Institut d'Informatique de l'Université de Fribourg (Suisse)

2(CREM) : Une méthode de reconnaissance structurelle de documents complexes basée sur des patterns bidimensionnels

Thèse de doctorat

soumise à la Faculté des Sciences de l'Université de Fribourg (Suisse) pour l'obtention du grade de Doctor Scientiarum Informaticarum

Lyse ROBADEY

de Lessoc (FR)

Thèse n° ?
Mécanographie, Université, Fribourg
2001

Acceptée par la Faculté des Sciences de l'Université de Fribourg, sur la proposition de:

- Professeur Rolf Ingold, Université de Fribourg, Suisse;
- Docteur Frédéric Bapst, Ecole d'Ingénieurs de Fribourg, Suisse;
- Professeur Jean-Marc Ogier, Université de La Rochelle, France.

Fribourg, 21 décembre 2001

Le Directeur de thèse:

Prof. Rolf Ingold

Le Doyen:

Prof. Alexander Von Zelewsky

Remerciements

S'il fallait résumer mon état d'esprit durant cette thèse par un seul mot, je dirais "incertitude". Certaines personnes m'ont beaucoup aidé par leurs collaborations, leurs conseils et leurs encouragements.

- Rolf Ingold a cru en moi et m'a toujours encouragé à poursuivre ma thèse. J'ai particulièrement apprécié la bonne humeur et la gentillesse avec laquelle il a encadré mon travail.
- Du premier au dernier jour de ma thèse, j'ai pu compté sur l'appui de Frédéric Bapst, tant sur le plan de la recherche que sur le plan moral. Le recul impressionnant avec lequel il aborde les domaines les plus divers en fait pour moi un scientifique de premier ordre ... presque un sage (avant l'âge). Du coup je suis très fière qu'il participe à mon jury de thèse.
- Merci à Jean-Marc Ogier pour le travail qu'il a fait en tant qu'expert.
- J'ai eu la grande chance de travailler auprès de Oliver Hitz et de bénéficier ainsi de sa remarquable compétence technique. J'admire spécialement cette attitude d'éveil intelligent qui lui permet d'acquérir jour après jour de nouvelles connaissances, visiblement sans effort. Si je devais faire le décompte de tout ce que j'ai appris grâce à lui, j'en aurais pour la vie des souris ...
- Laurent Karth m'a beaucoup aidé par ses connaissances mathématiques dans la phase de formalisation de ma méthode. De plus, la rédaction de la thèse est une étape particulièrement éprouvante et dans les pires moments de découragement, j'ai pu compté sur sa patience, sa compréhension et sa tendresse.
- Durant presque quatre ans, je me suis faite journellement bombardée de projectiles de toute nature, principalement des gommes. Le terroriste n'était autre que Folco Banfi, mon plus fidèle collègue de bureau. Mais je lui pardonne, car durant toutes ces années, il a toujours été 100% de mon côté.
- Marie-Rose et Félix, mes parents, ont toujours été présents. Ils m'ont soutenu dans les choix relatifs à ma formation, sans jamais essayer de les influencer.
- ... et encore ... Gérald et Richard se sont intéressés à ma recherche et ont su trouver les mots d'encouragement au bon moment, Nicolas et Sergio ont participé au baptême de *2(CREM)* (pas mal pour un Français et un Italien), et patapim et patapom.

Merci à tous !

Table des matières

Abstract

This thesis addresses the question of printed document recognition. We studied existing systems, first in a general context, by making the distinction between physical and logical structure recognition systems. Then, we focused on methods specific for *complex layout documents* and on methods having a *learning aptitude*. Since there do not seem to exist learning systems which are able to recognise complex layout documents, we chose to work in this direction.

First experiments, using simple methods, were applied for the physical structure recognition of newspaper pages. They have revealed the specific problems of the complex layout document analysis, in particular the problem of bidimensional organisation of information.

With this constatation in mind, we conceived *2(CREM)*[1], a general method for object classification that is specially suited for the recognition of objects that form a complex layout document. Indeed, *2(CREM)* has the ability to learn incrementally and enables the description and interpretation of an object in two dimensions; an object is modelled by a *configuration*, i.e. a characteristic set related to the object and to its bidimensional neighbourhood. The configurations are then compared to the analysed document model which includes, for every object class, a set of reference configurations called *patterns*. The system learns the model by *extension* (addition of a pattern to the class description) and by *specialisation* (addition of a characteristic to all the patterns of a class).

2(CREM) was implemented and then tested on four of the steps necessary in newspaper image recognition : the line segment recognition, the frame recognition, the merger of text lines in blocks and the logical labeling of text blocks. We represent the model, the data and the intermediate and final results by using XML. Our experience confirms the choice of XML as a standard for data representation in document recognition. The method was applied on several Los Angeles Times issues. The logical labeling has been tested on 29 pages which were composed of 977 objects in total. After around 150 elementary labeling operations by the user, 86% of the objects were correctely classified, 7% were not identified, 5% were in conflict and 2% were confused with another class. These results show that *2(CREM)* is a relevant learning method for the recognition of documents with complex layout.

Keywords document image analysis – document image segmentation – fonts recognition – document structural recognition – recognition driven by a model – document representation – physical and logical structures – documents with complex layouts – XML technology – document models – model learning – incremental learning

[1] stands for "Configuration REcognition Model for Complex Reverse Engineering Methods" : CREM+CREM = *2(CREM)* and is pronounced "double crème"

Résumé

Cette thèse s'inscrit dans la problématique de la reconnaissance de documents imprimés. Nous avons étudié les systèmes existants, tout d'abord de manière générale en distinguant les systèmes de reconnaissance de la structure physique des systèmes de reconnaissance de la structure logique. Puis, nous nous sommes focalisés sur les méthodes adaptées aux *documents complexes* ainsi que sur les méthodes *douées d'apprentissage*. A la croisée de ces deux axes, nous n'avons rien trouvé et choisi d'apporter notre contribution.

Des premières expériences, faisant appel à des méthodes simples, ont été appliquées pour la reconnaissance de la structure physique des pages de journaux. Elles ont permis de mettre en évidence les problèmes spécifiques à l'analyse de documents à structures complexes, en particulier le problème de l'organisation bidimensionnelle de l'information.

Fort de ces constatations, nous avons conçu *2(CREM)*[2], une méthode générale de classification d'objets s'appliquant particulièrement bien à la reconnaissance des objets qui constituent un document à structure complexe. En effet, *2(CREM)* est douée d'apprentissage incrémental et prévoit la description et l'interprétation d'un objet en tenant compte des deux dimensions; un objet est modélisé en une *configuration*, un ensemble de caractéristiques se rapportant à l'objet et à son voisinage 2D. Les configurations sont alors comparées au modèle du document analysé qui comprend, pour chaque classe d'objets, un ensemble de configurations de référence appelées *patterns*. L'apprentissage du modèle se fait par *extension* (ajout d'un pattern dans la description de la classe) et *spécialisation* (ajout d'une caractéristique à tous les patterns d'une classe).

2(CREM) a été implémentée puis testées dans quatre des phases de la reconnaissance d'images de journaux : la reconnaissance de filets, la reconnaissance de cadres, la fusion des lignes de texte en blocs et l'étiquetage logique des blocs de texte. Le modèle, les données et les résultats intermédiaires et finaux ont été représentés en XML. Nos expériences ont confirmé le choix de XML comme standard de représentation des données dans le domaine de la reconnaissance de documents. La méthode a été appliquée sur plusieurs exemplaires du Los Angeles Times. LŠétiquetage logique a été testé sur 29 pages comprenant en tout 977 objets. Après environ 150 opérations d'étiquetage élémentaire par l'utilisateur, 86% des objets ont été classés correctement, 7% n'ont pas été reconnus, 5% étaient en conflit et 2% ont été confondus avec une autre classe. Ainsi, nous estimons avoir démontré la pertinence de *2(CREM)* comme méthode de reconnaissance de structure complexes de documents qui soit douée d'apprentissage incrémental.

Mots-clés analyse d'images de documents – segmentation d'images de documents – reconnaissance de fontes – reconnaissance structurelle de documents – reconnais-

[2] pour Configuration REcognition Model for Complex Reverse Engineering Methods: CREM+CREM = *2(CREM)* et se prononce "double crème"

sance guidée par un modèle – représentation de documents – structures physiques et logiques – documents à structures complexes – technologie XML – modèles de documents – apprentissage de modèles – apprentissage incrémental

Chapitre 1

Introduction

Pour accéder au sens d'un document écrit, l'humain passe par trois niveaux de perception : la vue, la reconnaissance et la compréhension. De manière immédiate le document est perçu grâce aux organes de la vue. A noter que les documents braille sont une exception, l'écriture en relief permettant de substituer le sens du toucher au sens de la vue. Dans ce travail, l'appellation *document écrit* exclura ce type de document. La *reconnaissance* est l'identification par rapport à un référent. Elle s'applique aussi bien à l'image entière — identification du document comme étant une lettre ou une page de journal — qu'à des portions de l'image — identification d'une zone comme étant du texte ou une illustration. L'accession à ce niveau de perception par un humain dépend de son bagage culturel. Finalement, la *compréhension* consiste à donner du sens au document, à décoder le message que l'auteur a voulu transmettre.

On peut admettre qu'une machine est douée du sens de la vue dans la mesure où elle peut photographier (scanner) un document, le stocker et le retransmettre à volonté par affichage sur un écran ou par impression. La *reconnaissance de documents* cherche à donner à la machine la faculté de reconnaissance, voire de compréhension. Ainsi, un document écrit pourrait être scanné et retransmis sous un format d'édition tel HTML [47] ou LaTeX [42]. Cette thèse a pour objet la reconnaissance de documents *imprimés* à *structure complexe*. Parmi les documents écrits, on distingue les documents *imprimés* des documents manuscrits; la notion *à structure complexe* se réfère à des documents dont la mise en page est particulièrement riche comme par exemple des pages de journaux. De plus, nous voulions un système qui soit doué d'apprentissage : la reconnaissance n'est pas basée sur un ensemble de règles fixes qui décrivent ce qu'est un document, mais sur la connaissance acquise petit à petit au travers d'expériences.

1.1 Reconnaissance d'images de documents

Dans cette section nous allons situer la reconnaissance par rapport à la production de documents, puis nous parlerons des étapes de la reconnaissance, des structures de documents utilisées et des applications de la reconnaissance.

1.1.1 Production et reconnaissance

Un document imprimé est le résultat d'un processus de production en plusieurs étapes. La première étape est la saisie du contenu. Si l'édition est structurée, elle aboutit à la forme logique du document qui contient des éléments textuels ou graphiques auxquels on a associé des étiquettes logiques comme "titre", "liste" ou "tableau". La deuxième étape est une transformation de la forme logique en forme physique appelée *formattage*. La forme physique ne contient plus d'étiquettes : le sens véhiculé par les étiquettes est traduit en attributs typographiques et dans la mise en page. La *restitution* est l'étape suivante : elle transforme la forme physique du document en une image. Finalement le processus peut se terminer par l'*impression* afin d'obtenir un document imprimé.

L'édition peu structurée, comme celle que l'on pratique si l'on utilise le logiciel "Word", ne différencie pas la forme logique de la forme physique. Un document au format "Word" est directement la forme physique du document et la forme image est accessible par le chargement du document avec le logiciel "Word". XML [65], SGML [25] et Thot [52] sont des formats typiques pour la représentation de la forme logique. LaTeX représente aussi la forme logique mais de manière moins rigoureuse puisque l'on peut y spécifier des attributs typographiques.

Le formattage se fait à l'aide d'un outil de transformation. L'outil LaTeX permet de passer d'un fichier au format LaTeX à un fichier au format DVI [39]. D'autres supports de la forme physique sont les formats PostScript [32] ou PDF [33].

1.1.2 Étapes de la reconnaissance

La reconnaissance de documents est le processus inverse de la production. De la forme papier, elle essaie de remonter à la forme logique. La figure 1.1 illustre les deux processus.

La figure 1.2 illustre de manière plus détaillée les étapes de la reconnaissance de documents.

Le document papier est saisi à l'aide d'un scanner de manière à obtenir une image sous la forme électronique, c'est-à-dire une matrice de pixels avec des méta-informations renseignant sur l'interprétation des pixels (couleur, résolution).

L'image électronique obtenue par scannage est une image partiellement bruitée et

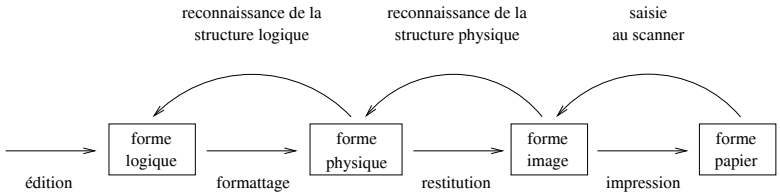


Figure 1.1: Processus de production et de reconnaissance de documents.

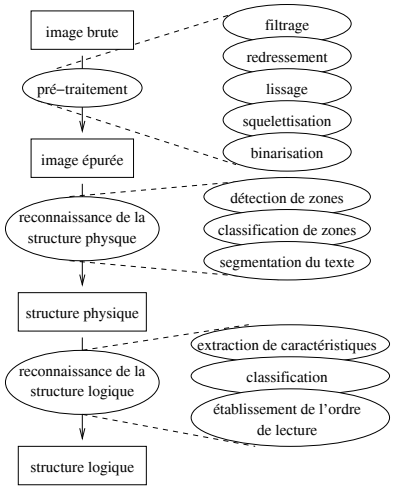


Figure 1.2: Étapes de la reconnaissance de documents.

biaisée appelée *image brute*. Le bruit peut provenir de distorsions ou de poussières accumulées à divers endroits de l'appareil et le biais est dû à une mauvaise position du document papier. Le *pré-traitement* consiste en une série d'opérations dont le but est la correction des imperfections et la préparation aux traitements futurs : on applique successivement des opérations de filtrage, de redressement, de lissage, de squelettisation ou de binarisation. Dans la littérature, on trouve de nombreuses descriptions de cette étape [16] [37] [60].

Le pré-traitement fournit une *image épurée*. Une image idéale est l'image obtenue par restitution lors de la production du document. L'objectif du pré-traitement est de se rapprocher le plus possible de l'image idéale pour faciliter les traitements futurs.

La *reconnaissance de la structure physique* (ou forme physique) consiste d'une part en la détection et la classification des différentes zones de l'image en texte, graphique, table, formule, dessin ou photo et d'autre part en la découpe du texte en colonnes, paragraphes, lignes, mots et signes. A chaque objet de la structure physique est associé un ensemble d'attributs qui décrit l'apparence de l'objet (taille, fonte ou

position).

Finalement, la *reconnaissance de la structure logique* (ou forme logique) consiste à associer des étiquettes logiques aux différents objets de la structure physique et à réorganiser ces objets conformément au flux de lecture. L'étiquetage logique se fait en fonction des attributs des objets physiques. Dans certaines approches, le recouvrement de l'ordre de lecture précède l'étiquetage logique.

1.1.3 Structures de documents

La figure 1.1 montre les trois formes électroniques sous lesquelles un document peut être représenté au cours de sa reconnaissance. La *forme image* du document peut être représentée par n'importe quel format image tel que GIF ou JPEG. Le format TIFF (Tagged Image File Format - [18]) avec une compression selon la norme CCITT groupe 3 ou 4 est particulièrement bien adapté aux documents écrits puisque la compression se fait sans perte. Nous nous intéressons plus particulièrement aux formes physiques et logiques car ce sont les structures visées par la reconnaissance.

La structure physique d'un document décrit l'apparence du document sans faire d'interprétation sur la sémantique de ses objets. Elle est parfois décrite par un arbre pour transcrire les liens hiérarchiques visibles qui existent entre les objets (ex. : un mot fait partie d'une ligne). De plus, chaque objet est décrit par un ensemble d'attributs tels que sa taille, sa fonte ou sa position. Des formats comme PostScript, PDF ou DVI ne sont pas prévus pour exprimer les résultats de la reconnaissance contrairement au format DAFS (Document Attribute Format Specification - [61]). Développé par RAF Technology, ce dernier format définit un type abstrait sous forme de librairie C et spécifie un format de fichier. Dans notre thèse nous avons inauguré l'utilisation d'un moyen plus ouvert de décrire la structure physique avec la norme XML qui permet de spécifier n'importe quel format désiré. L'explosion récente du nombre d'outils (librairies Java, éditeurs ou navigateurs) développés en rapport avec XML rend ce langage particulièrement attractif. La figure 1.4 met en parallèle la représentation de la structure physique du document de la figure 1.3 sous forme d'arbre avec notre proposition de représentation sous forme XML.

La structure logique d'un document décrit son contenu sémantique. Elle indique quel est le rôle de chaque objet dans l'expression du message véhiculé par le document. Elle spécifie par exemple qu'un objet est un *titre* ou un *résumé*. Comme pour la structure physique, la structure logique peut être représentée par un arbre et encodée en XML. La figure 1.5 met en parallèle la représentation de la structure logique du document de la figure 1.3 sous forme d'arbre avec sa représentation sous forme XML.

Deux documents différents peuvent avoir des structures plus ou moins ressemblantes. Si leurs structures comportent les mêmes étiquettes organisées hiérarchiquement de manière similaire, on dira que les deux documents appartiennent à la même classe. Une classe de documents physiques est décrite par une structure physique

Generic Texture Analysis Applied to Newspaper Segmentation

Paul Stefan Williams and Mike D. Alder
Centre for Intelligent Information Processing Systems
Department of Electrical & Electronic Engineering
The University of Western Australia
Nedlands W.A. 6009 AUSTRALIA
phone: +61-9-380-3656 fax: +61-9-380-1168
williams@ciips.ee.uwa.edu.au

ABSTRACT

This paper deals with the segmentation of greyscale newspaper images into the distinct regions of text, picture and background. Feature vectors are obtained from the image by analysing localised textural characteristics and textural variation. Analysis is performed in a generic way making few contextual assumptions. The contrast, size, orientation and resolution of the image are accounted for by a combination of this feature extraction process and subsequent parameterisations.

Given such implicit invariances we are able to perform an initial pixel based classification through the use of a modified quadratic neural network. The use of a simple flood fill based algorithm allows the successful segmentation of newspaper images into distinct rectangular regions. Results of newspaper segmentation show the effectiveness of these methods. Future work will be based on the local processing performed here and its suggested similarity to the behaviour of neurons in the human visual system.

Keywords: Syntactic Pattern Recognition, Quadratic Neural Networks, Feature Extraction, Image Processing, Image Recognition.

1. Introduction

This work deals with the general problem of texture based segmentation for grey scale images. We seek an automatic method for segmenting images into spatially disjoint homogeneous regions with little training. Newspaper images are suitable for this problem and appropriate for illustrating techniques and results.

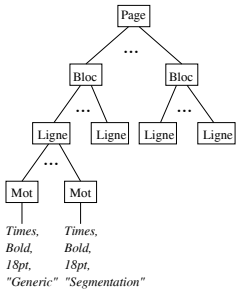
Structured document analysis methods [3] typically assume documents to consist of text, equations, diagrams and tables with a common orientation. They use preprocessing techniques, such as skew correction, to rotate the image appropriately, followed by OCR systems to provide a more thorough decomposition. In newspaper images pictures and objects not conforming to this global orientation inhibit these techniques and pose problems during later processing.

Methods presented maintain generality to texture based segmentation, making few assumptions about the contents of the image, or in this case the proportion of a newspaper's background, picture and text regions. Finally, a robust system is desired with insensitivity to noise, image size, orientation, resolution and intensity.

2. Method used to Obtain Feature Vectors

For newspaper segmentation it is reasonable to choose local descriptions identifying texture like properties at some scale. We use a block of Pattern windows to obtain a representation of a neighborhood of pixels as a point in a higher dimensional feature space. Assuming an effective local representation, large or sizeable homogeneous objects will form separable clusters in this space. Furthermore edges between regions, which are high in contextual information, will be anomalous to these clusters and hence detectable. The following method possesses such desired characteristics. It is described more completely and more generally in [4] and [3] respectively.

Figure 1.3: Image d'une page de document.



