65, décembre 2002.

[9] L. Couvreur, C. Ris, and C. Couvreur. Model-based blind estimation of reverberation time :application to robust asr in reverberents environments. *IEEE Transactions European Conference on Speech Communication and Technology (EUROSPEECH).*

[10] Bertrand Delezoide. *Modèles d'indexation multimedia pour la description automatique de films de cinéma*. PhD thesis, UNIVERSITÉ PARIS VI PIERRE ET MARIE CURIE, 2006.

[11] Bruce Devlin. Mxf-the material exchange format. *EBU TECHNICAL REVIEW*.

[12] John P. Eakins. Towards intelligent image retrieval. *Pattern Recognition*, 35(1) :3–14, 2002.

[13] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61 :2005, 2003.

[14] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, pages 264–271, 2003.

[15] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content : The qbic system. *Computer*, 28 :23–32, 1995.

[16] U. Gargi, R. Kasturi, and S. Antani. Performance characterization and comparison of video indexing algorithms. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 559–565, 1998.

[17] T. Gevers and A. W. M. Smeulders. Pictoseek : Combining color and shape invariant features for image retrieval, 2009.

[18] G. Golovchinsky. *From information retrieval to hypertext and back again : the role of interaction in the information exploration interface*. PhD thesis, University of Toronto, 1997.

[19] Bassem Haidar. *Services d'Indexation Multimédia Distribués*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 2005.

[20] Yoshihiko Hayashi, Katsutoshi Ohtsuki, Katsuji Bessho, Osamu Mizuno, Yoshihiro Matsuo, Shoichi Matsunaga, Minoru Hayashi, Takaaki Hasegawa, and Naruhiro Ikeda. Speech-based and video-supported indexing of multimedia broadcast news. In *SIGIR*, pages 441–442. ACM, 2003.

[21] Jing Huang, S Ravi Kumar, Mandar Mitra, Wei jing Zhu, and Ramin Zabih. Spatial color indexing and applications, 1998.

[22] ISO. Iso 15836 :2003 - information and documentation - the Dublin Core metadata element set. Standard, 2003.

[23] R. S. Jasinschi, N. Dimitrova, T. Mcgee, L. Agnihotri, J. Zimmerman, D. Li Philips, and D. Li. Integrated multimedia processing for topic segmentation and classification. In *IEEE International Conference on Image Processing*, pages 366–369, 2001.

[24] Andrew E. Johnson and Martial Hebert. Recognizing objects by matching oriented points. In *CVPR*, pages 684–689, 1997.

[25] Jacob Köhler, Stephan Philippi, Michael Specht, and Alexander Rüegg. Ontology based text indexing and querying for the semantic web. *Knowl.-Based Syst.*, 19(8) :744–754, 2006.

[26] Elie El Khoury. *Unsupervised video indexing based on audiovisual characterization of persons*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 2010.

[27] Harald Kosch and Paul Maier. Content-Based Image Retrieval Systems - Reviewing and Benchmarking . In *Proceedings of the 9th Workshop on Multimedia Metadata (WMM'09) held in conjunction with the 13th French Multimedia Conference on Compression and Representation of Audiovisual Signals (CORESA 2009)*, pages 1–21, 2009.

[28] Pierre-Yves Lambolez, Jean-Pierre Queille, Jean-Francois Voidrot, and Claude Chrisment. EXREP : a generic rewriting tool for textual information extraction. *Ingéniérie des Systèmes d'Information*, 3(4) :471–485, 1995.

[29] Frederick Wilfrid Lancaster, editor. *Information retrieval systems : characteristics, testing, and evaluation*. Information Science, 1979.

[30] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision (IJCV)*, 43(1) :29–44, 2001.

[31] Zhu Liu, Yao Wang, and Tsuhan Chen. Audio feature extraction and analysis for scene segmentation and classification. In *Journal of VLSI Signal Processing System*, pages 61–79, 1998.

[32] Lie Lu, Hao Jiang, and Hongjiang Zhang. A robust audio classification and segmentation method. In *ACM Multimedia*, pages 203–211, 2001.

[33] Stephane G. Mallat. A theory for multiresolution signal decomposition : the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11 :674–693, 1989.

[34] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 2002.