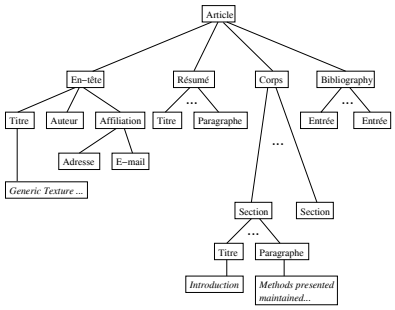
a) arbre d'une structure physique

```
<page>
  <bloc>
    <ligne>
      <mot fonte="Times" graisse="Bold"
           taille="18" contenu="Generic"/>
      ...
      <mot fonte="Times" graisse="Bold"
           taille="18" contenu="Segmentation"/>
    </ligne>
    ...
  </bloc>
  ...
</page>
```

b) structure physique sous forme XML

Figure 1.4: Structures physiques.

appelée *générique*; de même une classe de documents logiques est décrite par une structure logique générique. Les structures qui décrivent une instance de ces classes (un document particulier), sont appelées *spécifiques*. Les structures des figures 1.4 et 1.5 sont des structures spécifiques. La structure générique physique ou logique d'une classe de document est aussi appelée *modèle* physique ou logique de cette classe. En XML, les structures génériques sont décrites par des DTD (Document Type Definition). La figure 1.6 contient des extraits des DTDs des structures physique et logique du document de la figure 1.3. La description XML de la figure 1.4 est conforme à la DTD de la figure 1.6a) et la description XML de la figure 1.5 est conforme à la DTD de la figure 1.6b).



a) arbre d'une structure logique

```
<article>
  <en-tête>
    <titre contenu="Generic Texture ..."/>
    <auteur> ... </auteur>
    <affiliation> ... </affiliation>
  </en-tête>
  <résumé> ... </résumé>
  <corps>
    <section>
      <titre contenu="Introduction"/>
        ...
      <paragraphe contenu="Methods
                  presented maintained ..."/>
    </section>
      ...
  </corps>
  <bibliographie>... </bibliographie>
</article>
```

b) structure logique sous forme XML

Figure 1.5: Structures logiques.

```
<!ELEMENT page  (bloc)*>
<!ELEMENT bloc  (ligne)*>
<!ELEMENT ligne (mot)*>
<!ELEMENT mot   EMPTY>
<!ATTLIST mot fonte   NMTOKEN #REQUIRED>
<!ATTLIST mot taille  NMTOKEN #REQUIRED>
<!ATTLIST mot contenu NMTOKEN #REQUIRED>
```

a) structure physique générique

```
<!ELEMENT article (en-tête, résumé, corps,
                   bibliographie)>
<!ELEMENT en-tête (titre,auteur,
                   affiliation)>
<!ELEMENT titre EMPTY>
<!ATTLIST titre contenu NMTOKEN #REQUIRED>
          ...
<!ELEMENT résumé EMPTY>
<!ATTLIST résumé contenu NMTOKEN #REQUIRED>
<!ELEMENT corps (section)*>
<!ELEMENT section (titre,paragraphe*)>
          ...
```

b) structure logique générique

Figure 1.6: Structures génériques sous forme de DTD.

1.1.4 Applications

La reconnaissance de documents peut s'avérer utile dans trois types de situation : la récupération au format électronique des documents papier produits avant l'ère informatique, la récupération de documents papier produits pendant l'ère informatique et la récupération de documents électroniques.

L'intérêt de pouvoir convertir des archives papier au format électronique est indéniable puisque ainsi ces documents pourraient être accessibles à tout moment et immédiatement grâce à leur diffusion à travers Internet. Plus besoin de se rendre dans une bibliothèque et surtout, un document peut être consulté par plusieurs personnes simultanément. Des documents au format électronique ont en plus le grand avantage de pouvoir être indexés, et d'autant plus facilement si l'on dispose de la forme logique du document.

Mais pourquoi faire de la reconnaissance lorsqu'on dispose de la forme électronique? Une part non négligeable des documents produits de manière informatique reste

destinée à l'impression. Il arrive fréquemment que l'on ne dispose que de la version papier car la version électronique n'est pas en notre possession ou n'a pas été conservée.

Finalement, il est même parfois utile de faire de la reconnaissance directement sur des documents électroniques. Il est excessivement rare de disposer de la forme logique d'un document parce qu'elle n'a pas été conservée ou surtout parce qu'elle n'a jamais existé. Il est pourtant infiniment plus facile et efficace de faire des recherches à partir de la forme logique. Le classement par auteur, date, titre ou thème devient alors un jeu d'enfant. Appliquer la reconnaissance à la forme image de tels documents permet de retrouver la forme logique.

On trouve des applications concrètes de la reconnaissance de documents dans les domaines du tri postal, de la bureautique, du traitement des formulaires ou de l'archivage.

L'automation du tri postal porte principalement sur l'acheminement automatique du courrier grâce au déchiffrage des adresses : elle s'attaque à la reconnaissance des numéros postaux, des noms ou des numéros de rue. Une des grandes difficultés du domaine est la reconnaissance de l'écriture manuscrite. Les articles suivants décrivent des applications de ce domaine [45, 46, 68].

En bureautique, on rencontre le problème de la gestion des divers formats qui coexistent. Bien que le courant souffle dans le sens de la standardisation, on n'en est pas encore au format unique. Des filtres permettent le passage d'un format à l'autre, mais cette solution est loin d'être idéale puisque elle suppose un filtre pour tous les couples de format et si un format évolue, les filtres risquent de devenir obsolètes. Bapst [4] propose une approche plus générale basée sur deux idées : d'une part l'image est considérée comme un *format pivot* qui peut facilement être généré depuis n'importe quel format et d'autre part les techniques d'analyse d'images de documents aident à convertir vers la structure désirée l'information véhiculée par l'image.

Les formulaires ont la particularité d'avoir une structure très rigide, puisqu'ils sont constitués d'une partie pré-imprimée identique pour tous les formulaires d'une même classe. Une fois la classe du formulaire connue, il est facile d'isoler les zones d'intérêt du formulaire pour leur appliquer un traitement. Héroux [30] et Robadey [55] proposent un système basé sur la classification de formulaires alors que Xingyuan [72] adopte une méthode qui ne nécessite pas de classification préalable. De telles applications permettent d'éviter les tâches fastidieuses et lourdes de l'encodage manuel.

Nous sommes particulièrement intéressés par le domaine de l'archivage de journaux puisqu'il concerne des documents à structure complexe. De nombreux quotidiens ont maintenant un site Web sur lequel est disponible un extrait des derniers numéros. D'autres ne fournissent que la version PDF de leur journal. Dans le premier cas, on n'a accès qu'à une sélection du quotidien, alors que dans le deuxième cas, on

n'a une information non structurée et non indexée. A partir d'un PDF, un outil de reconnaissance peut retrouver sans frais la forme logique du document et permettre ainsi l'indexation de l'information. Soulevons toutefois que cet accès facilité à une foule d'informations serait sans doute limité par le problème des droits d'auteur. Avec les outils de reconnaissance, l'encryptage de l'information dans des documents PDF ou Postscript ne protège plus contre le plaggiat.

L'accès à d'anciennes parutions (produites avant l'ère informatique) n'est offert que par les bibliothèques. Si on peut relativement facilement y effectuer des recherches par date, la recherche par sujet est un travail extrêmement lourd. Un service de salle de lecture virtuelle disponible sur Internet serait précieux. Il permettrait non seulement l'accès aux articles à domicile, mais offrirait aussi, grâce aux technologies d'indexation, de puissants outils de recherche thématique. Malheureusement, une telle opération suppose une infrastructure considérable pour la digitalisation des documents. On pourrait imaginer solliciter l'aide des lecteurs : la consultation d'archives serait soumise au scannage de l'information recherchée. Ainsi, toute page de journal consultée au moins une fois entrerait gratuitement dans le service de salle de lecture virtuelle.

1.2 Reconnaissance de documents textuels à structure complexe

Par documents textuels nous désignons les documents dont l'essentiel de l'information est sous forme de texte structuré organisé en phrases et en paragraphes. Nous excluons donc les documents de type affiche, cartes ou tableaux.

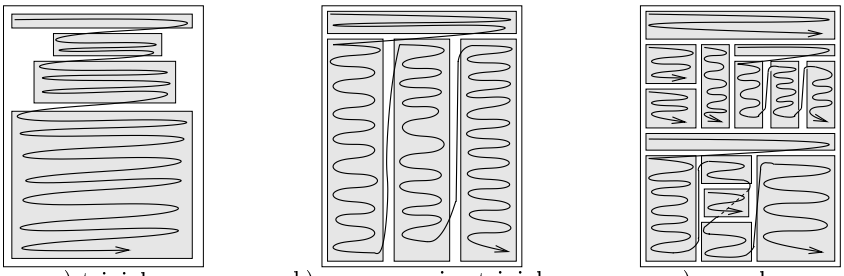
S'il est clair que certains documents ont une structure plus complexe que d'autres, il n'existe pas de critères universels pour décider si un document appartient à la classe complexe ou non. La comparaison des deux premières images de document de la figure 1.7 nous amène naturellement à classer le document a) dans les documents simples et le document b) dans les documents complexes. Par contre, la classification du document c) se révèle moins évidente.

On remarque que ce qui distingue le deuxième document du premier document est le nombre de colonnes de texte, la présence d'illustrations ou la diversité des types et tailles de fontes utilisés. D'autres critères permettent de distinguer les documents a) et c) du document b), comme la variabilité de la largeur des colonnes de texte, le nombre et la diversité des illustrations ou la manière dont les illustrations sont intégrées au contenu textuel. Tous ces critères permettent de qualifier le degré de complexité de la structure d'un document, mais ils ne mettent pas le doigt sur une différence fondamentale en matière de reconnaissance. Une autre différence concerne l'ordre de lecture. Cette différence dépend des autres critères : on peut distinguer 3

a) structure simple b) structure complexe c) structure simple ou structure complexe?

Figure 1.7: Images de documents dont la structure a des niveaux de complexité différents.

niveaux de complexité dans les flux de lecture illustrés par la figure 1.8.



a) trivial b) un peu moins trivial c) complexe

Figure 1.8: Niveaux de complexité du flux de lecture.

La première illustration représente le flux de lecture d'un document dont le texte est réparti sur une seule colonne : l'ordre de lecture y est trivialement de gauche à droite et de haut en bas. Sur la deuxième illustration, l'information est répartie en colonnes de taille égale qui occupent toute la hauteur du document si l'on exclut le haut du document où se trouve l'en-tête et le titre : il existe un ordre implicite entre les colonnes (de gauche à droite) et à l'intérieur d'une colonne, le flux de lecture va de gauche à droite et de haut en bas. Finalement, sur la dernière illustration les colonnes ont des tailles et des positions beaucoup plus chaotiques : le flux de lecture y est non continu et relativement imprévisible. Dans la suite de ce travail, lorsque nous parlerons de documents à structure complexe, nous ferons référence à des documents dont le flux de lecture est non trivial, semblable à celui du document de la troisième illustration.

En ce qui concerne la reconnaissance, la différence réside dans le fait qu'une fois la segmentation d'un document à structure simple effectuée, l'ordre de lecture peut être établi trivialement. Ce n'est pas le cas pour les documents à structure complexe. Le recouvrement de l'ordre de lecture demande une analyse plus poussée qui peut même être traitée dans l'étape de la reconnaissance de la structure logique. L'ordre de lecture établi, le document est linéarisé et l'analyse se poursuit dans un monde à une dimension : un objet a du sens par rapport à l'objet qui le précède et par rapport à l'objet qui le suit. Dans le cas des documents à structure complexe, l'analyse se poursuit dans un monde à deux dimensions : le problème est donc fondamentalement différent. La structure physique d'un document complexe ne peut être entièrement représentée sous la forme d'un arbre. On arrive à retrouver les relations hiérarchiques entre objets (par exemple la relation qui existe entre un mot et la ligne à laquelle il appartient), mais pas les relations d'ordre entre objets de même niveau. Comme le montre la figure 1.9, la structure physique d'un document peut être représentée par un graphe qui a une structure d'arbre sous-jacente.

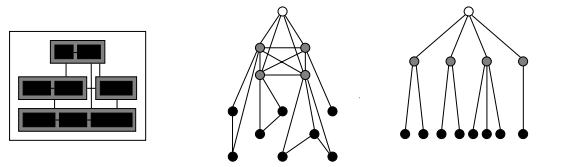


Figure 1.9: Structure physique d'un document complexe.

Dans les documents à structure complexe tels que nous les avons définis, on trouve les journaux (du type quotidien), les revues, les catalogues ou les prospectus. Les livres, les articles scientifiques ou les lettres font partie des documents à structure simple.

1.3 Systèmes doués d'apprentissage

Les systèmes de reconnaissance de documents peuvent être plus ou moins généraux. Certains sont destinés à la reconnaissance de documents bien particuliers, comme par exemple les enveloppes postales, d'autres sont destinés à un spectre plus large de documents, à l'extrême tous les documents. Les systèmes spécialisés dans la reconnaissance d'un type de document particulier sont évidemment plus performants et faciles à réaliser. Mais l'avantage de systèmes plus généraux est indéniable : non seulement ils ont un plus grand domaine d'applications, mais aussi ils ne deviennent pas obsolètes au moindre changement dans le format des documents analysés.

Les systèmes paramétrés par un modèle de documents essaient de profiter des avan-

tages respectifs des systèmes spécialisés et généraux en séparant la partie analyse d'une information spécifique à chaque type de document appelée *modèle*. Les modèles sont des données qui décrivent une classe de documents et qui sont interprétées par la partie analyse. Le système peut choisir le modèle en fonction du type de document analysé. La figure 1.10 montre l'interaction de la partie analyse avec le modèle dans un système de reconnaissance de documents.

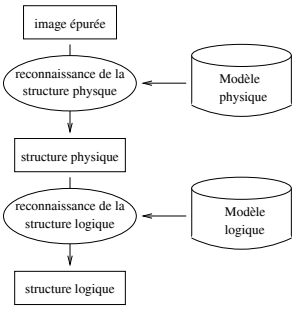


Figure 1.10: Étapes de la reconnaissance de documents dans un système qui utilise des modèles.

Si avec une telle approche on n'a plus besoin de réaliser un système complet pour chaque application visée, il reste le problème de la constitution des modèles. On peut envisager différentes solutions :

- Le modèle peut être saisi manuellement par un spécialiste de la reconnaissance. Cette solution est de loin la moins bonne car la tâche est longue et difficile. De plus, il est probable que le modèle ainsi créé ne prenne pas en compte tous les cas particuliers.
- Certains systèmes prévoient un module qui déduit automatiquement le modèle d'un échantillon de documents. Ce sont les *systèmes doués d'apprentissage*. La qualité du modèle dépendra du choix de l'échantillon d'apprentissage. Si l'échantillon n'est pas représentatif ou si les documents évoluent, il faudra créer un nouveau modèle.
- Finalement, il y a des systèmes doués d'*apprentissage incrémental* où le modèle évolue en cours d'utilisation du système. Un tel système n'est pas entièrement automatique, mais assisté par un opérateur. Le modèle évolue en fonction des corrections faites par l'opérateur.

La figure 1.11 illustre un scénario d'interaction entre un opérateur et un système doué d'apprentissage incrémental.

La charge d'assistance peut paraître plus lourde avec un système d'apprentissage incrémental, mais il ne faut pas s'y fier : les résultats produits par des systèmes entièrement automatiques doivent être corrigés. En effet, de tels systèmes n'atteignent jamais un taux de reconnaissance de 100%. La tâche de correction est souvent

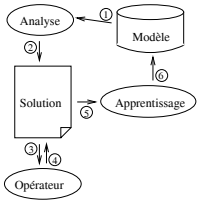


Figure 1.11: Scénario d'interaction entre un opérateur et un système doué d'apprentissage incrémental : 1) le système analyse le document en utilisant le modèle, 2) le système propose une solution à l'utilisateur, 3) l'opérateur prend connaissance de la solution, 4) l'opérateur corrige la solution, 5) le module d'apprentissage interprète les corrections faites par l'opérateur et 6) le module d'apprentissage corrige le modèle.

ennuyeuse car les systèmes commettent toujours les mêmes erreurs. Une approche assistée évite au système de commettre deux fois la même erreur : le comportement du système est modifié sur la base des erreurs détectées et de manière à ce qu'il ne les commette plus. Le temps et l'intérêt du travail réalisé par l'opérateur dépendra fortement du mode de communication homme-machine prévu. On privilégiera les approches où l'opérateur a l'initiative du traitement aux approches où c'est le système qui conduit la reconnaissance.

1.4 Objectifs de cette thèse

L'objectif de cette thèse est la conception, la mise au point et l'évaluation d'une méthode générale de reconnaissance qui 1) s'applique à des documents à structure complexe et 2) soit douée d'apprentissage incrémental. La méthode est générale dans le sens où elle peut être utilisée pour faire de la reconnaissance aussi bien de la structure physique que de la structure logique.

1.4.1 Choix en accord avec la philosophie *CIDRE*

Depuis 1994, les travaux de notre groupe de recherche sont orientés par le projet *CIDRE*[1] [6, 7, 8, 5, 9, 14, 15, 26, 27, 28, 29, 57] (pour Cooperative & Interactive Document Reverse Engineering). Ce projet est fondé sur une révision de toute la problématique en reconnaissance de documents qui s'organise selon les quatre axes suivants :

Reconnaissance assistée. L'idée est de minimiser la main d'oeuvre impliquée dans le processus de reconnaissance en autorisant l'utilisateur à exercer une influence *durant* la session de reconnaissance et non pas exclusivement dans

[1] projet soutenu par le Fonds National de la Recherche Scientifique, subside no 21-42'355.94.

une phase initiale de configuration et une phase ultérieure de correction des résultats. L'utilisateur peut ainsi modifier le modèle de document en cours d'emploi et éviter la tâche ennuyeuse de corrections des erreurs systématiques.

Réingénierie de documents au sens large. Les documents visés ne sont pas spécifiés : les plate-formes et les outils de reconnaissance développés dans le cadre du projet *CIDRE* devraient être capables de s'adapter à toute classe de document (journaux, articles scientifiques ou formulaires) et à n'importe quel format (PDF, Postscript ou papier).

Rôle de l'architecture logicielle. *CIDRE* préconise une architecture logicielle qui encourage la coopération homme-machine et la coopération entre diverses sources de connaissance : plutôt qu'un système complet de reconnaissance, on préfère de petits outils indépendants que l'on peut facilement combiner et faire collaborer en fonction de l'application visée.

Modèles de documents. Les outils développés dans le cadre de *CIDRE* devraient prévoir la possibilité de créer les modèles de documents de manière incrémentale durant la session de reconnaissance interactive.

Ces axes de recherche sont plus largement décrits et motivés dans la thèse de Bapst [4]. Tout au long de notre travail, nous avons essayé de rester fidèle à ces principes directeurs.

1.4.2 Documents à structure complexe

L'analyse de documents à structure complexe diffère de l'analyse de documents à structure simple surtout pour la reconnaissance de la structure logique et la partie fusion des lignes de texte en blocs de la reconnaissance de la structure physique. Nous nous sommes donc tout particulièrement intéressés à ces étapes de la reconnaissance. Une étude a été menée dans notre groupe de recherche par Rolf Brugger [14] : elle s'intéresse à la reconnaissance de la structure logique. Elle a abouti à une méthode de reconnaissance basée sur les n-grams généralisés qui a été testée sur la documentation du projet Linux disponible sur Internet. Comme ces documents ont une structure relativement simple, nous nous sommes demandés dans quelle mesure elle était applicable à des documents à structure complexe du type page de journal. Notre démarche est la suivante. Dans un premier temps, nous utilisons des méthodes simples pour faire de la segmentation de documents complexes. Cette étape permet d'une part de préparer les données nécessaires à la reconnaissance de la structure logique et d'autre part d'étudier qu'elle est la spécificité de l'analyse des structures complexes. Nous en avons déduit que le principal défi est la prise en compte d'une organisation de l'information dans la deuxième dimension. La méthode de Brugger n'est donc pas applicable telle quelle à des documents complexes, c'est pourquoi dans un deuxième temps, nous avons développé une méthode qui relève le défi.

1.4.3 Apprentissage incrémental

En accord avec la philosophie *CIDRE*, nous voulons une méthode qui donne un rôle prépondérant à l'utilisateur. Nous nous attachons donc à imaginer une architecture logicielle qui le permette. Notre étude ne comprend pas la mise au point d'une interface graphique qui facilite la communication : cet axe de recherche est exploré par Oliver Hitz, autre collaborateur de notre groupe. L'approche sera donc complètement indépendante de l'interface graphique.

Comme le préconise *CIDRE*, le modèle de document sera construit de manière interactive et incrémentale au cours de la reconnaissance. Aucune connaissance sur le type de documents traités ne fera partie de la méthode. Ainsi nous espérons que le système pourra s'adapter à toute sorte de documents sans que les sources soient modifiées ou sans même que des paramètres soient ajustés.

1.5 Organisation en chapitres

La thèse est organisée en six chapitres.

Le chapitre 2 fait le point sur l'*état de l'art* en reconnaissance de documents. Il contient deux sections générales qui traitent de la reconnaissance de la structure physique, respectivement logique. Deux sections sont plus ciblées sur les principaux aspects de notre recherche et présentent des études sur la reconnaissance de documents à structure complexe et sur les systèmes doués d'apprentissage.

Notre étude de la reconnaissance de documents à structure complexe s'est faite en deux temps. La première partie a consisté à développer un *système de reconnaissance de la structure physique basé sur des méthodes simples et traditionnelles*. Le chapitre 3 décrit le système et discute des problèmes spécifiques aux documents complexes qui ne peuvent être résolus avec des méthodes simples.

La suite de notre recherche a été la mise au point d'une méthode appelée *2(CREM)* [2] qui soit adaptée à la reconnaissance de documents à structure complexe et ne présente donc pas les faiblesses des méthodes simples. Le chapitre 4 est consacré à 2(CREM), *une méthode de reconnaissance structurelle basée sur des patterns*.

Le chapitre 5 présente l'*application de* 2(CREM) *à la reconnaissance d'images de journaux*. Il décrit 4 applications : la reconnaissance des filets, la reconnaissance des blocs, la fusion des lignes de texte en blocs et l'étiquetage logique des blocs. Il décrit aussi les outils utilisés pour l'extraction des caractéristiques nécessaires à ces applications et présente les résultats de tests effectués pour chacune des applications.

Enfin, le chapitre 6 expose les *conclusions de la thèse*. Nous énumérons les contribu-

[2] pour Configuration REcognition Model for Complex Reverse Engineering Methods: CREM+CREM = *2(CREM)* et se prononce "double crème".

tions scientifiques apportées par notre étude, ainsi que des extensions qui pourraient être approfondies dans des travaux ultérieurs.

Chapitre 2

État de l'art

Au début du XXème siècle déjà on faisait de la recherche en reconnaissance d'images de documents. En 1914 on présente des inventions pour remplacer les opérateurs des télégraphes et pour assister les aveugles : c'était les premiers OCR [48]. L'histoire de la reconnaissance des structures de documents est par contre beaucoup plus récente. Les premiers systèmes ont été développés pour des tâches bien ciblées comme le tri automatique du courrier (reconnaissance d'adresses) ou la reconnaissance de chèques postaux. Ce n'est que récemment que les systèmes ont intégré des modèles de documents et sont ainsi devenus plus flexibles et même doués d'apprentissage.

Dans cette section nous faisons d'abord un survol des techniques développées pour la reconnaissance des structures physiques et logiques, puis nous présentons des systèmes s'attaquant à la reconnaissance de documents à structure complexe et finalement nous parlons de systèmes doués d'apprentissage.

2.1 Reconnaissance de la structure physique

Parmi les méthodes de segmentation, on trouve des méthodes ascendantes, descendantes et mixtes. Les méthodes ascendantes procèdent par regroupement d'éléments en partant des composantes connexes. Dans une première étape, les pixels de l'image de départ ou d'une image transformée (par un filtre RLSA [71] par exemple), sont regroupés en composantes connexes. Une deuxième étape consiste à extraire des caractéristiques sur ces composantes afin de pouvoir les regrouper en zones homogènes. Dans les articles [23], [41], [21] et [20], de telles méthodes sont présentées. La technique "docstrum" proposée par O'Gorman [51] procède aussi de manière ascendante. Les composantes connexes ne sont pas regroupées en fonction de l'homogénéité de leurs propres caractéristiques, mais en fonction de caractéristiques sur les relations entre des paires de composantes voisines : recherche des *k plus proches voisins* des composantes, puis analyse de l'angle et de la distance séparant chaque paire de voisins.

Les méthodes descendantes partent de l'image entière et cherchent à la décomposer

récursivement en composantes de plus bas niveaux. Beaucoup de ces méthodes analysent le fond de l'image (zones blanches). Chez Gatos [22] et Antonacopoulos [2], les différentes zones d'intérêts sont regroupées grâce à l'analyse des zones blanches de l'image et extraites respectivement par une segmentation en composantes connexes et par une technique de suivi de contour. Krishnamoorthy [40] et Wang [69] appliquent la découpe en arbre *X-Y*, appelée aussi découpe récursive en utilisant le profil de projection. Dans ces deux derniers cas, la découpe est guidée par une connaissance a priori de la structure du document analysé. Cinque [17] propose plusieurs rééchantillonages de l'image qui ont pour effet la réduction de l'image et la mise en évidence de différentes zones. Dans chaque réduction, des fenêtres 16 x 16 pixels de l'image de départ sont remplacées par une caractéristique extraite sur cette fenêtre. L'extraction entre autres caractéristiques de la moyenne et de la variance de l'intensité des pixels, produit des images qui, combinées et seuillées, font ressortir le fond de l'image d'origine.

L'étiquetage des zones en texte, image ou graphique est en principe partie intégrante de la segmentation; il est est guidée par la connaissance de la structure générique des différentes zones pouvant constituer l'image (texte, image, graphique). Dans l'approche ascendante, les composantes connexes sont regroupées en fonction de critères spatiaux (seules les composantes voisines sont regroupées) et de critères d'homogénéité qui consistent à déterminer à quel type de bloc une composante appartient. La liste de caractéristiques fréquemment extraites sur les composantes connexes comprend dimensions, coordonnées, rapport hauteur-largeur, aire du rectangle englobant, densité des pixels noirs, longueur moyenne des segments noirs horizontaux, nombre de transitions noir/blanc.

Dans les approches descendantes de Gatos et Antonacopoulos [22, 2], la classification des zones n'est pas une étape nécessaire à la segmentation. La segmentation ne s'effectue que par l'analyse du fond de l'image. Dans une étape ultérieure, Gatos étiquette les zones en texte/non texte: il applique une FFT (Fast Fourier Transform) sur la projection horizontale de la zone afin de détecter les fréquences dominantes. Une zone texte est une zone avec une fréquence nettement dominante. Les approches de Krishnamoorthy [40] et Wang [69] qui utilisent la découpe en arbre *X-Y*, ont besoin de la connaissance du type des zones pour leur segmentation. Alors que Krishnamoorthy travaille sur des grammaires construites à partir du profil de projection horizontal seuillé, Wang extrait des statistiques sur différentes combinaisons de segments ("run length") de la région analysée.

Azokly [3] adopte une approche mixte (descendante et ascendante). Il combine un algorithme de découpe hiérarchique basée sur l'analyse de *rectangles structurants* (rectangles blancs qui constituent le fond de l'image) avec un algorithme de fusion de composants gouverné par des règles décrivant les structures à reconnaître.

L'approche descendante utilise souvent un modèle du format du document (sorte de "feuille de style inversée") qui guide le système dans son action de décomposition [11].

L'approche ascendante ne requiert pas de connaissance sur la présentation globale du document, une certaine connaissance sur l'aspect des éléments de base suffit. L'approche descendante s'applique donc bien aux documents dont la feuille de style est connue car elle est plus rapide et efficace. Par contre l'approche ascendante sera plus adaptée pour analyser des documents dont on ne peut prévoir le format, les documents à structure complexe par exemple.

2.2 Reconnaissance de la structure logique

Dans le processus de production de documents, le passage de la structure logique à la structure physique se fait grâce à des règles de présentation du document. Selon le système d'édition, ces règles sont plus ou moins implicites. Un éditeur peu structuré saisira le contenu et la présentation du document de manière non différentiée. L'édition structurée synthétise les règles de présentation sous la forme d'une feuille de style indépendante du contenu du document et applicable à un ensemble de documents.

En reconnaissance de documents, le processus inverse — passage de la structure physique à la structure logique — ne peut se faire sans la prise en compte, là aussi de manière plus ou moins implicite, des règles de présentation. En effet, même si le contenu logique peut être en partie déduit du contenu textuel, il est avant tout exprimé par la présentation du document (fonte ou mise en page). A la différence de la production de documents, la reconnaissance de document guidée par des règles de présentation n'est pas univoque : plusieurs structures différentes peuvent être générées à partir d'une image de documents et d'un ensemble de règles de présentation. Les méthodes dont les règles de présentation font partie du contrôle s'appliquent à un ensemble restreint de documents : les documents produits en respectant ces règles. A l'inverse, les méthodes qui isolent ces règles à l'intérieur d'un modèle sont souvent douées d'apprentissage et par là s'appliquent à une gamme beaucoup plus large de documents. Ces dernières méthodes feront l'objet du point 2.4.

La reconnaissance de la structure logique comprend deux étapes : l'étiquetage des blocs et la transformation de la structure physique en structure logique. L'ordre dans lequel ces étapes sont effectuées dépend des méthodes; elles ont même parfois lieu simultanément. L'étiquetage consiste à attribuer une étiquette logique à un bloc qui donne une indication sur le rôle du bloc dans le document. Parmi les étiquettes les plus courantes on trouve "titre" et "texte de base". Dans la transformation de la structure on va principalement fusionner des blocs physiques appartenant à la même entité logique et déterminer un ordre de lecture entre les entités logiques. La figure 2.1 est une illustration de la transformation de la structure physique. L'étiquetage des blocs, qu'il ait lieu avant, après ou en même temps que la transformation de la structure, se fait en deux étapes successives : l'extraction des caractéristiques et la

classification.

Si l'on fait abstraction de la capacité d'apprentissage, les différents systèmes de reconnaissance logique décrits dans la littérature se distinguent par 1) les caractéristiques extraites, 2) la méthode de classification utilisée et 3) la manière dont la structure est remaniée. De plus, certains systèmes particulièrement souples, permettent la remise en cause de résultats en cours de reconnaissance.

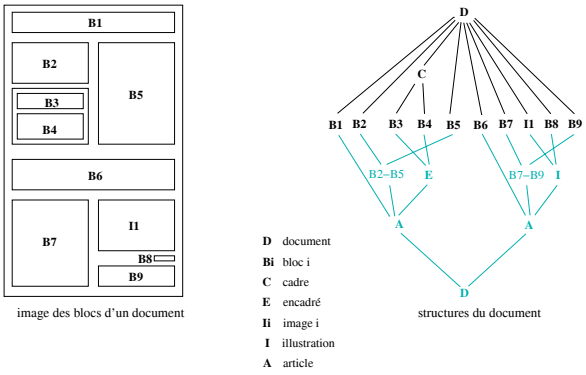


Figure 2.1: Exemple de transformation de la structure physique (en noir) en structure logique (en gris).

2.2.1 Extraction de caractéristiques

La plupart du temps, les caractéristiques sont extraites sur des images binaires du document. Il y a des caractéristiques qui se rapportent aux propriétés intrinsèques du bloc, d'autres à son voisinage ou à sa position dans la page. On peut classer les caractéristiques qui se rapportent directement au bloc en plusieurs catégories.

Les caractéristiques morphologiques. Les caractéristiques les plus fréquentes sont les dimensions du bloc, son élongation (rapport entre la hauteur et la largeur), sa densité (rapport entre le nombre de pixels noirs et le nombre total de pixels). Wang et Srihari [69] analysent les transitions noir-blanc pour chaque ligne de balayage du bloc.

Les caractéristiques structurelles. Elles s'appliquent par exemple aux blocs textuels et décrivent l'organisation des lignes de texte qui composent le bloc par le nombre de lignes, la position relative des lignes, l'interligne et des caractéristiques typographiques telles que la mise en page des lignes (centrées, justifiées, alignées à droite ou à gauche), la fonte dominante, la taille de la fonte dominante ou le style de la fonte dominante (souligné, italique, gras). Les caractéristiques structurelles peuvent également s'appliquer aux formules et aux tableaux.

Le contenu textuel. Il apporte parfois une aide précieuse à la reconnaissance de la structure logique. Ishitani [35] utilise des caractères tels que "2.", "*" ou "1)" comme des indices pour reconnaître une liste et les symboles mathématiques comme des indices pour reconnaître une formule. Klink [38] qui s'occupe de tri postal recherche des mots ou des suites de mots comme "dear Mr".

Certaines caractéristiques ne se rapportent pas directement au bloc, mais à son voisinage. Elles ont, pour la plupart, trait aux positions relatives ou à l'étiquette logique des blocs. A noter que cette dernière caractéristique implique une approche heuristique de la classification car l'étiquette logique est précisément l'information recherchée : nous reparlerons de telles approches plus loin. Klink [38] recherche des liens entre le contenu textuel d'un bloc et de ses voisins : il formule des règles stipulant par exemple que pour se voir assigner une étiquette y, un bloc doit avoir un mot en commun avec un autre bloc. Une telle règle pourrait décrire la relation qui existe entre le champ "destinataire" et "salutations" d'une lettre.

Cinque [17], Spitz [59] et Belaïd [10] utilisent la position absolue pour reconnaître des blocs tels que les en-têtes ou les pieds de page.

Le tableau 2.1 classe différentes approches de reconnaissance de la structure logique en fonction des caractéristiques extraites.

2.2.2 Classification

La classification utilise les caractéristiques extraites sur les blocs pour leur attribuer une étiquette. Les caractéristiques sont interprétées en tenant compte d'une certaine connaissance sur le document analysé. Cette connaissance correspond aux règles de présentation dont nous avons parlé au début du point 2.2; rappelons qu'elle peut être partie intégrante du contrôle ou isolée dans un modèle. La méthode de classification choisie est souvent liée aux types de caractéristiques et à la représentation de la connaissance. Beaucoup de méthodes synthétisent la connaissance sous forme de règles qui font partie du contrôle. Voici un exemple d'une règle utilisée par Cinque [17] : *a Heading has to be composed of at most two rows and must not exceed one quarter of the vertical dimension of the page.*

Les caractéristiques morphologiques extraites sur un bloc sont souvent représentées sous forme de vecteurs. La connaissance consiste en un partitionnement de l'espace formé par les caractéristiques, chaque élément de la partition correspondant à une classe. Parmi les classifieurs classiques on trouve les règles de Bayes ou le k-plus proche voisin. Esposito [19] utilise un réseau de neurones dont les poids et les seuils forment la connaissance sur le type de document analysé.

D'autres utilisent des modèles statistiques. Héroux [24] calcule un vecteur moyen de caractéristiques morphologiques pour chaque classe. La classification consiste à choisir la classe dont le vecteur moyen a une distance minimale au vecteur du bloc

auteur	M	S	P	L-P	T	L-T	L-E
Belaïd [10]	X	X	X	X			
Brugger [13]	X	X		X			X
Cinque [17]	X		X	X			
Esposito [19]	X						
Héroux [24]	X	X				X	
Hu [31]	X		X	X			
Ishitani [35]	X				X		
Klink [38]	X			X	X		X
Lam [41]	X						
Lebourgeois [43]	X			X	X		X
Niyogi [49]	X	X		X			
Spitz [59]				X			
Tsujimoto [62]	X			X			
Walischewski [67]				X			
Wang [69]	X						
Yamaoka [73]	X			X			

M caractéristiques morphologiques
S caractéristiques structurelles
P position absolue
L-P positions relatives
T contenu textuel
L-T lien textuel
L-E lien entre les étiquettes

Tableau 2.1: Classification des méthodes de reconnaissance de la structure logique sur la base des caractéristiques extraites.

à classer. Lebourgeois [43] définit des probabilités d'appartenance à une classe en fonction des caractéristiques observées; la classification est une heuristique appelée "relaxation probabiliste" qui consiste à optimiser une solution globale par des modifications découlant de mesures locales. L'algorithme est appliqué de manière itérative. Quant à Brugger [13], il a imaginé une généralisation du modèle des n-grams de manière à ce qu'on puisse représenter non seulement des structures linéaires, mais aussi des structures hiérarchiques. Il peut donc représenter le modèle d'une classe de documents par ses n-grams généralisés. Là aussi la classification est une heuristique.

Walischewski [67] et Héroux [24] représentent les relations qu'entretiennent un bloc avec son voisinage par un graphe. Héroux modélise une classe par le sous-graphe isomorphe à tous les échantillons d'apprentissage et classe un bloc en tenant compte de la distance qui sépare le graphe du bloc et les graphes des classes. Cette distance

est évaluée à partir du sous-graphe isomorphe aux graphes du bloc et du modèle. Walischewski quant à lui représente tout le document par un graphe. Le modèle est une synthèse des graphes de l'échantillon d'apprentissage. Il regroupe les noeuds de l'ensemble des graphes et leur assigne une probabilité d'apparition. Là aussi la classification consiste en la recherche d'isomorphismes de sous-graphes. Les probabilités d'apparition permettent de classer les différentes alternatives s'il y en a.

Belaïd [10] représente le modèle de la structure logique par une grammaire qui est inférée automatiquement de plusieurs échantillons de la structure physique ainsi que d'un étiquetage logique fourni par l'utilisateur.

La méthode de Hu [31] propose un modèle décrit par une grammaire attribuée hors-contexte. La structure du document logique générique est représentée par des règles de production de la grammaire hors-contexte, alors que l'aspect physique des éléments logiques est représenté par les attributs correspondants. L'incertitude est gérée au moyen de la logique floue, et le processus est guidé par un algorithme de programmation dynamique.

Le tableau 2.2 classe différentes approches de reconnaissance de la structure logique en fonction du type de classification choisi.

2.2.3 Construction de la structure logique à partir de la structure physique

La plupart des systèmes centrent la reconnaissance sur l'étiquetage des blocs et la transformation de la structure n'est qu'une petite étape terminale. Tsujimoto [62] propose un système de reconnaissance basé sur la transformation de la structure. La structure physique est représentée sous la forme d'un arbre et grâce à quatre règles de transformation il obtient l'arbre de la structure logique. D'autres règles permettent par la suite d'attribuer des étiquettes logiques aux différents éléments de l'arbre. Ces règles sont câblées dans le contrôle et par là rendent le système rigide. Pourtant l'approche est originale et élégante. Les règles choisies sont le résultat d'une réflexion poussée sur la structure générique à la plupart des documents et ainsi le système a un degré de généralité étonnant. Brugger lui aussi part de la transformation d'une structure d'arbre, mais son modèle statistique rend le système beaucoup plus souple que celui de Tsujimoto puisqu'il permet l'apprentissage et la remise en cause de résultats [13].

Yamaoka [73] et Niyogi [49] traitent simultanément la transformation de la structure et l'étiquetage logique, mais contrairement au système de Yamaoka, celui de Niyogi permet la remise en cause.

L'ordre dans lequel l'étiquetage logique et la transformation de la structure sont appliqués diffère selon les approches. Cela montre que la connaissance de l'étiquetage logique est utile à la transformation de la structure et vice-versa. C'est pourquoi

auteur	R	P	S	A	I	G	N
Belaïd [10]						X	
Brugger [13]			X	X			
Cinque [17]	X						
Esposito [19]							X
Héroux [24]			X		X		
Hu [31]						X	
Ishitani [35]	X						
Klink [38]	X						
Lam [41]		X					
Lebourgeois [43]			X				
Niyogi [49]	X						
Spitz [59]	X						
Tsujimoto [62]	X			X			
Walischewski [67]					X		
Wang [69]		X					
Yamaoka [73]					X		

R règles
P partition de l'espace
S modèle statistique
A transformation d'arbres
I isomorphisme de sous-graphes
G modèle syntaxique (grammaire)
N réseau de neurones

Tableau 2.2: Classification des méthodes de reconnaissance de la structure logique sur la base du type de classification adopté.

certains systèmes prévoient la remise en cause afin de profiter au mieux de cette double source d'information.