[35] K. Martin and Y. Kim. 2pmu9. instrument identification : a pattern-recognition approach. *136th Meet. Ac. Soc. of America*.

[36] J. Martinez. Mpeg-7 overview (version 10). http ://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm, Oct. 2005.

[37] Jianhao Meng, Yujen Juan, and Shih-Fu Chang. Scene change detection in a mpeg compressed video sequence. *Proc. SPIE*.

[38] Alessandro Micarelli, Filippo Sciarrone, and Mauro Marinilli. Web document modeling. In *Adaptive Web 2007*, pages 155–192, 2007.

[39] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. Generic object recognition with boosting. *PAMI*, 28 :2006, 2004.

[40] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1986.

[41] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28, 2002.

[42] E. Scheirer. *Music-Listening Systems*. PhD thesis, MIT Media Laboratory, 2000.

[43] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1997.

[44] Bouchra Soukkarieh and Florence Sèdes. Dynamic services adaptation to the user's context. In *ICIW '09 : Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services*, pages 223–228, Washington, DC, USA, 2009. IEEE Computer Society.

[45] S. R. Subramanya and Abdou Youssef. Wavelet-based indexing of audio data in audio/multimedia databases. In *Proceedings of MultiMedia Database Management Systems*, pages 46–53, 1998.

[46] Technical Standardization Committee on AV & IT Storage Systems and Equipment. Exchangeable image file format for digital still cameras : Exif Version 2.2. Technical Report JEITA CP-3451, April 2002.

[47] George Tzanetakis, Georg Essl, and Perry Cook. Audio analysis using the discrete wavelet transform. In *Proc. Conf. in Acoustics and Music Theory Applications. WSES*, 2001.

[48] Manik Varma and Andrew Zisserman. Classifying images of materials : Achieving viewpoint and illumination independence. In *Lecture Notes in Computer Science*, pages 255–271, 2002.

[49] Remco C. Veltkamp and Mirela Tanase. Content-based image retrieval systems : A survey. Technical report, 2000.

[50] Gang Wang and Ye Zhang Li Fei-fei. Using dependent regions for object categorization in a generative framework. In *CVPR*, pages 1597–1604. IEEE Computer Society, 2006.

[51] Itheri Yahiaoui, Nicolas Hervé, and Nozha Boujemaa. Shape-based image retrieval in botanical collections, 2006.

[52] Dongqing Zhang and Shih-Fu Chang. A generative-discriminative hybrid method for multi-view object detection. In *IEEE CVPR*, New York City, New York, June 2006.

[53] Tong Zhang and C.-C. Jay Kuo. Content-based classification and retrieval of audio. In *SPIE's 43rd Annual Meeting - Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, pages 432–443, 1998.

Annexe A : Concepts et outils utilisés

6.1 UML (Unified Modeling Language)

UML [1] est la spécification OMG (Object Management Group) la plus fréquemment utilisé pour modéliser différentes structures, architectures et comportements d'une application. Également, ce formalisme est utilisé pour spécifier des processus business ainsi que des structures de données.

Le métamodèle UML fournit une panoplie d'outils permettant de représenter l'ensemble des éléments du monde objet (classes, objets, ...) ainsi que les liens qui les relie.

Par exemple, le diagramme de classe de la Figure 6.2 représente une possible modélisation de la classe Lettre qui a une Origine (ayant comme attribut le nom de l'expéditeur) et une Destination (ayant comme attribut le nom et l'adresse du destinataire).

Nous avons fait une **modélisation des données** des descriptions des algorithmes d'indexation en utilisant le formalisme d'UML (Data Modeling [2]).

La modélisation de données représente l'exploration des structures orientées données. Comme des autres objets issus de la modélisation, les modèles de données peuvent être utilisés dans plusieurs buts, des modèles conceptuels de haut niveau aux modèles physiques de données. Du point de vue du développeur, la modélisation de données est similaire avec la modélisation de classes. Avec la modélisation de données, on identifie des types des entités pendant qu'avec la modélisation de classes on identifie

1. `http://www.uml.org/`
2. `http://www.agiledata.org/essays/dataModeling101.html`

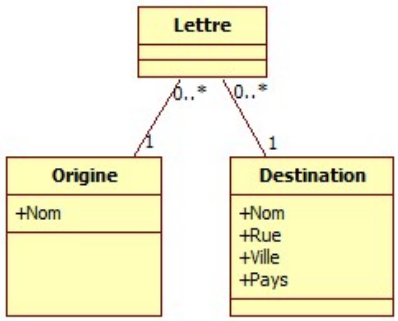


FIGURE 6.2 – Exemple de diagramme de classes UML

des classes. Les attributs des données sont assignés aux entités comme on assigne des types de données et des opérations à une classe. Ils existent des associations entre les entités, similaires avec les relations entre les classes : des relations, l'héritage, la composition et l'agrégation.