2.2.4 Systèmes prévoyant la remise en cause

Les systèmes particulièrement bien adaptés à la remise en cause sont ceux qui représentent le modèle par des données statistiques puisque l'idée est d'itérer le système jusqu'à obtenir une solution globale qui soit localement acceptable partout. L'acceptabilité d'une solution locale est évaluée grâce aux données statistiques et à un seuil. Brugger [13] conserve plusieurs alternatives dans un arbre de recherche. Seule la meilleure alternative est explorée, mais il se réserve la possibilité d'explorer les autres alternatives en cas d'échec.

D'autres systèmes pratiquent la remise en cause par un contrôle particulièrement évolué. Chez Ishitani [35], chaque étape de reconnaissance est encapsulée dans un module; ces différents modules collaborent, orchestrés par un module supplémentaire appelé "module de modification d'objets". Niyogi [49] quant à lui fait collaborer ses modules grâce à des règles de stratégie et de contrôle isolées dans le modèle de connaissance.

Lebourgeois [43] utilise les étiquettes logiques comme caractéristique. Il pratique une heuristique qui est proche de la remise en cause appelée *relaxation probabiliste*. Cette méthode a été décrite par Rosenfeld [56] pour la reconnaissance des formes. Elle "permet par itérations successives de modifier la classification d'un objet en fonction de la compatibilité locale avec les objets voisins jusqu'à ce qu'une solution globale compatible partout localement soit trouvée."

Le système de Ogier [50] prévoit également la remise en cause. Bien qu'il ait été appliqué à la reconnaissance de plans cadastraux, il peut s'appliquer à tout type de documents dont les composants sont organisés de manière hiérarchique. Le système a d'ailleurs été repris par Héroux [24] et appliqué à des documents écrits. Cette approche s'inspire du système d'interprétation des images chez l'humain qui est un processus cyclique faisant coopérer les modes de perception syncrétique (vision globale) et analytique (vision locale) jusqu'à l'obtention d'une interprétation cohérente de l'image. Dans son système, Ogier analyse la cohérence de l'interprétation d'une image à différents niveaux hiérarchiques (du bas niveau au haut niveau). La cohérence d'un objet dépend de sa cohérence interne (évaluée en fonction de ses composants) et de sa cohérence externe (évaluée en fonction de son voisinage). En cas d'incohérence, des *solutions remèdes* sont proposées. Un cycle est constitué de l'analyse de la cohérence de la solution courante et de la proposition de *solutions remèdes*. Les cycles sont itérés jusqu'à l'obtention d'une interprétation cohérente.

2.3 Reconnaissance de documents à structure complexe

Parmi les approches citées jusqu'ici, toutes ne s'appliquent pas à la reconnaissance de structures complexes. En ce qui concerne le recouvrement de la structure physique, le traitement de structures complexes a stimulé la recherche sur les trois plans suivants : la distinction entre les zones texte, photo et graphique, la détection et la représentation de zones non rectangulaires et finalement l'organisation des différentes zones en une structure pas forcément hiérarchique.

2.3.1 Classification des zones de l'image

Pour classer les zones de l'image, la plupart des méthodes, dont celles de Cinque [17], Fan [20], Lam [41] ou Williams [70] extraient, outre des caractéristiques sur la dimension de la zone, des caractéristiques sur la répartition des pixels de la zone : moyenne, densité et surtout variance. Sivaramakrishnan [58] fait une étude plus fine mais qui ne s'applique que sur des images binaires en étudiant les *runs*[1] de la zone. Il extrait le nombre de *runs*, leur longueur moyenne ainsi que leur variance. Quant à Gatos [22], il compare la FFT des zones à classer.

2.3.2 Détection et représentation de zones non rectangulaires

Il arrive, suivant le type de documents, que les zones ne correspondent plus à des rectangles. Chez Antonacopoulos [2], Belaïd [10] et Williams [70], la zone est un polygone décrit par une liste de points. L'extraction de telles zones est assez élaborée. Antonacopoulos représente le fond de l'image (les pixels blancs en général) par un pavage fait des plus grands rectangles blancs appelés rectangles structurants, c'est-à-dire des rectangles ayant la plus grande surface possible. Ensuite, il construit un graphe dont les noeuds sont les rectangles structurants et les arêtes les relations d'adjacence verticale (en dessus ou en dessous) entre les rectangles. L'étude des cycles du graphe permet de délimiter les zones de l'image. En effet, les rectangles qui forment un cycle entourent une zone. Antonacopoulos obtient ainsi une description très fine de la forme d'une zone. Belaïd a imaginé une méthode proche de celle d'Antonacopoulos : étude des plages blanches de l'image, représentation sous forme de graphe et analyse des cycles du graphe. Williams utilise une technique toute différente : il applique sur l'image des masques composés de k^2 blocs qui contiennent chacun n^2 pixels (cf. illustration 2.2). A chaque masque correspond un vecteur de caractéristiques à $k^2 + 2$ dimensions : pour chaque bloc sa variance (k^2) ainsi que les coordonnées x et y de l'origine du masque (2). Un réseau de neurones permet de classer les vecteurs de caractéristiques en *texte*, *illustration*, *fond*, *limite* et *autre*. Les portions de l'image classées *limite* permettent d'extraire les contours des zones de l'image.

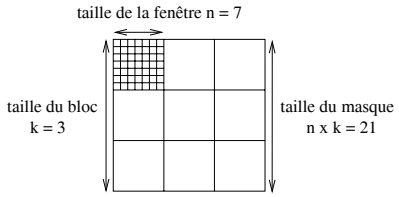


Figure 2.2: Masque utilisé par Williams.

[1] Les lignes de balayage d'une image binaire sont composées d'une alternance de segments noirs et blancs appelés *runs*.

2.3.3 Organisation des zones en structure

L'organisation des composants de l'image en une structure est plus difficile que pour des documents simples. Lam [41], qui adopte une approche ascendante, dirige son regroupement grâce à deux seuils prenant chacun trois valeurs différentes en fonction de la taille des composants traités. Ces deux seuils correspondent à l'espacement maximal qui sépare deux lettres, respectivement deux mots. Beaucoup d'approches sont descendantes et guidées par une description générique de la structure à reconnaître. C'est le cas de l'approche de Nagy [48]. Azokly [3] utilise aussi une description générique, mais son approche est mixte.

Pour la reconnaissance de la structure logique, les approches qui prennent en compte les positions relatives des blocs sont souvent les mieux adaptées aux documents complexes. La plupart des analyseurs de la structure physique retournent une structure hiérarchique qui peut être décrite par la grammaire suivante :

$$
\begin{aligned}
\textit{document} &\rightarrow \textit{page}^{+} \\
\textit{page} &\rightarrow \textit{région}^{+} \\
\textit{région} &\rightarrow (\textit{colonne}|\textbf{filet}|\textbf{illustration}|\textbf{cadre}) \\
\textit{colonne} &\rightarrow \textit{blocs}^{+} \\
\textit{blocs} &\rightarrow \textit{ligne}^{+} \\
\textit{ligne} &\rightarrow \textit{mot}^{+} \\
\textit{mot} &\rightarrow \textbf{signe}^{+}
\end{aligned}
$$

Pourtant, une telle structure ne comprend pas toute l'information contenue dans l'image du document. Dans des documents à structure complexe, la disposition des blocs les uns par rapport aux autres a une sémantique assez riche qu'il convient d'interpréter. Pour représenter les positions relatives d'objets, Klink [38], Walischewski [67] et Azokly [3] utilisent une extension à deux dimensions des relations de Allen [1]. Ce formalisme permet de décrire la position relative de deux intervalles à l'aide des 13 relations schématisées sur la figure 2.3. Pour qualifier la position relative de deux objets, on fait une projection verticale et horizontale des rectangles englobants. On obtient ainsi 2 x 2 segments à comparer, ce qui donne 13 x 13 relations possibles. Klink a enrichi ces relations en leur attribuant une notion de distances, ce qui lui permet d'exprimer des règles du style "une légende doit être située sous une table; la distance entre la légende et la table ne doit pas excéder quatre fois la hauteur de la ligne de la légende". La description de la position relative des blocs fait partie de la structure physique mais il semble que seules les approches qui traitent de la reconnaissance logique s'en soucie.

Niyogi [49], Wang [69], Lam [41], Tsujimoto [62], Ishitani [35] et Klink [38] décrivent tous des systèmes de reconnaissance de la structure logique de documents complexes. Nous allons parler un peu de celui de Tsujimoto qui nous paraît particulièrement original et intéressant. Dans ce système, les positions relatives entre les blocs sont

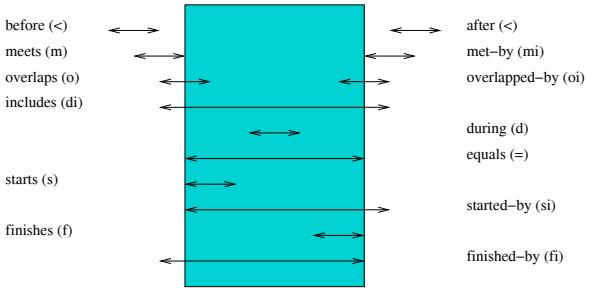
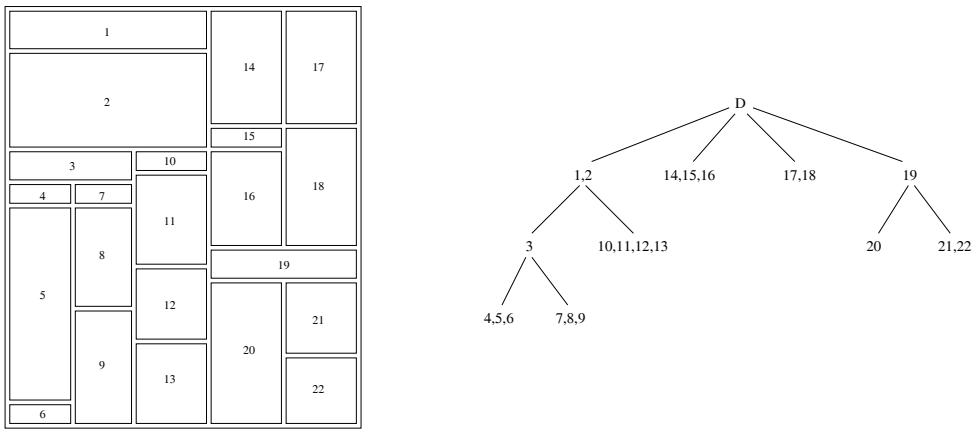


Figure 2.3: Les 13 intervalles de Allen représentés par Walischewski.

représentées par une structure hiérarchique dont la sémantique est tout à fait inhabituelle. On y distingue trois cas : 1) si un bloc a un seul autre bloc situé directement en dessous de lui, alors les deux blocs appartiennent au même noeud de la structure; 2) si un bloc a plus d'un bloc situé directement en dessous de lui, alors le bloc dominant devient le père des autres blocs; 3) si un bloc a plusieurs blocs situés directement en dessus de lui, alors il devient le frère du père des blocs dominants. La figure 2.4 illustre une telle structure et le document auquel elle se rapporte.



a) page de document segmentée en blocs de texte

b) représentation de la structure physique sous forme hiérarchique

Figure 2.4: Exemple fourni par Tsujimoto pour illustrer sa représentation de la structure physique.

Bien que ce système ne soit pas aussi fin que celui de Allen, il saisit très bien une des règles de mise en page implicites les plus générales qui sont à l'origine de la représentation de beaucoup de documents à structure complexe. Par un élégant processus de transformation d'arbres et en tenant compte de *séparateurs* tels que des filets ou des cadres, Tsujimoto obtient la structure logique du document. Malheureusement

l'applicabilité de ce système reste limitée puisqu'il ne peut traiter les documents de type mosaïque par exemple.

Les systèmes de reconnaissance logique de Niyogi, Wang, Lam, Tsujimoto, Ishitani et Klink sont tous basés sur des règles fixes – ce qui ne permet pas d'apprentissage – et sont donc limités à un certain type de document.

2.4 Systèmes doués d'apprentissage

Le développement de systèmes doués d'apprentissage pour la reconnaissance de documents a débuté il y a une dizaine d'années avec la prise de conscience que des systèmes universels s'adaptant à tout type de documents étaient du domaine de l'illusion. Dans les systèmes traditionnels, la connaissance relative à un type de document est implicite et contenue dans la partie analyse du système. L'isolement de ces connaissances dans un modèle indépendant de l'analyse rend le système beaucoup plus général. Au lieu de réécrire tout le système pour chaque type de document, il suffit d'adapter ou de remplacer le modèle. Moins le système est doué en apprentissage, plus le coût de création du modèle est élevé. Si le modèle est entièrement décrit par un expert humain, la faculté d'apprentissage est inexistante. A l'extrême, on a un modèle inféré automatiquement à partir d'un échantillon d'apprentissage constitué de documents déjà reconnus. A noter que même dans ce cas, un opérateur est nécessaire pour constituer l'échantillon et son "fond de vérité".

On distingue plusieurs formes de modèles en fonction de la nature de l'information qui y est représentée. Les *modèles statistiques* sont inspirés comme leur nom l'indique des méthodes statistiques. Les valeurs de caractéristiques obtenues sur un échantillon d'apprentissage sont combinées à l'aide d'opérateurs statistiques. On peut utiliser simplement la moyenne (vecteur moyen de caractéristiques) ou des méthodes plus évoluées comme les méthodes bayésiennes. Les *modèles stochastiques* considèrent les objets à reconnaître comme une suite d'états. Le modèle décrit ces états à l'aide de probabilités de transitions d'état à état et de probabilités d'observations par état. Les *modèles structurels* (ou syntaxiques) permettent de représenter les données structurelles et contextuelles qui se rapportent à un type d'objets. L'information est représentée sous forme de graphe ou de grammaire. Lorsque le modèle est un graphe, la classification consiste à faire de la comparaison de graphes, par contre lorsque le modèle est une grammaire on vérifie si l'objet est une phrase du langage défini par la grammaire.

La plupart des systèmes développés ont trait à la reconnaissance de la structure logique. Des systèmes de reconnaissance de la structure physique doués d'apprentissage sont moins nécessaires, surtout pour des documents à structure simple car les structures physiques génériques varient moins d'une classe de documents à l'autre que les structures logiques génériques. Cependant on trouve certaines méthodes dans

la littérature. Le masque de Williams, dont nous avons parlé dans le point précédent, en est une bonne illustration. Le masque permet d'extraire des caractéristiques sur une portion d'image et de les classer grâce à un réseau de neurones : ainsi les points de contour des zones de l'image sont détectés. Le modèle consiste en les poids et les seuils du réseau. La technique des arbres XY de Nagy [48] est guidée par une grammaire qui décrit les successions de segments noirs et blancs sur la projection de la page d'un document; la saisie de cette grammaire est faite par un expert humain. Le système peut donc être adapté à un plus grand nombre de types de documents, mais n'est pas réellement doué d'apprentissage.

Le modèle de Walischewski [67] est un graphe dont les noeuds sont un inventaire des étiquettes logiques apparaissant dans le type de document modélisé et les arêtes des probabilités sur la position relative existant entre les deux noeuds reliés. Rappelons que la position relative chez Walischewski est décrite verticalement et horizontalement par une des treize configurations de Allen. La construction du modèle se fait à l'aide d'un échantillon de documents reconnus : des règles à base de probabilités sont inférées.

Héroux [24] travaille avec un modèle dont les connaissances sont organisées de trois manières différentes. A chaque type d'objets sont associées diverses informations : une information statistique décrit l'objet proprement dit sous la forme d'un vecteur moyen de caractéristiques, une information probabiliste qui est l'étiquette de l'objet parent avec une valeur de confiance associée et une information structurelle sous la forme d'un graphe représentant l'organisation des fils de l'objet. Le modèle est constitué grâce à un apprentissage supervisé : le vecteur de caractéristiques est le vecteur moyen de l'ensemble des vecteurs de caractéristiques des objets de la base d'apprentissage, l'étiquette de l'objet parent est l'étiquette la plus fréquemment rencontrée dans la base pour ce type d'objets et le graphe est le sous-graphe commun à tous les graphes représentant la structure des objets de la base d'apprentissage. La reconnaissance se fait par l'intermédiaire de trois classifieurs qui correspondent aux trois formes de connaissance et dont les résultats sont fusionnés. Le classifieur statistique retourne la distance entre le vecteur de caractéristiques du modèle et celui extrait sur l'objet à reconnaître. Le classifieur probabiliste retourne 0 si l'étiquette du parent observé ne correspond pas à celle du modèle et *la valeur de confiance* sinon. Quant au classifieur structurel il retourne une distance entre le graphe des fils de l'objet à classer et le graphe du modèle. Cette distance est obtenue en combinant le nombre de noeuds du sous-graphe isomorphe et le nombre de noeuds de chacun des graphes.

Brugger [13] a généralisé le modèle des n-grams pour qu'il puisse prendre en compte des structures d'arbres. Son algorithme de reconnaissance logique consiste à construire l'arbre de la structure logique sur l'arbre de la structure physique en s'aidant des probabilités exprimées par le modèle. Les probabilités modélisent aussi bien les relations entre frères (probabilité qu'un objet porte l'étiquette λ étant données les

étiquettes des deux frères situés avant dans la structure physique) qu'entre père et fils. Le modèle est inféré automatiquement à partir d'échantillons étiquetés en mettant à jour des tables de fréquences.

Un des avantages des méthodes statistiques est leur prédisposition à l'apprentissage incrémental. En effet, le modèle étant souvent constitué de moyennes ou de probabilités, il est facile de le mettre à jour pour prendre en compte un échantillon supplémentaire. Le modèle peut donc évoluer et n'est pas construit une fois pour toutes durant l'apprentissage.

2.5 Conclusion

La plupart des approches proposées dans la littérature s'appliquent à des documents à structure simple. Des études ont été menées sur l'analyse de documents à structure complexe. Parmi celles que nous avons présentées, toutes émettent des hypothèses relativement fortes sur le type de document traité. Nous pensons que pour qu'une approche soit générale elle doit être adaptative et prévoir un modèle des documents analysés qui soit indépendant de l'analyse. La génération manuelle de tels modèles représentant un travail considérable, nous plaidons pour des systèmes doués d'apprentissage qui soient capables de générer plus ou moins interactivement et automatiquement les modèles. Nous n'avons pas connaissance d'approches appliquables à des documents à structure complexe et qui soient douées d'apprentissage. C'est pourquoi cette thèse tente de combler ce manque par la mise au point d'une méthode de reconnaissance de documents complexes douée d'apprentissage.

Chapitre 3

Reconnaissance de documents complexes avec des méthodes simples

Le problème de la reconnaissance d'images de documents complexes est étudié dans cette thèse en deux temps. Dans une première étape, nous avons développé un système de reconnaissance de la structure physique des pages de journaux [53]. Nous nous sommes volontairement limités à l'utilisation de méthodes simples, le but n'étant pas d'obtenir un système performant, mais une sensibilisation à la spécificité de l'analyse d'images de journaux. Le système a également servi à la production des données nécessaires à *2(CREM)*[1], un système de reconnaissance de la structure logique basé sur des patterns. *2(CREM)* est le résultat de la deuxième étape de la thèse et est décrit dans les chapitres 4 et 5.

L'application visée est la reconnaissance de pages de documents à structure complexe de type journaux. Elle a été évaluée sur des pages du Los Angeles Times. Dans la première section nous présenterons des applications de la reconnaissance d'images de journaux et une deuxième section décrira les documents choisis pour nos expériences. Ensuite, nous présenterons les algorithmes simples utilisés dans ce premier système pour la segmentation de documents à structure complexe. Dans la section suivante, nous parlerons de la représentation des données : les choix faits se sont révélés concluants et nous avons gardé la même représentation des données pour la méthode *2(CREM)*. Nous terminerons par une présentation de résultats et une conclusion, énumérant les problèmes que les méthodes simples ne peuvent résoudre.

[1] pour Configuration REcognition Model for Complex Reverse Engineering Methods : CREM+CREM = *2(CREM)* et se prononce "double crème".

3.1 Applications de la reconnaissance d'images de journaux

L'avènement d'Internet a démocratisé l'accès à l'information. Cette information est constituée entre autres de documents textuels et visuels produits principalement durant ces quelques trois ou quatre dernières années. Une masse incroyable de documents traitant de tous les sujets possibles et imaginables est disponible instantanément et en tout temps. Si les documents accessibles par Internet ne couvrent pas loin de tous les domaines, tous les types d'information ne sont pas disponibles. Une recherche sur des articles de journaux n'est pas possible. On trouve tout au plus des articles de l'année courante. Comme nous l'avons vu dans l'introduction, seules les bibliothèques offrent de tels services.