Notre modèle sera représenté par un diagramme de classe présentée dans la section 3.2.

6.2 XML (Extensible Markup Language)

XML[3] est un langage de balisage générique. À l'origine créé pour faire face au défi de la publication électronique à grande échelle, XML joue un rôle important dans l'échange d'une grande variété de données sur le Web. Le World Wide Web Consortium (W3C), promoteur de standards favorisant l'échange d'informations sur Internet, recommande la syntaxe XML pour exprimer des langages de balisages spécifiques. De nombreux langages respectent la syntaxe XML : XHTML[4], SVG[5], etc.

Différents outils autour des contenus XML permettent de vérifier qu'un document XML est conforme (ou valide) à une syntaxe donnée. Par exemple, la norme XML

3. http://www.w3.org/XML/
4. http://www.w3.org/MarkUp/
5. http://www.w3.org/Graphics/SVG/

définit une définition de type de document appelée DTD [6] (Document Type Definition), c'est-à-dire une grammaire permettant de vérifier la conformité du document XML. La norme XML n'impose pas l'utilisation d'une DTD pour un document XML, mais elle impose par contre le respect exact des règles de base de la norme XML (i.e., un document XML doit toujours être bien formé).

Un document XML est structuré en trois parties :

- la première partie, appelée prologue permet d'indiquer la version de la norme XML utilisée pour créer le document (cette indication est obligatoire) ainsi que le jeu de caractères (en anglais encoding) utilisé dans le document ;
- la deuxième partie est constituée de l'arbre d'éléments. Un document XML doit avoir une structure hiérarchique avec un seul élément racine. La règle d'imbrication simplifie les documents XML en leur conférant une structure d'arbre, ce qui signifie que tout élément fils est complètement inclus dans son père. Une balise (nommée également tag ou label) est une suite de caractères encadrés par ≪<≫ et ≪>≫, comme par exemple <nom>. Un élément XML est composé d'une balise d'ouverture <nom>, le contenu de l'élément et une balise de clôture </nom>. Des attributs peuvent être ajoutés aux balises, ayant la syntaxe ≪ nomattribut=“valeur” ≫ étant situés au début de l'élément après le nom de la balise. Par exemple :
  <document type=“article” > <titre> Bases Documentaires </titre> </document> ;
- la troisième partie est représentée par les commentaires qui peuvent apparaître dans n'importe quelle partie du document en dehors d'autres balises.

En prenant le même exemple que dans la section 6.1 sa modélisation XML pourrait être la suivante :

<Lettre>
  <Origine Nom=“nomO”>
  <Destination Nom=“nomD” Rue=“valeurRue” Ville=“valeurVille” Pays=“valeurPays”>
</Lettre>

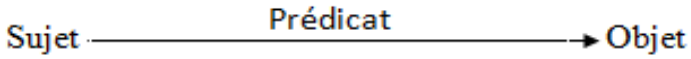
Les instances de descriptions d'algorithmes d'indexation (donc les instances de notre modèle) seront sauvegardées dans des fichiers XML.

6. http://www.w3.org/TR/xhtml1/dtds.html

6.3 RDF (Resource Description Framework)

RDF [7] est un modèle de graphe destiné à décrire de façon formelle les ressources du Web ainsi que les entités qui peuvent être identifiées de manière unique (par des URI), de façon à permettre le traitement automatique de telles descriptions. Développé par le W3C, RDF est le langage de base du Web sémantique [3].

Un graphe RDF est un ensemble de triplets :



- Le **sujet** représente la ressource à décrire (identifié par une URI) ;
- Le **prédicat** représente un type de propriété applicable à cette ressource (identifié par une URI) ;
- L'**objet** représente une donnée ou une autre ressource : c'est la valeur de la propriété (identifié par une URI ou par un littéral).