L'édition de journal sous format papier est un domaine dans lequel l'expérience est immense et ce métier a atteint une certaine perfection. Malgré l'impact toujours grandissant d'Internet, les quotidiens continueront à paraître sous forme papier, car c'est la forme qui convient le mieux à la lecture, spécialement à la lecture suivie. Mais la diffusion sur Internet et la lecture par l'intermédiaire d'un navigateur comportent de nombreux atouts. Si un navigateur ne convient pas particulièrement bien pour de la lecture suivie, il est idéal pour d'autres types de lecture : lecture sélective, survol, recherche d'information. Actuellement, chaque éditeur conçoit son propre site et diffuse soit un PostScript ou un PDF de la version papier soit une sélection d'articles sous format HTML/XML avec des liens et des index. La mise à disposition sur Internet de toutes les archives de journaux pourraient se faire grâce à la conversion des images. A partir de la forme logique du journal, on pourrait facilement générer des indexations intelligentes et des services du genre "salle de lecture virtuelle" sont susceptibles d'apparaître. Ils proposeraient au lecteur un vaste choix de journaux du jour et des archives.

3.2 Documents ciblés : les exemplaires du Los Angeles Times

Nous effectuons nos expériences, aussi bien pour la première méthode basée sur des algorithmes simples que pour la méthode *2(CREM)*, sur la forme électronique d'exemplaires du Los Angeles Times (LAT). Ces derniers pouvaient être télé-chargés depuis Internet sous le format PDF. Par une conversion du document PDF au format TIFF, nous obtenons des images synthétiques en tons de gris et d'une résolution de 300 pixels par pouce. Nous les appelons *images idéales* puisqu'elles ne comportent pas le bruit et les distorsions produites par l'opération de saisie d'un document à l'aide d'un numérateur (scanner). Le but ultime du projet est la reconnaissance de

la structure logique de ces exemplaires. Dans ce chapitre, nous traitons la reconnaissance de la structure physique, étape préalable indispensable. Le chapitre 5 traite de la reconnaissance de la structure logique avec la méthode *2(CREM)* présentée dans le chapitre 4. Du point de vue de la structure physique, la page est l'unité à segmenter.

Figure 3.1: Exemple d'une page du Los Angeles Times.

La figure 3.1 représente la partie supérieure de la page de couverture d'un exemplaire du LAT. On y voit une structure complexe qui contient des illustrations, des filets et des cadres ainsi que du texte en différentes polices et réparti en colonnes de largeurs différentes.

Dans le LAT on trouve des cadres regroupant plusieurs éléments (colonnes de texte ou illustrations) et des cadres ne contenant qu'une illustration. Le contenu d'un cadre regroupant plusieurs éléments sera dorénavant appelé "encadré". Voici la définition de la structure physique que nous avons adoptée pour le LAT. Elle met en évidence une macrostructure et une microstructure dont les définitions respectives s'écartent un peu de l'acception usuelle :

- Une page est une entité appelée macrostructure :

$$
\begin{aligned}
\textit{exemplaire du LAT} &\rightarrow (\textit{page})^{+} \\
\textit{page} &\rightarrow \textit{macrostructure} \\
\textit{macrostructure} &\rightarrow (\textit{région}|\textbf{filet})^{+} \\
\textit{région} &\rightarrow (\textit{colonne}|\textbf{illustration}|\textit{encadré}) \\
\textit{encadré} &\rightarrow \textbf{cadre}, \textit{macrostructure} \\
\textit{colonne} &\rightarrow \textit{microstructure} \\
\textit{microstructure} &\rightarrow (\textit{lettrine}?, \textit{ligne}^{+})^{+} \\
\textit{ligne} &\rightarrow \textit{mot}^{+} \\
\textit{mot} &\rightarrow \textbf{signe}^{+} \\
\textit{lettrine} &\rightarrow \textbf{signe}
\end{aligned}
$$

  Une macrostructure se définit récursivement. Elle est composée de régions de

différents types. Une région est la plus grande portion d'image qui se distingue clairement des autres parties du document. Cette distinction peut se faire par la texture du contenu, mais aussi par un cadre ou un filet; une région texte non regroupée par un délimiteur, ne comporte qu'une seule colonne de texte. On obtient ainsi trois types de région : des colonnes de texte, des illustrations et des encadrés. Une illustration est un élément atomique, une colonne est une microstructure et un encadré est une macrostructure.

- Une microstructure se décompose en lignes (parfois il peut y avoir une lettrine), les lignes en mots et les mots en signes. Bien qu'elles ne soient formées que d'un signe, les lettrines sont situées dans la hiérarchie du document au même niveau que les lignes de texte car physiquement elles s'étalent sur plusieurs lignes de texte et ne peuvent être attribuées à une ligne et à un mot.

3.3 Segmentation de documents complexes par des algorithmes simples

Dans l'introduction, nous avons présenté une classification des méthodes de segmentation. Nous avons choisi une stratégie mixte pour notre algorithme. Il procède d'une part de manière ascendante : il part de composantes de base (composantes connexes) et reconstitue la structure par fusions successives. D'autre part, le traitement des encadrés se fait de manière descendante. Dans la phase de segmentation, l'unité analysée est une page du LAT et notre algorithme comporte trois étapes : 1) la détection des éléments graphiques et des encadrés, 2) la segmentation des régions texte en lignes et en mots et 3) la fusion des lignes en colonne.

Reprenant chaque étape de notre algorithme, nous allons décrire son fonctionnement, les problèmes rencontrés ainsi que des propositions qui visent à leur résolution.

3.3.1 Détection des éléments graphiques et des encadrés

Le but de cette étape est la détection des éléments graphiques ainsi que la détection des encadrés (contenu d'un cadre). Par éléments graphiques on entend tous les éléments non textuels, c'est-à-dire les cadres, les filets et les illustrations. A la fin de cette étape, deux des trois types de régions auront été détectés : les illustrations et les encadrés. Il restera à analyser la partie textuelle du document.

La détection des illustrations a déjà largement été traitée dans la littérature [2, 20, 22, 51], c'est pourquoi nous avons peu investi dans cette direction. Nous formulons l'hypothèse que toutes les illustrations sont contenues dans un cadre (ne contenant lui-même pas de cadre) et constituées de tons de gris.

La détection des cadres dans une image idéale ne pose pas de problèmes majeurs.

Une fois les cadres détectés, il s'agit de déterminer si leur contenu est un encadré ou une illustration. **Les encadrés** sont définis par la négation des illustrations. Sur la deuxième image de la figure 3.1, apparaissent deux types d'encadré : l'un ne contient que du texte (pas de tons de gris), l'autre contient une image et du texte. D'après notre définition, une illustration est un élément atomique et un encadré est une macrostructure. L'algorithme de segmentation sera donc appliqué itérativement à l'intérieur de l'encadré.

La détection des filets se fait sur une image dont on a d'abord éliminé les cadres et les illustrations et ensuite extrait les composantes connexes. Un filet est un segment noir s'étalant sur toute la largeur ou sur toute la hauteur d'une composante connexe et étant plus long qu'un seuil donné (l'utilisation d'un seuil permet d'exclure les petits segments contenus dans les caractères). Sur la deuxième image de la figure 3.2, trois filets n'ont pas été détectés : ils ne traversent pas complètement la composante connexe à laquelle ils appartiennent à cause de la présence d'un quatrième filet. Nous résolvons ce problème en filtrant les filets détectés puis en itérant la méthode sur l'image obtenue. Ce processus est répété jusqu'à stabilisation. L'exemple de la figure 3.2 montre les étapes nécessaires à la détection et au filtrage des filets.

Figure 3.2: Filtrage des filets : deux opérations de filtrage sont nécessaires; sur la deuxième image, les deux filets verticaux, ainsi que le souligné du "Los Angeles Times" n'ont pas été éliminés car c'est un filet horizontal qui limite la composante connexe vers le bas.

Les techniques décrites sont très simples. Elles fonctionnent dans le cas général mais devraient être affinées pour traiter tous les cas rencontrés dans le LAT. Si la technique décrite ci-dessus pour la détection des cadres donne 100% de satisfaction, l'hypothèse émise sur les **illustrations** est réductrice. En effet, on trouve des illustrations sous des formes beaucoup plus variées :

- Certaines illustrations ne sont pas contenues dans des cadres. Leur filtrage nécessite des techniques plus évoluées, surtout si l'image n'est pas de forme rectangulaire. Dans [2] on trouve la description d'une telle méthode.
- D'autres illustrations ne sont pas constituées de tons de gris mais sont binaires.

On remarque que la distinction entre des zones de texte et du graphisme n'est pas

aussi triviale que notre algorithme le suggère, notamment lorsqu'on se trouve face à des logos ou à des lettrines (cf. figure 3.3).

a) logo b) lettrine

Figure 3.3: Cas pour lesquels la distinction texte/graphisme est plus difficile.

Comme nous reconnaissons pour filets les segments noirs s'étalant sur toute la largeur ou la hauteur d'une composante connexe, nous ne considérons pas les filets discontinus comme filet. Cependant, nous avons traité ce sujet dans [55]. Seuls les filets plus grands qu'un seuil donné sont retenus. La fixation de ce seuil est critique car la taille de certains filets est plus petite que celle des lettrines et des lettres contenues dans les gros titres. La figure 3.4 montre un titre avant et après une opération de filtrage des filets tel que nous le pratiquons.

(a) image originale (b) image après filtrage des filets

Figure 3.4: Effet du filtrage des filets sur les caractères de grande taille.

Une solution consisterait à conserver les filets qui intersectent une ligne de texte.

3.3.2 Segmentation des régions texte en lignes et en mots

Cette étape ne concerne que la partie textuelle du document. On travaille donc sur une image dont on a éliminé les composants graphiques : filets, cadres et illustrations.

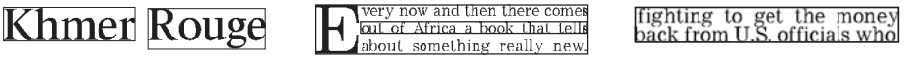
La segmentation en lignes se fait par l'application d'un filtre RLSA (Run Length Smearing Algorithm) [71] horizontal et vertical, la réunion des deux images obtenues puis l'extraction des composantes connexes sur l'image résultante.

Cette approche pose les problèmes classiques de fixation de seuils. Dans RLSA, le seuil détermine la longueur au-dessous de laquelle un segment blanc est noirci. L'espace qui sépare deux lettres du titre étant plus grand que celui qui sépare deux colonnes de texte contiguës, il n'existe pas un seuil global pour tout le document : le seuil devrait être adaptatif et calculé en fonction du contexte.

Dans un premier temps, nous choisissons un seuil qui soit adapté à la taille de la fonte la plus représentée (un histogramme de la hauteur des composantes connexes nous donne une estimation de cette taille). Le seuil adopté est en fait la taille de la fonte la plus représentée. En effet, nous avons observé que généralement l'espacement

inter-mots est plus petit que la hauteur des lignes alors que l'espacement inter-lignes est plus grand.

Deux catégories d'erreur apparaissent : la sur-segmentation des lignes comportant des caractères de grande taille (les lignes de titre par exemple) et la fusion de plusieurs lignes en une seule. La deuxième catégorie d'erreurs est due soit à la présence de lettrines soit à la présence de caractères qui se touchent d'une ligne à l'autre et ainsi groupent les lignes en une seule composante connexe sur l'image RLSA. La figure 3.5 illustre les situations dans lesquelles surviennent ces trois types d'erreur.



(a) (b) (c)

Figure 3.5: Problèmes de segmentation en lignes : (a) ligne comportant des caractères de grande taille ⇒ sur-segmentation, (b) lignes débutant par une lettrine ⇒ sous-segmentation, (c) lignes comportant des caractères se touchant d'une ligne à l'autre ⇒ sous-segmentation.

Le pas suivant consiste à extraire les signes du texte et à les ajouter à la structure reconnue en tant que fils des lignes extraites. Dans une image idéale, les signes correspondent la plupart du temps aux composantes connexes. A noter que les petites composantes du style ponctuation ou signes diacritiques sont éliminées grâce à un filtre qui supprime toute composante dont la largeur et la hauteur sont plus petites que six pixels.

Afin de détecter et corriger les trois types d'erreurs illustrés par la figure 3.5, une opération de correction est appliquée au résultat de segmentation en lignes. On calcule l'histogramme de la hauteur des lignes pour déterminer h, la hauteur de ligne la plus fréquente. Toutes les lignes dont la hauteur est significativement plus grande que h sont considérées comme suspectes et nécessitent un post-traitement. Ce post-traitement comprend trois étapes :

Détection des lettrines. Dans un premier temps, on cherche à détecter si la ligne débute par une lettrine. On réapplique sur la ligne l'opération RLSA et la segmentation en composantes connexes après avoir éliminé le premier signe. Si l'on obtient plusieurs composantes on en déduit que le premier signe est une lettrine. La ligne est donc divisée en plusieurs lignes et le premier signe est reconnu comme lettrine et n'appartient à aucune des lignes.

Détection des lignes ayant des lettres soudées. Si l'on obtient une seule composante, il ne s'agit pas du cas des lettrines et nous devons donc déterminer si deux lettres se touchent provoquant la fusion de deux lignes ou s'il s'agit d'une ligne qui contient des gros caractères. On analyse la hauteur des signes qui composent la ligne. Si la hauteur moyenne est plus petite que le tiers de la hauteur de la ligne, on en déduit qu'il s'agit du cas des lignes fusionnées. Ce

cas n'a pas été résolu automatiquement pour l'instant. Une étiquette "suspect" est attribuée à la ligne.

Détection des lignes à gros caractère. Si l'on ne se trouve dans aucun des deux cas précédents, on considère que la ligne est composée de caractères de grande taille. Ces lignes sont regroupées par catégories en fonction de leur hauteur. Pour chaque catégorie, on produit une image RLSA avec un seuil adapté à l'espacement des mots estimé en fonction de la hauteur des lignes. Les lignes sont fusionnées en fonction du résultat de l'extraction des composantes connexes sur l'image RLSA.

Pour l'extraction des mots on procède de la même manière que pour la correction des lignes de grande taille. On regroupe les lignes en différentes catégories en fonction de leur hauteur. Pour chaque catégorie on produit une image sur laquelle on applique RLSA avec un seuil adapté à la hauteur moyenne des lignes de la catégorie. Une segmentation en composantes connexes nous donne la segmentation en mots.

Les limites de ces techniques sont les limites de toute technique basée sur la fixation de seuils. L'utilisation de seuils adaptatifs améliore la situation, mais il reste le problème des lignes situées à la frontière de deux catégories (leur hauteur est très proche des seuils fixant les limites entre catégories).

3.3.3 Fusion des lignes en colonne

Le but de la dernière étape de l'algorithme est la fusion des lignes de texte de la macrostructure en colonnes. A ce stade, la structure reconnue contient divers éléments : lignes de texte, filets, encadrés et illustrations. L'algorithme de fusion fonctionne de la manière suivante. Soit E l'ensemble de tous les éléments reconnus qui n'ont pas encore été traités et C l'ensemble de tous les éléments en construction. Un élément e appartenant à E est fusionné avec un élément c appartenant à C si les conditions suivantes (illustrées par la figure 3.6) sont remplies :

- parmi les éléments appartenant à C et situés directement en dessus de l'élément e, seul l'élément c a des coordonnées x qui chevauchent celles de l'élément e (cas a et b);
- parmi les éléments appartenant à E situés directement sous l'élément c, seul l'élément e a des coordonnées x qui chevauchent celles de l'élément c (cas a et c);
- l'élément e est de type *ligne de texte* et l'élément c est de type *colonne*.

Si les conditions sont remplies, e est éliminé de E et est intégré à c. Si les conditions ne sont pas remplies, l'élément e est éliminé de l'ensemble E et devient un nouvel élément de l'ensemble des éléments en construction C. Si e est de type *ligne de texte*

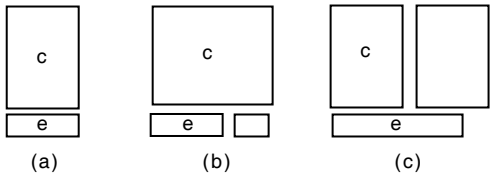


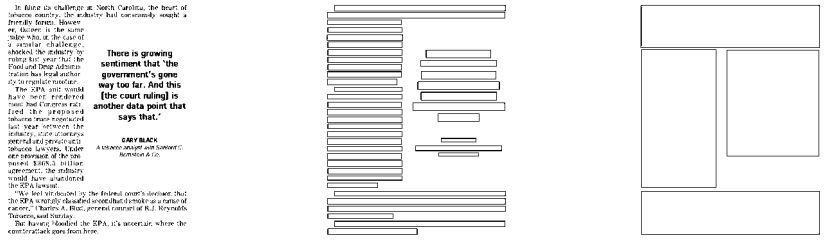
Figure 3.6: Règles de segmentation : (a) e est fusionnée avec c, (b) et (c) e et c ne sont pas fusionnés.

il deviendra un élément de type *colonne*, sinon il ne changera pas de type. Tous les éléments appartenant à C situés directement en dessus de e et dont les coordonnées x chevauchent celles de e sont éliminés de l'ensemble C, car ce ne sont plus des éléments en construction. Tous les éléments appartenant à E situés directement en dessous de c et dont les coordonnées x chevauchent celles de c deviennent de nouveaux éléments de l'ensemble C.

L'algorithme est appliqué itérativement sur tous les éléments appartenant à l'ensemble E dans l'ordre croissant de leurs coordonnées y puis x, c'est-à-dire de gauche à droite et de haut en bas, conformément au sens de lecture usuel.

A noter que dans cet algorithme de fusion des lignes de texte en blocs, les élément *filet*, *illustration* et *encadrés* sont pris en compte. Ils ont un rôle de séparateur : deux lignes de texte séparées par un de ces éléments n'appartiendront pas au même bloc de texte.

La figure 3.7a) illustre le cas particulier d'un bloc polygonal. Notre algorithme de regroupement des lignes de texte en blocs produit la segmentation illustrée par la figure 3.7c) : le bloc polygonal est représenté par trois blocs de forme rectangulaire.



a) image originale b) segmentation en lignes c) fusion des lignes en colonnes

Figure 3.7: Segmentation d'éléments encastrés.

Cet algorithme est relativement bien adapté aux pages du LAT puisque, comme nous le montrerons plus tard dans la section *Résultats et conclusion*, 98,7 % des colonnes ont été segmentées correctement. Les problèmes découlent des étapes précédentes ou de cas particuliers. Un exemple de cas particulier est illustré par la figure 3.8. Suite à la première étape, il y a deux colonnes en construction et cinq lignes non

traitées (3.8b) et après la deuxième étape (3.8c), il y a une colonne construite et trois colonnes en construction. Dans les étapes suivantes, les deux lignes situées en bas à droite sont fusionnées à la colonne en construction située en haut à droite, car la ligne située au centre de l'image n'est plus considérée comme une fille potentielle de la colonne puisqu'elle n'appartient plus à l'ensemble des éléments non traités. On voit facilement quelles modifications apporter à l'algorithme pour qu'il aboutisse à la fusion souhaitée (3.8e) et résolve ce cas particulier, mais il est irréaliste de déterminer a priori tous les cas particuliers.

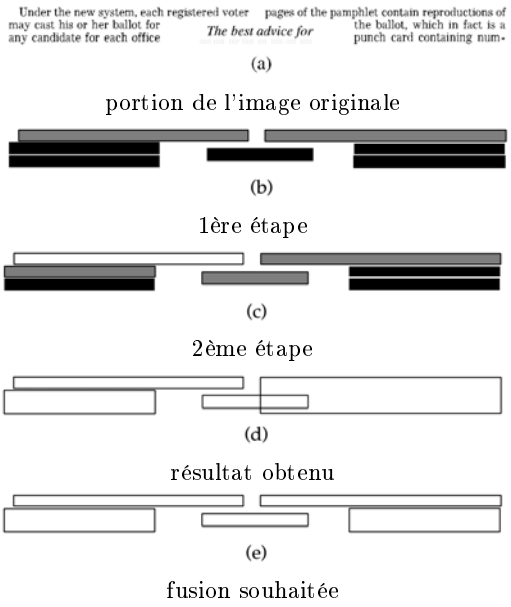


Figure 3.8: Cas particulier dans la fusion des lignes en colonnes : les éléments gris sont les colonnes en construction, les éléments noirs sont les éléments non traités et les éléments blancs sont les colonnes construites.