Un document RDF ainsi formé correspond à un graphe orienté étiqueté. Chaque triplet correspond alors à un arc orienté dont le label est le prédicat, le nœud source est le sujet et le nœud cible est l'objet (dans le formalisme graphique, l'URI est représentée par une ellipse et le littéral par un rectangle).

En reprenant l'exemple de la section 6.1, un graphe RDF possible qui lui correspond est illustré dans la Figure 6.3.

Les documents RDF peuvent être sérialisés dans différentes syntaxes. Deux exemples sont : RDF/XML [8] et Turtle [9].

Le langage RDF/XML [10](proposé par le W3C) offre une syntaxe XML pour décrire des graphes RDF. Le code RDF/XML pour le graphe de la Figure 6.3 est le suivant :

7. http://www.w3.org/TR/REC-rdf-syntax/
8. RDF/XMLhttp://www.w3.org/TR/REC-rdf-syntax/
9. http://www.w3.org/TeamSubmission/turtle/
10. http://www.w3.org/TR/rdf-syntax-grammar/

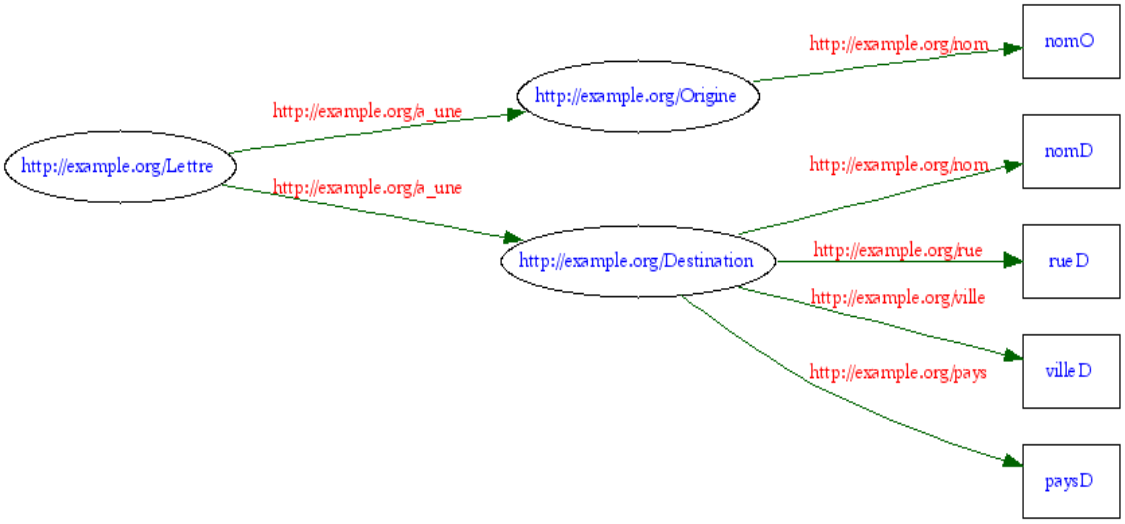


FIGURE 6.3 – Exemple de graphe RDF

```
<?xml version="1.0" ?>
<rdf :RDF xmlns :rdf="http ://www.w3.org/1999/02/22-rdf-syntax-ns"
xmlns :ex="http ://example.org/">
<rdf :Description rdf :about="http ://example.org/Lettre">
<ex :a une>
<rdf :Description rdf :about="http ://example.org/Origine">
<ex :nom>nomO</ex :nom>
</rdf :Description>
</ex :a une>
<ex :a une>
<rdf :Description rdf :about="http ://example.org/Destination">
<ex :nom>nomD</ex :nom>
<ex :rue>rueD</ex :rue>
<ex :ville>villeD</ex :ville>
<ex :pays>paysD</ex :pays>
</rdf :Description>
</ex :a une>
</rdf :Description>
</rdf :RDF>
```

Le langage Turtle offre une syntaxe plus légère et lisible pour décrire des graphes RDF. La variante Turtle pour le graphe de la Figure 6.3 est :

```
@prefix rdf :<“http ://www.w3.org/1999/02/22-rdf-syntax-ns”>.
@prefix ex :<“http ://example.org/”>.
ex :Lettre ex :a une ex :Origine.
ex :Origine ex :nom “nomO”.
ex :Lettre ex :a une ex :Destination.
ex :Destination ex :nom “nomD”.
ex :Destination ex :rue “rueD”.
ex :Destination ex :ville “villeD”.
ex :Destination ex :pays “paysD”.
```

Annexe B : Exemple de cas d'utilisation du prototype

Afin de mieux expliquer le fonctionnement de notre prototype nous allons présenter un exemple d'exécution du prototype qui va montrer toutes ses fonctionnalités.

Nous allons reprendre l'exemple de la section 4.4. Nous allons considérer la collection de descriptions déjà présentée et la requête R={person, car, parking, color}.

Pour le premier ensemble d'hypothèses :

- nous ne considérons pas de contexte ;
- nous cherchons des algorithmes qui prennent en entrée des vidéos ;
- nous voulons sortir en premier les algorithmes qui extraient beaucoup d'éléments ;
- nous voulons les chaines les plus courtes ;
- nous ne désirons pas des algorithmes qui extraient des éléments qui ont été déjà extraits ;

Nous allons avoir le resultat qui est presenté dans la figure 6.3. Nous pouvons observer que les resultats affichés correspondent avec ceux presentés dans la section 4.4.

Si nous allons sélectionner le contexte du premier serveur distant qui est stocké dans un fichier XML présenté dans la suite, nous pouvons observer que la liste de résultats a été modifiée (6.5). Les listes contenant les algorithmes : Detecting Persons et Person Detection n'existent plus car ces deux ne respectent pas la condition OS=Windows ;

```
<?xml version="1.0" encoding="UTF-8"?>
<context>
<Hardware>
          <DeviceType>portable</DeviceType>
          <DeviceName>sony</DeviceName>
          <Memory>4</Memory>
          <ScreenWeidth>1280</ScreenWeidth>
```