3.4 Représentation des données

Nous avons choisi XML pour représenter nos données. Dans cette section, nous situons brièvement XML par rapport à d'autres formats dans la problématique de représentation des données en reconnaissance de documents, puis nous montrons comment les caractéristiques de XML en font un format digne de choix. Un dernier point illustre la manière dont nous avons utilisé XML. Le choix de ce format est également motivé par Hitz [28].

3.4.1 XML comme format de représentation des données

En reconnaissance de documents, le problème de la représentation des données se pose sans cesse. Avec chaque nouvel algorithme, nous sommes confrontés au choix du format le mieux adapté. Quelques formats généraux comme DAFS [61] et beaucoup de formats dédiés à une application particulière ont été développés. Malheureusement, aucun n'est suffisamment général et extensible pour convenir à toutes les situations potentielles. Cette diversité dans les formats nuit fortement à l'échange de données entre différents environnements, plate-formes et même entre différents chercheurs.

Nous avons choisi d'utiliser la technologie XML, car nous pensons qu'elle résout les problèmes de généralité et d'extensibilité. XML a eu un impact étonnant dans une foule de domaines différents. Il permet de représenter n'importe quel type d'information.

XML (eXtensible Markup Language) est un langage de description de documents structurés. Comme HTML, il utilise des balises pour structurer l'information. A l'instar de SGML dont il est une simplification et contrairement à HTML, XML est un métalangage qui va permettre d'inventer à volonté de nouvelles balises pour isoler toute information élémentaire ou agrégat d'informations élémentaires. Les DTD (Document Type Definition) et les schémas sont des formalismes utilisés pour définir les balises et la structure des langages respectant la syntaxe XML et dédiés à la description de documents spécifiques. De même HTML respecte la syntaxe SGML et est défini à l'aide d'une DTD. XML distingue deux classes de documents, les documents bien formés et les documents valides. Un document est "bien formé" lorsqu'il obéit aux règles syntaxiques du langage XML. Un document est "valide" s'il est "bien formé" et s'il respecte une structure type définie explicitement dans une DTD ou un schéma.

Avec XML, nous représentons aussi bien les résultats finaux de reconnaissance que les résultats intermédiaires.

3.4.2 Avantages de XML

Nous voyons différents avantages à l'utilisation de XML comme langage de représentation des données. XML étant prévu pour la représentation de documents structurés, son utilisation pour la représentation de la structure logique des documents tombe un peu sous le sens. Par contre, la généralisation de son utilisation à la représentation de la structure physique ou à la représentation de résultats intermédiaires requiert une justification. Les caractéristiques suivantes parlent pour l'utilisation de XML comme format général de représentation des données.

Un standard ouvert et largement accepté. XML est un standard ouvert qui

permet la représentation de données d'une manière simple, flexible et accessible aussi bien à la machine qu'à l'humain. Il est largement accepté, aussi bien du côté de la recherche que du côté de l'industrie. Une quantité d'applications supportant XML sont développées dans des domaines très divers.

XML est accompagné d'extensions très utiles. Par exemple, XSL [66] peut intervenir dans la visualisation de résultats de reconnaissance comme le propose Hitz [26] ou XML-QL [63] est un langage pour exprimer des requêtes sur des données XML dans le style des requêtes formulées à une base de données.

Un format simple et unique. Avoir un seul et même métaformat pour représenter les données facilite la tâche du chercheur. Le développement ainsi que la maintenance de différents résultats est grandement simplifié. D'un autre côté, les documents au format XML sont des données textuelles manipulables par des éditeurs de texte.

Une interface standardisée. Il existe des interfaces (API) standardisées pour simplifier la manipulation de données XML dans les programmes d'applications. Le DOM en est une dont des implémentations sont disponibles pour plusieurs langages de programmation.

3.4.3 Utilisation de XML

D'une part, nous utilisons XML pour la représentation des résultats finaux et intermédiaires. L'idée est de disposer de formats le plus homogènes possible. Comme la structure logique est largement dépendante du contexte de l'application, définir une seule DTD (Document Type Definition) pour la structure logique de n'importe quel document n'est pas possible. En annexe se trouve la DTD construite pour la structure logique du Los Angeles Times. Par contre, nous pensons qu'il est possible de définir une DTD générale pour la reconnaissance de la structure physique de documents. Lefèvre avait déjà fait une proposition qui utilisait SGML [44]. Voici l'exemple d'un extrait d'une DTD XML pour la structure physique ainsi que l'extrait d'un document valide pour cette DTD. Cette DTD est très générale et pourrait convenir à la plupart des documents. Sa version intégrale se trouve en annexe.

L'utilisation de XML ne garantit pas vraiment un format unique pour toutes les applications (notamment pour représenter la structure logique) puisque chaque DTD définit un autre format. Le langage XSLT [64] définit un standard pour transformer des documents XML en d'autres documents. Ainsi, deux documents XML contenant le même type d'information dans leur propre format peuvent parfois être transformés en un seul et même format qui respecte une DTD donnée. Par exemple, la DTD qui précède définit une structure physique plus simple et générale que celle adoptée pour le Los Angeles Times. Mais la structure physique d'un exemplaire du Los Angeles Times peut facilement être transformée de manière à être conforme à la DTD : une macrostructure devient alors un bloc.

```
<!ELEMENT  document  (page+)>
<!ELEMENT  page      (block+)>
<!ELEMENT  block     (((column|block|
                       graphic|...)+)>
<!ELEMENT  column    (paragraph+)>
<!ELEMENT  paragraph (line+)>
<!ELEMENT  line      (word+)>
<!ELEMENT  word      (glyph+)>
<!ELEMENT  glyph     (#PCDATA)>
  ...
<!ATTLIST    document
  image      CDATA           #IMPLIED
  resolution CDATA           #IMPLIED
  ...
>
<!ATTLIST    page
  number     CDATA           #IMPLIED
  ...
>
<!-- number of the page -->
<!ATTLIST    block
  type       (rect|polygon)  "rect"
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
<!-- position and shape of the block -->
...
```

a) structure physique générique

```
<document
    image=
      "/home/images/lyse/LA-tiff/19-3-a-1.tif"
    resolution="300">
  <page>
    <block>
      <column>
        <paragraph>
          <line>
            <word>
              <glyph>L</glyph>
              <glyph>O</glyph>
              <glyph>S</glyph>
              <glyph>A</glyph>
              <glyph>N</glyph>
              <glyph>G</glyph>
              <glyph>E</glyph>
              <glyph>L</glyph>
              <glyph>E</glyph>
              <glyph>S</glyph>
              <glyph>T</glyph>
              <glyph>I</glyph>
              <glyph>M</glyph>
              <glyph>E</glyph>
              <glyph>S</glyph>
            </word>
          ...
</document>
```

b) structure physique spécifique

Figure 3.9: Représentation des structures physiques avec XML.

Oliver Hitz, collaborateur dans notre groupe de recherche, imagine une autre utilisation de XML [26, 29]. Il propose une solution pour la visualisation et la manipulation de résultats de reconnaissance de documents. Son système, baptisé *XMIllum*, s'applique à des données XML et est configurable par des feuilles de style au format XSLT. Que l'on veuille visualiser des résultats de segmentation, de reconnaissance de la structure logique ou autre, l'application reste la même. Seule une feuille de style spécifie comment les résultats doivent être présentés. Ainsi on a plusieurs vues possibles sur un seul document selon le type d'information que l'on veut privilégier. La figure 3.10 montre le même document XML présenté une fois par une mise en évidence de la segmentation en blocs de texte et une autre fois par une mise en évidence de la segmentation en lignes de texte.

XMIllum prévoit également la manipulation des données et devient ainsi un outil précieux pour faire de la reconnaissance assistée. Nous avons largement utilisé XMIllum pour visualiser les résultats des deux méthodes développées et surtout pour l'évaluation de la méthode *2(CREM)* qui est basée sur un apprentissage incrémental : la partie interface homme-machine y est donc spécialement critique.

Figure 3.10: Visualisation de résultats de reconnaissance avec XMIllum.

3.5 Résultats et conclusion

Le but de notre étude était de mettre en relief les vrais problèmes de segmentation de documents à structure complexe en s'affranchissant des problèmes de dégradation dus à la saisie de l'image. Nous avons choisi des techniques simples qui fonctionnent dans le cas général. Pour certaines des étapes, ces techniques donnent de relativement bons résultats, pour d'autres elles ne sont pas suffisantes. Des tests ont été effectués sur 31 pages du LAT. Le tableau 3.1 présente les résultats de détection des illustrations, des lignes de texte et des colonnes.

	Illustrations	**Lignes de texte**	**Colonnes**
Reconnu	60 (78%)	14558 (99.8%)	841 (98.7%)
Non reconnu	17 (22%)	30 (0.2%)	11 (1.3%)

Tableau 3.1: Résultats des tests de segmentation réalisés sur 31 pages du LAT.

Les méthodes simples utilisées pour la détection des cadres et des filets fonctionnent bien. Parmi les filets, seuls les filets non continus ne sont pas reconnus. Ce type de filet est très rare. Quant aux cadres, ils ne sont pas reconnus uniquement lorsqu'ils présentent un défaut, par exemple lorsqu'ils ne sont pas connexes ou lorsque le contenu déborde le cadre, mais ces cas étaient extrêmement rares dans les documents analysés.

Un investissement minimum a été apporté à la détection des éléments graphiques et en regard des résultats obtenus, il est clair que la technique est insuffisante. Sur les 31 pages testées, 17 illustrations non reconnues ont été dénombrées (des illustrations binaires ou non délimitées par un cadre).

Pour les lignes de texte nous obtenons un taux de reconnaissance élevé. Il faut cependant souligner que 12 des 31 pages analysées contenaient des lignes mal segmentées. Les échecs sont dus à des erreurs dans la production du document (lignes dépassant le bloc auquel elles appartiennent et étant ainsi fusionnées avec une ligne du bloc

voisin), à un mauvais choix des seuils ou encore à la méthode elle-même. En effet, même avec un choix optimal de seuils, la technique ne permet pas de segmenter correctement des lignes en fonte courrier. Cette fonte a des espacements inter-caractères sensiblement plus grands que les autres fontes et ses espacements inter-mots sont dans certains cas plus grands que les espacements inter-lignes.

Quant à la détection des colonnes de texte, elle pose problème sur 6 des 31 pages. Les échecs sont tous dus à des cas particuliers tels que présentés dans la figure 3.8. La figure 3.11 montre le résultat de la segmentation en blocs d'une page du Los Angeles Times. Sur la figure 3.12 on voit des résultats de segmentation en blocs erronés qui correspondent à des cas particuliers de configurations des colonnes.

Figure 3.11: Page du Los Angeles Times segmentée avec l'algorithme de fusion des lignes de texte en blocs.

On remarque sur la gauche de la figure 3.12b) une deuxième erreur : il s'agit d'une erreur de segmentation en lignes qui est répercutée sur la segmentation en blocs.

(a) (b)

Figure 3.12: Deux cas où la segmentation en blocs pose problème.

Chapitre 4

2(CREM) : méthode de reconnaissance structurelle basée sur des patterns

2(CREM), la méthode que nous avons mise au point, est une méthode générale qui est utile à la reconnaissance de la structure physique aussi bien que logique. Elle est également décrite dans [54]. Son but est la classification (ou reconnaissance) d'objets physiques tels que les blocs de texte ou les filets. Elle procède par un recensement de tous les *patterns* (motifs) que peut former un objet d'une classe donnée avec ses voisins.

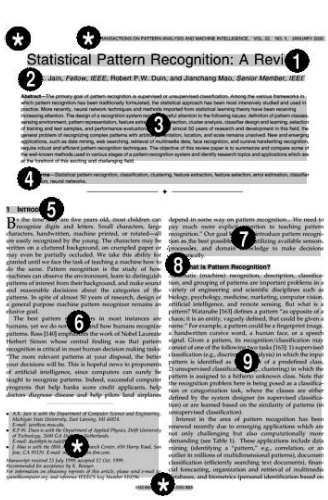
4.1 Choix fondamentaux

Notre objectif était la conception d'une méthode pour la reconnaissance des structures qui, d'une part, s'applique à des documents à structure complexe et, d'autre part, permette un apprentissage incrémental.

Rappelons que nous classons dans les documents à structure simple ceux qui sont organisés physiquement sur une ou deux colonnes et logiquement en titre, sous-titre et contenu (ex.: articles scientifiques, lettres ou livres). Les étapes de l'analyse de tels documents sont les suivantes : 1) reconnaissance de la structure physique puis 2) analyse de la structure logique. La recherche de la structure physique consiste à détecter si l'on a un document à une ou deux colonnes, à isoler les colonnes puis à découper la page en blocs texte ou illustration et les blocs de texte en lignes, mots et signes. La reconnaissance logique consiste à mettre en correspondance la structure physique avec un modèle hiérarchique de la structure logique.

La richesse des structures change la problématique. L'information n'est plus organisée de manière linéaire (de gauche à droite et de haut en bas), mais dans un espace à deux dimensions. Ceci pose le double problème de l'identification des blocs de

texte et de la détermination de l'ordre de lecture. Un bloc de texte est une région homogène qui ne comprend qu'une seule colonne. L'homogénéité se traduit par une fonte unique et un interligne constant. Pour des documents simples, le découpage en blocs se fait par l'analyse des espaces blancs et des fontes. On peut facilement établir des règles avec des seuils adaptatifs pour l'interprétation des espaces blancs. Avec des documents à structure complexe, on ne connaît ni le nombre, ni l'emplacement des colonnes. De plus, les blocs peuvent être de forme non rectangulaire. Tous les différents cas de structuration physique ne peuvent être décrits par des règles prédéfinies : il se présente toujours de nouveaux cas, surtout si l'on change de classe de documents. C'est pourquoi une approche avec apprentissage incrémental est particulièrement bien adaptée. La figure 4.1 compare la découpe en blocs et l'ordre de lecture d'un document simple et d'un document complexe. Les pastilles noires contiennent un numéro ou une lettre qui désignent l'ordre dans lequel les blocs doivent être lus. Les étoiles indiquent que l'ordre est indifférent pour le bloc en question. On remarque pour le document à structure complexe plusieurs groupes de blocs ordonnés entre eux mais indépendants par rapport aux autres (système de numérotation différent).



a) document à structure simple

b) document à structure complexe

Figure 4.1: Découpe en blocs et ordre de lecture.

Si, comme pour les documents simples, on représente la structure physique par un arbre, on perd l'information contenue dans les relations de voisinage : cette information est indispensable pour le regroupement des blocs en articles ou pour la reconnaissance de l'étiquette logique d'un bloc. Nous proposons donc de représenter la structure physique de documents complexes par un graphe afin d'y transcrire les relations de voisinage. Il est difficile de mettre en correspondance un graphe de

la structure physique avec un modèle hiérarchique logique. La transformation du graphe en arbre se fait par le recouvrement de l'ordre de lecture. Cette étape est implicite dans l'analyse de documents à structure simple puisque l'ordre de lecture est connu : de gauche à droite et de haut en bas.

A partir de la structure physique, il est en fait plus facile de procéder à un étiquetage logique des blocs avant de chercher à retrouver l'ordre de lecture. En effet, la fonte d'un bloc de texte est un indice assez fiable sur son étiquette logique. A cela viennent s'ajouter des indices intrinsèques comme le nombre de lignes et des indices externes comme le voisinage du bloc (séparateurs ou autres blocs). Nous avons donc opté pour un étiquetage logique qui précède le recouvrement de l'ordre de lecture. Il a fallu abandonner le modèle hiérarchique de la structure au profit d'un modèle qui permette d'exprimer la deuxième dimension et qui représente la structure sous la forme d'un graphe. Une fois les blocs étiquetés, le recouvrement de l'ordre de lecture est nettement plus aisé. Rien n'empêche ensuite d'utiliser un modèle hiérarchique de la structure logique comme moyen de confirmation/remise en cause des étiquettes logiques. Le tableau 4.1 synthétise les étapes pour la reconnaissance de documents à structure simples et complexes ainsi que les étapes de la méthode *2(CREM)*.

structure simple	**structure complexe**	
	une approche possible	*2(CREM)*
1) reconnaissance de la structure physique 2) mise en correspondance avec un modèle logique hiérarchique	1) reconnaissance de la structure physique 2) recouvrement de l'ordre de lecture 3) mise en correspondance avec un modèle logique hiérarchique	1) reconnaissance de la structure physique 2) étiquetage logique avec un modèle sous forme de graphe 3) recouvrement de l'ordre de lecture 4) remise en cause / confirmation avec un modèle logique hiérarchique

Tableau 4.1: Étapes de l'analyse de documents à structure simple et complexe et étapes de la méthode choisie.

4.2 *2(CREM)* : une méthode générale de classification d'objets

A la base de notre approche de la reconnaissance structurelle, il y a un principe général de classification d'objets qui est simple et naturel. Appliqué à la reconnaissance

de documents, le principe peut paraître très complexe, c'est pourquoi dans cette section nous allons le présenter en faisant abstraction du domaine de la reconnaissance de documents. Nous illustrerons notre propos par des exemples de classification d'animaux.

Le but de *2(CREM)* est d'identifier des objets en les associant à une classe, tout en augmentant la capacité d'identification. Il est important de souligner ce double objectif de classification et d'apprentissage car les deux processus sont intimement liés et la classification ne peut se comprendre sans la construction de la connaissance.

On peut imaginer que la construction de la connaissance chez l'humain se fasse de la façon suivante. On montre un chien à un enfant et on lui dit "c'est un chien". Si l'enfant ne connaît aucun autre nom d'animaux, il y a des chances pour qu'il identifie tous les animaux comme étant des chiens. Puis, lorsqu'il désigne un chat et l'appelle chien, on le corrige; il semble alors naturel qu'il recherche la caractéristique qui permette de distinguer le spécimen de chien qu'il connaît de ce nouvel animal. Si le spécimen était un grand chien, ce pourrait être la taille. L'enfant a maintenant deux représentations d'animaux auxquelles il associe la classe "chien", respectivement "chat" et il sait que pour différencier ces deux classes, la taille est une caractéristique pertinente. Lorsqu'il va rencontrer un petit chien, il risque de l'appeler chat. Si on le corrige, il va associer une deuxième représentation à la classe "chien" et trouver une nouvelle caractéristique pour distinguer les chiens des chats, par exemple le cri.

Ça n'est probablement pas exactement le principe de l'apprentissage chez l'humain, mais c'est le principe de notre méthode.

Tout objet se décrit par une quantité innombrable de caractéristiques. Il est raisonnable de penser que toute classification — faite aussi bien par un humain que par une machine — se base sur l'interprétation de quelques caractéristiques pertinentes. C'est pourquoi dans notre approche nous modélisons l'objet par un ensemble de caractéristiques pertinentes avant de le classer. Cette modélisation est appelée *configuration*. Les caractéristiques sont pertinentes par rapport au domaine des objets que l'on veut classer. Par exemple, des caractéristiques pertinentes pour la classification d'un animal seraient le mode de déplacement, la taille, le nombre de pattes ou la couleur, alors que pour un polygone, on aurait le nombre de côtés, les angles ou la relation entre la longueur des côtés.

Il existe en fait deux niveaux de caractéristiques pertinentes : les caractéristiques pertinentes pour un domaine (les animaux) et les caractéristiques pertinentes pour une classe d'objets (les chiens). Les caractéristiques pertinentes pour une classe d'objets sont un sous-ensemble des autres caractéristiques. Elles sont utilisées pour la classification proprement dite alors que les autres caractéristiques sont utilisées pour la construction de la configuration. Par exemple, la couleur serait une caractéristique pertinente pour la classe zèbre, alors que pour la classe kangourou on aurait le mode de déplacement.

Dans notre modèle de classification, nous avons une partie statique commune à toutes les classes qui sont les caractéristiques pertinentes pour le domaine des objets à classer et une partie dynamique propre à chaque classe. Cette partie est constituée d'un ensemble de configurations de référence appelées *patterns* (les patterns correspondent aux deux représentations de chien qui étaient la référence pour l'enfant) et un ensemble de caractéristiques pertinentes par rapport à la classe (par exemple la couleur pour la classe zèbre) désignées à l'aide d'un sélecteur de caractéristiques.

Donc pour classer un animal, nous allons construire une configuration qui contient les valeurs des caractéristiques énumérées dans la partie statique, puis nous allons comparer la configuration avec les patterns de chaque classe d'animaux connue. La comparaison porte sur les caractéristiques propres à chaque classe. Par exemple pour qu'un animal soit classé zèbre, il faut qu'il ait la couleur noir et blanc.

Si le système produit un résultat erroné, ce résultat sera corrigé et retourné vers le système. Le système analysera l'erreur et modifiera le modèle en conséquence. Par exemple, si une chèvre noir et blanc est associée à la classe zèbre, on va opérer les deux modifications suivantes sur le modèle :

- une *extension* : la configuration de la chèvre devient un pattern supplémentaire de la classe chèvre, et
- une *spécialisation* : on ajoute une caractéristique à la classe zèbre de manière à ce que la chèvre ne puisse plus y être classée, par exemple la taille.

Le modèle peut également subir une *généralisation* : une caractéristique est supprimée d'une classe. Une telle modification a lieu lorsqu'une caractéristique est inutile ou lorsqu'elle est trop discriminante. Par exemple la caractéristique de la taille n'est finalement pas une bonne caractéristique pour la classe "chien". A noter que dans nos réalisation la généralisation n'a pas été intégrée.

4.3 *2(CREM)* et les approches classiques de la reconnaissance des formes

La reconnaissance des formes a pour but de classer des observations. On distingue généralement trois étapes à la reconnaissance des formes : 1) le pré-traitement, 2) l'extraction de caractéristiques et 3) la classification. Les quatre approches classiques les plus connues sont l'iconographie ("template matching"), les méthodes statistiques, les méthodes structurales (ou syntaxiques) et les réseaux de neurones. *2(CREM)* ne peut être assimilée à l'une de ces approches mais elle présente des points communs avec les trois premières approches que nous allons brièvement décrire en nous inspirant de [36]. Ensuite nous allons situer *2(CREM)* par rapport à ces approches avant de passer à une description plus formelle dans le point suivant. Ainsi nous espérons jeter une passerelle entre les notions très concrètes de l'analyse de documents et

un formalisme mathématique abstrait. La figure 4.2 est une illustration de notre méthode qui devrait aider à la concrétisation des notions exposées dans le point suivant.

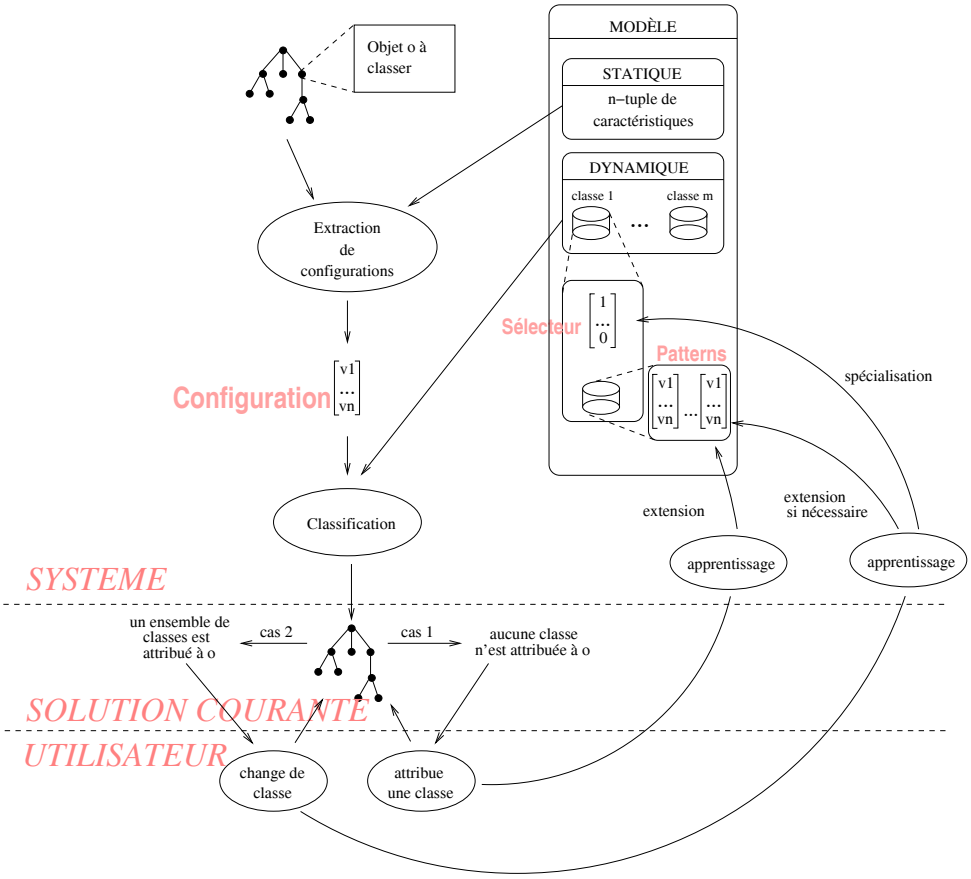


Figure 4.2: La méthode *2(CREM)*.

4.3.1 Trois approches classiques de la reconnaissance des formes

L'iconographie. Elle est plus connue sous son appellation anglaise "template matching". Le modèle d'une classe est un patron ("template"), c'est-à-dire un prototype des observations à reconnaître. Une fonction de *mise en correspondance* ("matching") retourne une mesure de similitude entre une observation et un patron.

La méthode statistique. Dans la méthode statistique, chaque observation est re-

présentée par n caractéristiques et est donc un point dans un espace à n dimensions. Le but est de trouver les frontières de décision de cet espace de caractéristiques afin de pouvoir classer les observations. Les frontières sont déterminées par un modèle probabiliste qui est soit spécifié au départ, soit inféré d'une base d'apprentissage.

La méthode syntaxique. L'approche syntaxique (appelée aussi structurelle) est particulièrement bien adaptée aux observations de nature complexe : elles sont décrites comme une composition d'éléments plus simples appelés *primitives* et ayant des liens entre eux. Ces observations sont souvent représentées par des graphes. Les classes sont décrites par des grammaires et définissent un langage auquel les observations doivent appartenir pour être assimilées à la classe. Une observation est donc une phrase et les primitives forment l'alphabet utilisé par les langages des différentes classes.

4.3.2 *2(CREM)*

Dans notre méthode nous cherchons à reconnaître les objets physiques d'un document tels que des blocs de texte ou des objets structurant la présentation du document (filets ou cadres). Ces objets, simples en eux-mêmes, ne peuvent la plupart du temps être reconnus sans l'analyse de leur contexte. C'est pourquoi les observations que nous allons classer seront des modélisations de ces objets physiques et de leur contexte en entités appelées *configurations*. Ces configurations peuvent être représentées par des graphes dont les arcs et les sommets sont étiquetés. Comme dans l'approche syntaxique, on cherche à classer un graphe, mais ici l'objet à reconnaître n'est pas un objet complexe constitué de primitives, mais une primitive entourée d'autres primitives.

Comme dans l'approche statistique, une configuration peut être vue comme un tuple de caractéristiques si l'on désigne chaque sommet du graphe par sa relation avec l'objet à reconnaître. Une caractéristique sera par exemple "la taille de la fonte du voisin supérieur le plus à droite". Nos observations deviennent alors des points dans un espace de caractéristiques. La comparaison avec l'approche statistique s'arrête ici car nous n'utilisons pas d'opérateurs statistiques pour construire le modèle.

Partant de là, nous allons décrire notre méthode en parlant de l'extraction de caractéristiques, du modèle du document, de la classification et de l'apprentissage.

Extraction de caractéristiques. Les caractéristiques d'un objet sont :

- ses attributs (dimensions, fonte, ...),
- les attributs de ses voisins,
- ses liens avec ses voisins (positions relatives, comparaison des tailles respectives, ...),

- les liens entre ses voisins.

Chaque objet est représenté par un n-tuple de caractéristiques valué (tuple de couples caractéristiques-valeurs, ex. "taille de la fonte du voisin supérieur le plus à droite"-"16"), la *configuration*. A chaque objet correspond un point dans un espace à n dimensions (n est le nombre de caractéristiques dont nous disposons).

Modèle de document. Le modèle est la description des classes d'objets pour un type de documents. Il est composé d'une partie statique commune à toutes les classes d'objets :

- un ensemble de n caractéristiques pouvant s'appliquer aux objets que l'on veut classer,

et d'une partie dynamique qui décrit chaque classe :

- un ensemble de configurations de référence pour la classe appelées *patterns* dont les caractéristiques sont les mêmes que celles de la partie statique du modèle; un pattern est donc également un point d'un espace de dimension n.
- un n-tuple binaire (0 ou 1) pour sélectionner les caractéristiques de la partie statique et appelé *sélecteur de caractéristiques*.

Le sélecteur de caractéristiques désigne les caractéristiques à prendre en considération pour la classification; il engendre un espace à m dimensions, m étant le nombre de composantes du sélecteur à 1; par projection, les patterns deviennent des points dans cet espace. Les points de l'espace non occupés par un pattern appartiennent à la classe *inconnue*. A noter que le système garantit qu'aucun pattern ne se superpose à un autre pattern du modèle : d'une part, une configuration devient un nouveau pattern d'une classe du modèle seulement si elle ne se superpose à aucun point (pattern) de la classe, d'autre part, toutes les classes du modèle qui ne sont pas la classe de l'objet représenté par la configuration et dont un pattern se superpose avec la configuration sont projetées dans un espace dans lequel aucun de leurs patterns ne se superpose plus à la configuration.

Classification. Sont associées à un objet toutes les classes du modèle pour lesquelles il existe un pattern avec la même projection que la configuration de l'objet dans l'espace engendré par les caractéristiques de la classe considérée.

Apprentissage. L'apprentissage se fait par spécialisation ou extension d'une classe d'objets : la spécialisation correspond à l'ajout d'une caractéristique et l'extension à l'ajout d'un pattern.

Dans l'approche statistique, le modèle est une partition de l'espace de caractéristiques déterminée par les probabilités. Notre modèle ressemble plus à celui de l'iconographie avec son patron, mais chez nous chaque classe est représentée par plusieurs patrons que nous appelons patterns ainsi que par un ensemble de caractéristiques; ces caractéristiques font partie de la mise en correspondance puisqu'elles spécifient les caractéristiques à prendre en compte pour la classification. Ce modèle à plusieurs patrons permet un apprentissage incrémental.

L'approche des "k plus proches voisins" présente aussi certaines similitudes avec notre approche. Elle consiste à définir une mesure de distance entre observations (configurations) et à regrouper en classes les observations qui sont proches. Dans *2(CREM)* nous n'avons pas de mesure de distance : pour qu'une configuration se voie attribuer une classe, il faut qu'elle ait exactement les mêmes valeurs qu'un des patterns de la classe sur un ensemble de caractéristiques donné.

Sur la figure 4.2, on retrouve *2(CREM)* avec sa partie statique commune à toutes les classes et sa partie dynamique propre à chaque classe. L'analyse se fait en 2 partie : l'extraction de caractéristiques (extraction de configurations) et la classification (mise en correspondance). L'apprentissage se fait par réaction aux interventions de l'utilisateur; nous sommes restés volontairement vagues sur ce point car une description plus précise est donnée dans la section suivante. La communication entre le système et l'utilisateur se fait au travers de la solution courante que chacune des parties peut consulter et modifier.

4.4 Formalisation de *2(CREM)*

Dans cette section nous allons présenter notre méthode de manière formelle. Après quelques notions en rapport avec la théorie des graphes, nous parlerons de notre modélisation de la structure physique d'un document, du modèle de notre approche ainsi que des algorithmes d'extraction de caractéristiques, de classification et d'apprentissage. Une série d'exemples illustrera les principaux points de notre formalisation.

4.4.1 Théorie des graphes

Le but de ce point est de définir la notion de graphe étiqueté et attribué en utilisant la notion d'ensemble étiqueté et attribué. Les graphes étiquetés et attribués seront utilisés pour représenter la structure physique d'un document.

Définition : ensemble étiqueté

Soit E un ensemble fini de noms appelés étiquettes. Soit X un ensemble quelconque. Le couple (X, μ) avec $\mu : X \to \mathcal{P}(E)$[1] est appelé *ensemble étiqueté par E*. (En règle générale, une seule étiquette est assignée à un élément de X, mais il peut arriver qu'il y ait plusieurs étiquettes.)

Avant de définir la notion d'ensemble attribué, précisons qu'un attribut est une paire <nom, valeur>.

Définition : ensemble attribué

Soit $D = \{d_i, i \in I\}$ un ensemble fini de noms d'attributs et pour chaque $d_i \in D$, soit V_i le domaine des valeurs qui lui est associé. Posons $V = \left(\bigcup_{i \in I} V_i\right) \cup \{\text{"indéfini"}\}$. Soit X un ensemble quelconque. Le couple (X, ν) avec $\nu : X \to T$, où T est l'ensemble des fonctions de D vers V, est appelé *ensemble attribué par T*.

Définition : ensemble étiqueté et attribué

Un triplet (X, μ, ν) est appelé *ensemble étiqueté par E et attribué par T* si (X, μ) est un ensemble étiqueté par E et (X, ν) est un ensemble attribué par T.

Définition : graphe orienté

Soit S un ensemble fini. Le couple (S, A) avec $A \subseteq S \times S$ est appelé *graphe orienté*. Les éléments de S sont appelés *sommets* et ceux de A sont appelés *arcs*.

Définition : graphe étiqueté et attribué

Soit (S, A) un graphe orienté. Un *graphe étiqueté et attribué* est un couple (Σ_S, Σ_A) où :

- $\Sigma_S = (S, \mu_S, \nu_S)$ est un ensemble de sommets étiqueté et attribué et
- $\Sigma_A = (A, \mu_A, \nu_A)$ est un ensemble d'arcs étiqueté et attribué.

4.4.2 Structure physique d'un document

La structure physique d'un document pour une application donnée sera représentée par un graphe étiqueté et attribué. Les sommets du graphe représentent les objets physiques dont les étiquettes seront par exemple "bloc", "mot", "caractère", "illustration", "filet", ... Ces sommets possèdent comme attributs dimension, classe de la fonte, nombre de lignes, ... Les arcs entre les sommets dénotent des relations de voisinage étiquetées selon l'orientation (gauche, droite, supérieure, inférieure, ...

[1] Pour un ensemble E, $\mathcal{P}(E)$ désigne l'ensemble de tous les sous-ensembles de E.

) et possèdent des attributs comme distances, positions relatives, comparaisons de tailles ...

Voici notre premier exemple : nous allons y proposer quelques abréviations qui seront reprises dans chaque exemple et nous illustrerons la représentation de la structure physique d'un extrait de document sous la forme d'un graphe orienté, étiqueté et attribué.

Exemple 1: structure physique sous forme de graphe étiqueté et attribué

Avant de présenter un exemple de structure physique, nous allons énumérer les abréviations utilisées pour cet exemple et les suivants.

La table 1 énumère des identificateurs d'attributs sur les objets et sur les liens entre objets ainsi que le domaine des valeurs que peuvent prendre ces attributs :

Abréviations	**Attributs**	**Domaines de valeurs**
$Type$	type d'un objet	$\{blocDeTexte, filet\}$
$Fonte$	classe de la taille de la fonte d'un objet	$\{petit, moyen, grand\}$
$Nbligne$	nombre de lignes de texte d'un objet	$\{1, 2\text{-}5, 5\text{-}\infty\}$
Pos	position relative de deux objets	$\{V_{IG}, V_{ID}, ID, \ldots\}$
$CompF$	comparaison des tailles de la fonte de 2 objets	$\{<, >, =\}$

table 1

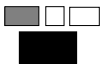
A noter que *Type* désigne en même temps l'étiquette associée aux objets et l'un des attributs associés aux objets, alors que *Pos* désigne en même temps l'étiquette associée aux arcs et l'un des attributs associés aux arcs. Cette redondance s'explique par le fait que *Type* et *Pos* sont d'une part des étiquettes permettant de désigner un objet ou un groupe d'objets, et d'autre part des attributs sur un objet ou un couple d'objets.

La table 2 énumère quelques abréviations utilisées pour les étiquettes des arcs, désignant les relations de voisinage entre objets :

ID	identité (désigne l'objet-même)
V_{SG}	voisin supérieur gauche d'un objet
V_{SD}	voisin supérieur droit d'un objet
V_{IG}	voisin inférieur gauche d'un objet
V_{ID}	voisin inférieur droit d'un objet
V_{DS}	voisin droit supérieur d'un objet
$V_{IG}.V_{IG}$	voisin inférieur gauche du voisin inférieur gauche d'un objet

table 2

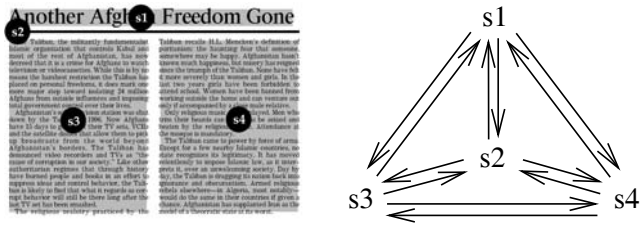
La figure ci-dessous illustre la différence qu'il existe entre un voisin supérieur gauche (V_{SG}) et un voisin gauche supérieur (V_{GS}) : le rectangle noir représente l'élément de référence et le rectangle gris son voisin V_{SG}, respectivement V_{GS}.



a) V_{SG}

b) V_{GS}

Les deux figures ci-dessous illustrent la représentation de la structure physique d'un document sous la forme d'un graphe étiqueté et attribué (Σ_S, Σ_A) avec $\Sigma_S = (S, \mu_S, \nu_S)$ étiqueté par E_S et attribué par $T_S \equiv D_S \rightarrow V_S$, et $\Sigma_A = (A, \mu_A, \nu_A)$ étiqueté par E_A et attribué par $T_A \equiv D_A \rightarrow V_A$.



Nous allons énumérer les sommets, arcs, étiquettes et attributs de la structure physique en utilisant les abréviations définies dans l'exemple 1.