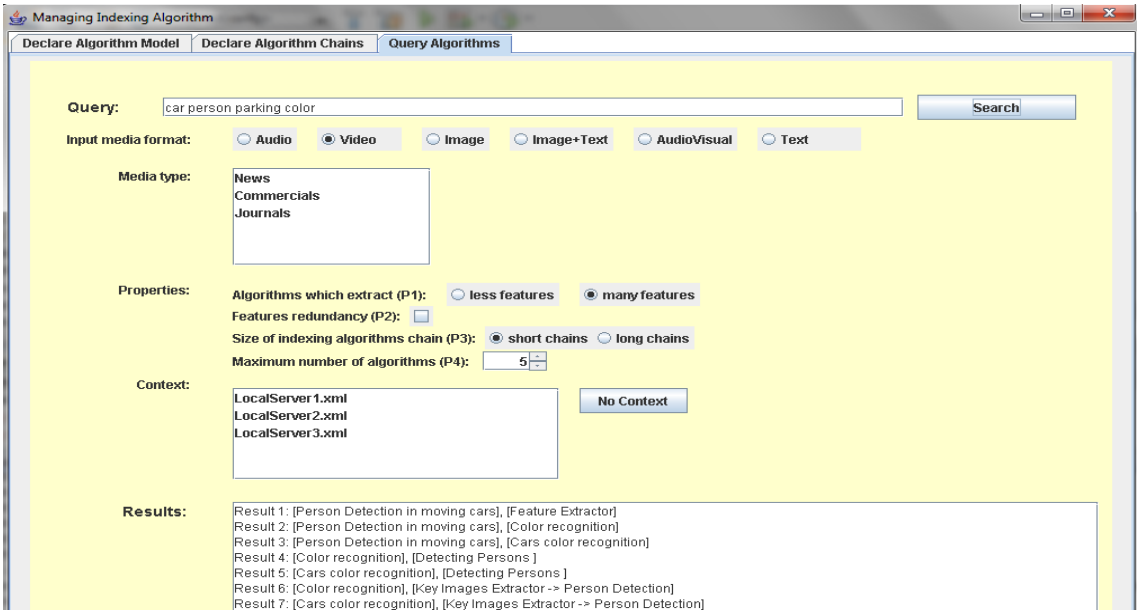


FIGURE 6.4 – L'Applet qui permet d'introduire une requête et d'instancier des contraintes et des propriétés

```
          <Screenheidth>800</Screenheidth>
          <processor>2.5</processor>
          <TextCapable>yes</TextCapable>
          <ImageCapable>yes</ImageCapable>
          <VideoCapable>yes</VideoCapable>
          <AudioCapable>yes</AudioCapable>
</Hardware>
        <Software>
          <OperatingSystem>Windows</OperatingSystem>

         </Software>
    <Network>
      <NetworkProfile>
        <NetworkType>Wifi</NetworkType>
        <Bandwidth>690</Bandwidth>
      </NetworkProfile>
    </Network>

</context>
```

En plus, si nous choisissons comme type de média en entrée *AudioVisual* nous aurons

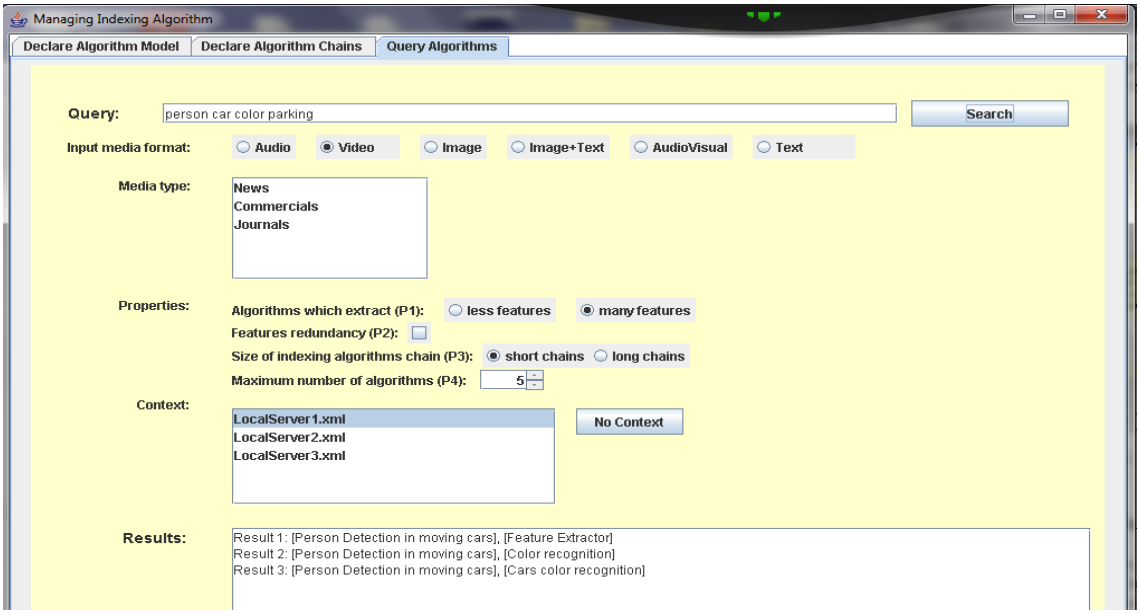


FIGURE 6.5 – Les résultats après avoir choisit un contexte

le résultat de la figure 6.6. Nous pouvons observer que nous avons seulement deux résultats (le résultat qui contient l'algorithme Feature Extractor est éliminé parce qu'il n'existe aucune chaine qui contient cet algorithme et qui peut traiter du contenu Audiovisuel). Comme aucun algorithme du résultat ne permettait le traitement du contenu audiovisuel on va avoir une chaine pour chacun d'entre eux. Notons qu'on a augmenté aussi le nombre maximum d'algorithmes. Si on avait laissé la valeur initiale, on n'aurait eu aucun résultat (figure 6.7).

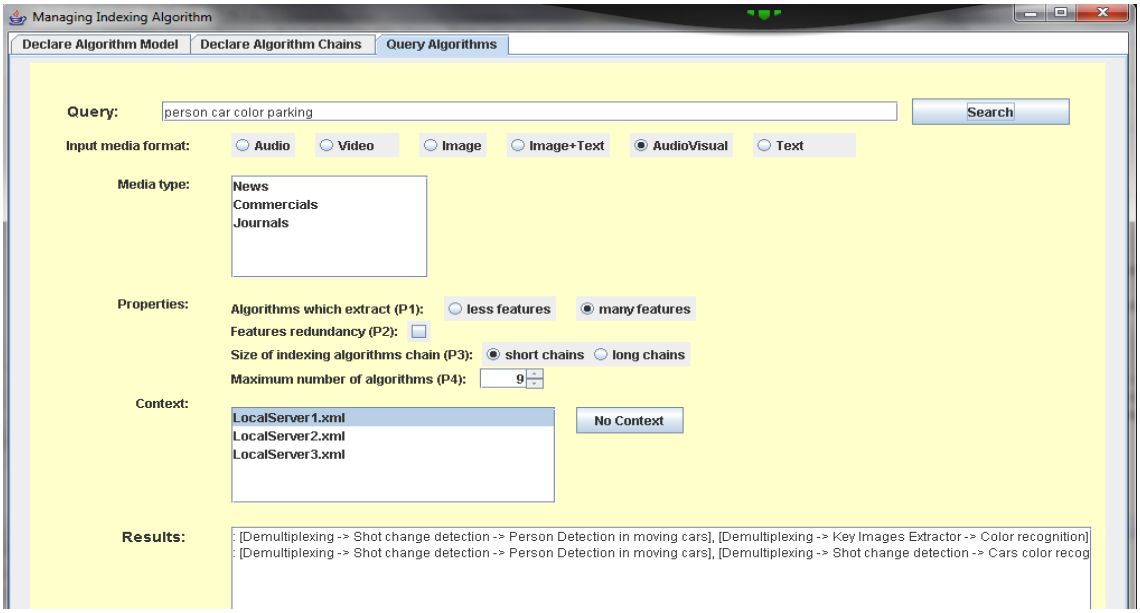


FIGURE 6.6 – Les résultats après avoir choisit un contexte et un autre type de média en entrée

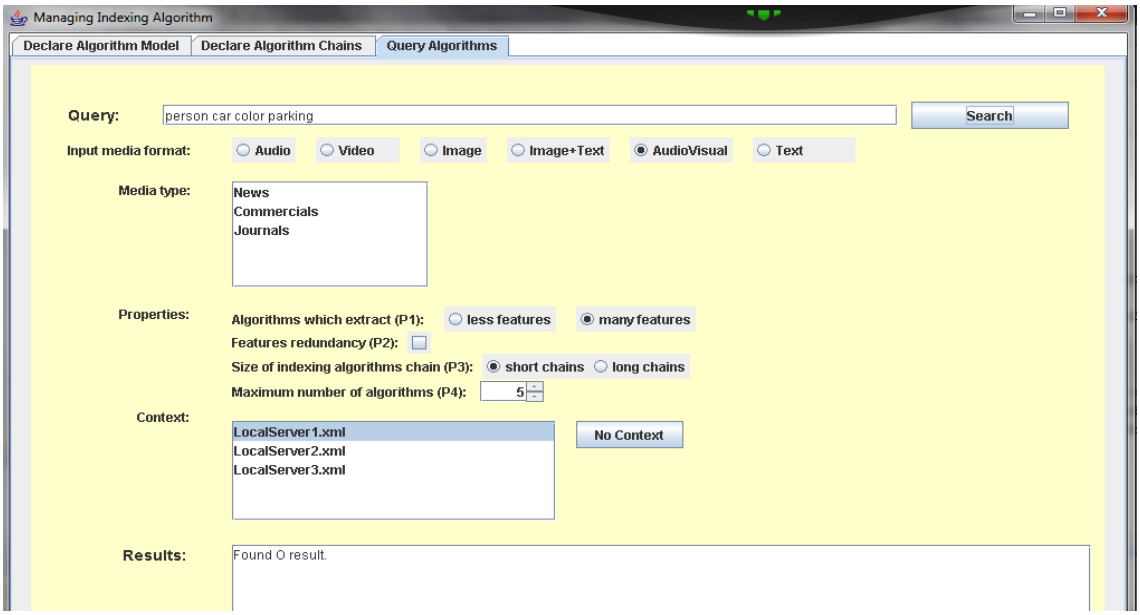


FIGURE 6.7 – Les résultats après avoir choisit un contexte et un autre type de média en entrée