Sommets et arcs :

$$S = \{s_1, s_2, s_3, s_4\}$$
$$A = \{< s_1, s_2 >, < s_1, s_3 >, < s_1, s_4 >, < s_2, s_1 >, < s_2, s_3 >, < s_2, s_4 >, < s_3, s_1 >, < s_3, s_2 >, < s_3, s_4 >, < s_4, s_1 >, < s_4, s_2 >, < s_4, s_3 >\}$$

Étiquettes sur les sommets :

$$E_S = \{blocDeTexte,\ filet\}$$

Le tableau suivant fait correspondre à chaque sommet s son étiquette $\mu_S(s)$:

Sommets	Etiquettes
s_1	*blocDeTexte*
s_2	*filet*
s_3	*blocDeTexte*
s_4	*blocDeTexte*

Étiquettes sur les arcs :

$$E_A = \{V_{SG}, V_{SD}, V_{IG}, V_{ID}, V_{DS}, V_{DI}, V_{GS}, V_{GI}, V_{SG}.V_{SG}, V_{SG}.V_{SD}, V_{SD}.V_{SD}, V_{SD}.V_{SG}, V_{IG}.V_{IG}, V_{IG}.V_{ID}, V_{ID}.V_{ID}, V_{ID}.V_{IG}\}$$

Le tableau suivant fait correspondre à chaque paire de sommets $< s_i, s_j >$ son ensemble d'étiquettes $\mu_A(< s_i, s_j >)$:

	s_1	s_2	s_3	s_4
s_1	$\{ID\}$	$\{V_{IG},\ V_{ID}\}$	$\{V_{IG}.V_{IG},\ V_{IG}.V_{ID}\}$	$\{V_{ID}.V_{IG},\ V_{ID}.V_{ID}\}$
s_2	$\{V_{SG},\ V_{SD}\}$	$\{ID\}$	$\{V_{IG}\}$	$\{V_{ID}\}$
s_3	$\{V_{SG}.V_{SG},\ V_{SG}.V_{SD},\ V_{SD}.V_{SG},\ V_{SD}.V_{SD}\}$	$\{V_{SG},\ V_{SD}\}$	$\{ID\}$	$\{V_{DS},\ V_{DI}\}$
s_4	$\{V_{SG}.V_{SG},\ V_{SG}.V_{SD},\ V_{SD}.V_{SG},\ V_{SD}.V_{SD}\}$	$\{V_{SG},\ V_{SD}\}$	$\{V_{GS},\ V_{GI}\}$	$\{ID\}$

Noms d'attributs sur les sommets :

$$D_S = \{Type,\ Fonte,\ Nbligne\}$$

Le tableau suivant fait correspondre à chaque sommet s la classe de sa fonte $\nu_S(s)(Fonte)$:

Sommets	Fonte
s_1	*moyen*
s_2	*indéfini*
s_3	*petit*
s_4	*petit*

Le tableau suivant fait correspondre à chaque sommet s son nombre de lignes $\nu_S(s)(Nbligne)$:

Sommets	Nbligne
s_1	1
s_2	2-5
s_3	5-∞
s_4	5-∞

Noms d'attributs sur les arcs :

$$T_A \quad = \quad \{Pos, CompF\}$$

Le tableau suivant fait correspondre à chaque paire de sommets $< s_i, s_j >$ sa position $\nu_A(< s_i, s_j >)(Pos)$:

	s_1	s_2	s_3	s_4
s_1	ID	V_{IG}	$V_{IG}.V_{IG}$	$V_{ID}.V_{IG}$
s_2	V_{SG}	ID	V_{IG}	V_{ID}
s_3	$V_{SG}.V_{SG}$	V_{SG}	ID	V_{DS}
s_4	$V_{SG}.V_{SG}$	V_{SG}	V_{GS}	ID

Le tableau suivant fait correspondre à chaque paire de sommets $< s_i, s_j >$ la comparaison de la taille de fontes des deux sommets $\nu_A(< s_i, s_j >)(CompF)$:

	s_1	s_2	s_3	s_4
s_1	$=$	*indéfini*	$<$	$<$
s_2	*indéfini*	*indéfini*	*indéfini*	*indéfini*
s_3	$>$	*indéfini*	$=$	$=$
s_4	$>$	*indéfini*	$=$	$=$

4.4.3 Modèle

Dans ce point nous décrivons le modèle: il contient la connaissance du système sur les objets qu'il doit classer.

Soit une structure physique décrite par un graphe étiqueté et attribué (Σ_S, Σ_A) avec $\Sigma_S = (S, \mu_S, \nu_S)$ étiqueté par E_S et attribué par $T_S \equiv D_S \rightarrow V_S$, et $\Sigma_A = (A, \mu_A, \nu_A)$ étiqueté par E_A et attribué par $T_A \equiv D_A \rightarrow V_A$. Soit $\Omega = (\omega_1, \ldots, \omega_n)$ l'ensemble des classes.

Un modèle est défini par un triplet (C, Σ, Π) où :

- $C =< c_1, \ldots, c_m >$ avec $c_i \in (D_S \cup D_A) \times E_A$ représente les caractéristiques à extraire du graphe de la structure physique pour générer une configuration. Les composants de C seront notés $c =< c.d, c.e >$, où c est une caractéristique, $c.d \in (D_S \cup D_A)$ est un nom d'attribut et $c.e \in E_A$ est une étiquette d'arc.

- $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ où $\sigma_i \in \{0,1\}^m$ est un sélecteur désignant quelles caractéristiques sont déterminantes pour la classe ω_i.
- $\Pi = \{\pi_1, \ldots, \pi_n\}$ où $\pi_i = \{p_1^i, \ldots, p_{l_i}^i\}$ est un ensemble de patterns, c'est-à-dire des configurations de référence pour la mise en correspondance, valable pour la classe ω_i.

Remarquons que C est statique (fixé par l'application) alors que Σ et Π évoluent dynamiquement conformément à l'algorithme d'apprentissage.

L'exemple suivant présente un modèle qui pourrait convenir au document de l'exemple précédent.

Exemple 2: modèle
Le modèle se définit par rapport à tout objet s à classer. Les caractéristiques énumérées dans la partie statique n'ont de sens que par rapport à cet objet. Voici un modèle (C, Σ, Π) qui contient en partie dynamique une seule classe *auteur* représentée par un seul pattern $p_1^{auteur} \in \pi_{auteur}, \pi_{auteur} \in \Pi$ et le sélecteur de caractéristiques correspondant $\sigma_{auteur} \in \Sigma$.

Partie statique

$$C = \begin{bmatrix} < Type, ID > \\ < Type, V_{IG} > \\ < Fonte, ID > \\ < CompF, V_{IG} > \end{bmatrix}$$

Partie dynamique uniquement la classe *auteur* :

- un unique pattern qui est un quadruplet de valeurs correspondant aux caractéristiques du quadruplet C:

$$p_1^{auteur} = \begin{bmatrix} blocDeTexte \\ blocDeTexte \\ moyen \\ = \end{bmatrix}$$

  Le premier composant de p_1^{auteur} est la valeur *blocDeTexte* qui correspond à la première caractéristique $<Type,ID>$ de C : cela signifie que l'objet représentatif de la classe *auteur* sur lequel a été extraite le pattern était de type *blocDeTexte*. Le dernier composant de p_1^{auteur} est la valeur $=$, ce qui signifie que l'objet avait la même taille de fonte que son voisin inférieur gauche.

- un sélecteur des caractéristiques de C :

$$\sigma_{auteur} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

  Le 1er et le 3ème élément de σ_{auteur} ont la valeur 1, ce qui signifie que les caractéristiques pertinentes pour la classe *auteur* sont le type et la fonte de l'élément. Etant donné l'unique pattern représentatif de la classe *auteur*, tous les éléments de type *blocDeTexte* et ayant une fonte de la classe *moyen* seront attribués à la classe *auteur*.

4.4.4 Extraction de caractéristiques

L'extraction de caractéristiques a pour but de modéliser un objet par les caractéristiques qui sont pertinentes pour sa classification. Nous allons maintenant décrire la procédure d'extraction de caractéristiques sur un objet d'une structure physique représentée par un graphe étiqueté et attribué.

Soit une structure physique décrite par un graphe étiqueté et attribué (Σ_S, Σ_A) avec :

- $\Sigma_S = (S, \mu_S, \nu_S)$ étiqueté par E_S et attribué par $T_S \equiv D_S \rightarrow V_S$ et
- $\Sigma_A = (A, \mu_A, \nu_A)$ étiqueté par E_A et attribué par $T_A \equiv D_A \rightarrow V_A$.

Soit un modèle (C, Σ, Π) avec $C =< c_1, \ldots, c_m >$. L'extraction des caractéristiques de l'objet $s \in S$ produit une configuration $< \gamma_1, \ldots, \gamma_m > \in (V_S \cup V_A)^m$ déterminée par la fonction χ définie de la manière suivante :

$$\gamma_i = \chi(s, c_i) \quad = \quad \begin{cases} \nu_S(s')(c_i.d) & \text{si il existe } s' \text{ tel que } c_i.e \in \mu_A(s, s') \\ & \text{et si } c_i.d \in dom\ \nu_S(s'), \\ \nu_A(s, s')(c_i.d) & \text{si il existe } s' \text{ tel que } c_i.e \in \mu_A(s, s') \\ & \text{et si } c_i.d \in dom\ \nu_A(s, s'), \\ \textit{indéfini} & \text{sinon.} \end{cases}$$

Par extension, $\chi(s, C)$ désigne la configuration de l'objet s extraite d'après C et *dom* désigne le domaine de définition.
L'extraction de caractéristiques est illustrée par l'exemple suivant.

Exemple 3: extraction
Extraction de caractéristiques sur l'objet s_1 de l'exemple 1 d'après le quadruplet C du modèle de l'exemple 2.

$$\chi(s_1, C) = \begin{bmatrix} \chi(s_1, < Type, ID >) \\ \chi(s_1, < Type, V_{IG} >) \\ \chi(s_1, < Fonte, ID >) \\ \chi(s_1, < CompF, V_{IG} >) \end{bmatrix} = \begin{bmatrix} blocDeTexte \\ filet \\ moyen \\ indéfini \end{bmatrix}$$

4.4.5 Classification

Dans ce point nous allons décrire l'algorithme qui associe un objet à une des classes du modèle en interprétant la configuration de l'objet.

Soit un modèle (C, Σ, Π) avec :

- $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$,
- $\Pi = \{\pi_1, \ldots, \pi_n\}$ et
- $\pi_i = \{p_1^i, \ldots, p_{l_i}^i\}$.

Soit γ la configuration d'un objet s extraite par rapport à C. Soit $\Omega = \{\omega_1, \ldots, \omega_n\}$ l'ensemble des classes. γ "matche" la classe ω_i et on note $\gamma \otimes \omega_i$ si et seulement si il existe $p \in \pi_i$ tel que $\gamma \diamond \sigma_i = p \diamond \sigma_i$, où l'opérateur de sélection $\diamond$ est défini comme suit : supposons $\gamma =< \gamma_1, \ldots, \gamma_m >$ et $\sigma =< \sigma_1, \ldots, \sigma_m >$, alors $\gamma \diamond \sigma =< x_1, \ldots, x_m >$ est défini par :

$$x_i = \begin{cases} \gamma_i & \text{si } \sigma_i = 1, \\ null & \text{si } \sigma_i = 0. \end{cases}$$

Trois cas peuvent se présenter :

- il existe un et un seul ω_i tel que $\gamma \otimes \omega_i$: s est classé dans ω_i.
- il n'existe aucun ω_i tel que $\gamma \otimes \omega_i$: s ne peut pas être classé de manière univoque et une procédure d'apprentissage par extension est nécessaire.
- il existe ω_1, ω_2 avec $\omega_1 \neq \omega_2$ tel que $\gamma \otimes \omega_1$ et $\gamma \otimes \omega_2$: s ne peut pas être classé et une procédure d'apprentissage par spécialisation est nécessaire.

Dans l'exemple suivant nous classons un des objets de la structure physique de l'exemple 1, conformément au modèle de l'exemple 2.

Exemple 4: classification
La classification consiste à faire une sélection sur la configuration de l'objet à classer et à la comparer avec tous les patterns du modèle sur lesquels on a également opéré une sélection. A la seule classe du modèle de l'exemple 2 est associé un sélecteur dont la 1ère et la 3ème composante sont à 1. Ainsi la configuration $\chi(s_1, C)$ de l'exemple 3 est attribuée à la classe *auteur* :

$$\sigma_{auteur} \diamond \chi(s_1, C) = \sigma_{auteur} \diamond p_1^{auteur}$$

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ filet \\ moyen \\ indéfini \end{bmatrix} = \begin{bmatrix} blocDeTexte \\ null \\ moyen \\ null \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ blocDeTexte \\ moyen \\ = \end{bmatrix}$$

4.4.6 Apprentissage

2(CREM) est douée d'apprentissage incrémental qui se fait par extension et spécialisation du modèle. L'algorithme responsable de l'apprentissage est décrit dans ce dernier point.

Soit un modèle $M = (C, \Sigma, \Pi)$ et soit une configuration γ associée à un objet s extraite selon C. Il s'agit d'étendre le modèle en $M' = (C, \Sigma', \Pi')$ de manière à ce que γ "matche" $\omega_0 \in \Omega$ et seulement ω_0.

$b_{[i]}$ est un sélecteur de caractéristiques dont seule la $i$ème composante vaut 1.

Algorithme d'apprentissage :

`SI` $\neg(\gamma \otimes \omega_0)$ `ALORS`

ajouter γ à π_0
chercher un ensemble de caractéristiques dicriminantes $\{c_{l_1}, \ldots, c_{l_r}\}$
tel que $\gamma \diamond (\sigma_0 + \sum_{i=1}^{r} b_{[l_i]}) \neq p \diamond (\sigma_0 + \sum_{i=1}^{r} b_{[l_i]})$
pour tout $p \in \pi_k$ et pour tout $\pi_k \in \Pi$ avec $k \neq 0$
remplacer σ_0 par $\sigma_0 + \sum_{i=1}^{r} b_{[l_i]}$

POUR toute classe $\omega_k \neq \omega_0$ tel que $\gamma \otimes \omega_k$
chercher une caractéristique discriminante c_i
tel que $\gamma \diamond (\sigma_k + b_{[i]}) \neq p \diamond (\sigma_k + b_{[i]})$ pour tout $p \in \pi_k$
remplacer σ_k par $\sigma_k + b_{[i]}$

Dans l'exemple final, nous allons simuler un scénario complet d'apprentissage incrémental. Le système propose une classification conforme à son modèle. Puis, un utilisateur corrige le système en indiquant pour les objets mal classés la classe avec laquelle ces objets devraient "matcher". L'interaction est répétée jusqu'à ce que l'utilisateur juge que tous les objets du document analysé sont classés correctement.

Exemple 5: Apprentissage

Nous allons classer tous les objets de l'exemple 1. Nous reprenons le modèle de l'exemple 2 mais avec un sélecteur de caractéristiques pour la classe *auteur* dont toutes les composantes sont à 0. Ainsi on peut mieux illustrer le comportement du système lorsqu'il n'a pratiquement pas de connaissance. Nous simulerons les interventions de l'utilisateur; à chaque intervention de l'utilisateur, nous montrerons l'état du modèle avec les dernières modifications représentées en gras.

Classification initiale
Le sélecteur de caractéristique de l'unique classe *auteur* a toutes ses composantes à 0. Cela signifie qu'aucune caractéristique n'est considérée pour déterminer l'appartenance d'un objet à la classe et que tout objet, quel que soit l'ensemble de patterns de la classe, est assimilé à cette classe.

Première intervention de l'utilisateur
L'utilisateur fait une première correction en attribuant la classe *titre* à l'objet s_1 au lieu de la classe *auteur* choisie par le système; il en découle les changements suivants sur le modèle:

spécialisation adaptation de la classe *auteur* en modifiant le sélecteur de caractéristiques σ_{auteur} de manière à ce que s_1 ne puisse plus y être classé: le pattern représentant la classe *auteur* et la configuration de s_1 ont des valeurs différentes pour les caractéristiques $<Type, V_{IG}>$ et $<CompF, V_{IG}>$ qui sont les 2ème et 4ème composants de C, le système choisit donc une de ces caractéristiques et met la valeur 1 comme composant correspondant du sélecteur. Le choix de la caractéristique est déterminé par une heuristique qui sera présentée dans le point suivant.

extension ajout de la classe *titre*: la configuration de s_1 devient le pattern de la classe et le sélecteur est construit de manière à ce qu'aucun pattern du modèle ne puisse être assimilé à cette classe. Là aussi le système a le choix entre les caractéristiques $<Type, V_{IG}>$ et $<CompF, V_{IG}>$: le sélecteur ne contiendra donc que des valeurs 0, sauf pour le 2ème ou le 4ème composant (en fonction de l'heuristique adoptée).

Partie statique du modèle :

$$C = \begin{bmatrix} < Type, ID > \\ < Type, V_{IG} > \\ < Fonte, ID > \\ < CompF, V_{IG} > \end{bmatrix}$$

Partie dynamique :

- classe *auteur*

$$p_1^{auteur} = \begin{bmatrix} blocDeTexte \\ blocDeTexte \\ moyen \\ = \end{bmatrix} \quad \sigma_{auteur} = \begin{bmatrix} 0 \\ \mathbf{1} \\ 0 \\ 0 \end{bmatrix}$$

- **classe *titre***

$$\boldsymbol{p_1^{titre}} = \begin{bmatrix} \mathbf{blocDeTexte} \\ \mathbf{filet} \\ \mathbf{moyen} \\ \mathbf{indéfini} \end{bmatrix} \quad \boldsymbol{\sigma_{titre}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

A chaque changement du modèle, le système reclasse les objets du document traité.

- s_1 est classé *titre* :

$$\sigma_{auteur} \diamond \chi(s_1, C) \neq \sigma_{auteur} \diamond p_1^{auteur}$$

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ filet \\ moyen \\ indéfini \end{bmatrix} = \begin{bmatrix} null \\ filet \\ null \\ null \end{bmatrix} \neq \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ blocDeTexte \\ moyen \end{bmatrix}$$

$$\sigma_{titre} \diamond \chi(s_1, C) = \sigma_{titre} \diamond p_1^{titre}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ filet \\ moyen \\ indéfini \end{bmatrix} = \begin{bmatrix} null \\ null \\ null \\ indéfini \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ filet \\ moyen \\ indéfini \end{bmatrix}$$

- s_2 est classé *auteur* et *titre* :

$$\sigma_{auteur} \diamond \chi(s_2, C) = \sigma_{auteur} \diamond p_1^{auteur}$$

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \diamond \begin{bmatrix} filet \\ blocDeTexte \\ indéfini \\ indéfini \end{bmatrix} = \begin{bmatrix} null \\ blocDeTexte \\ null \\ null \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ blocDeTexte \\ moyen \\ = \end{bmatrix}$$

$$\sigma_{titre} \diamond \chi(s_2, C) = \sigma_{titre} \diamond p_1^{titre}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \diamond \begin{bmatrix} filet \\ blocDeTexte \\ indéfini \\ indéfini \end{bmatrix} = \begin{bmatrix} null \\ null \\ null \\ indéfini \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \diamond \begin{bmatrix} blocDeTexte \\ filet \\ moyen \\ indéfini \end{bmatrix}$$

- s_3 est classé *titre* :

$$\sigma_{auteur} \diamond \chi(s_3, C) \neq \sigma_{auteur} \diamond p_1^{auteur}$$

$$\begin{bmatrix}0\\1\\0\\0\end{bmatrix} \diamond \begin{bmatrix}blocDeTexte\\ind\acute{e}fini\\moyen\\ind\acute{e}fini\end{bmatrix} = \begin{bmatrix}null\\ind\acute{e}fini\\null\\null\end{bmatrix} \neq \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \diamond \begin{bmatrix}blocDeTexte\\blocDeTexte\\moyen\end{bmatrix}$$

$$\sigma_{titre} \diamond \chi(s_3, C) = \sigma_{titre} \diamond p_1^{titre}$$

$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \diamond \begin{bmatrix}blocDeTexte\\ind\acute{e}fini\\moyen\\ind\acute{e}fini\end{bmatrix} = \begin{bmatrix}null\\null\\null\\ind\acute{e}fini\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \diamond \begin{bmatrix}blocDeTexte\\filet\\moyen\\ind\acute{e}fini\end{bmatrix}$$

- s_4 est classé également *titre* puisqu'il a la même configuration que s_3.

Deuxième intervention de l'utilisateur
L'utilisateur corrige le système en classant l'objet s_2 dans la classe *souligneur* (s_2 avait été classé *auteur* et *titre* par le système). Le système adapte la partie dynamique du modèle en conséquence :

- classe *auteur*

$$p_1^{auteur} = \begin{bmatrix}blocDeTexte\\blocDeTexte\\moyen\\=\end{bmatrix} \quad \sigma_{auteur} = \begin{bmatrix}\mathbf{1}\\1\\0\\0\end{bmatrix}$$

- classe *titre*

$$p_1^{titre} = \begin{bmatrix}blocDeTexte\\filet\\moyen\\ind\acute{e}fini\end{bmatrix} \quad \sigma_{titre} = \begin{bmatrix}\mathbf{1}\\0\\0\\1\end{bmatrix}$$

- **classe *souligneur***

$$\boldsymbol{p_1^{souligneur}} = \begin{bmatrix}\mathbf{filet}\\\mathbf{blocDeTexte}\\\mathbf{ind\acute{e}fini}\\\mathbf{ind\acute{e}fini}\end{bmatrix} \quad \boldsymbol{\sigma_{souligneur}} = \begin{bmatrix}\mathbf{1}\\\mathbf{0}\\\mathbf{0}\\\mathbf{0}\end{bmatrix}$$

Dans le nouveau classement du système, s_1, s_3, et s_4 sont attribués à la classe *titre* alors que l'objet s_2 est attribué à la classe *souligneur*.

troisième intervention de l'utilisateur
L'utilisateur corrige le système en classant l'objet s_3 dans la classe *paragraphe*. Le système adapte le modèle en conséquence :

- classe *auteur*

$$p_1^{auteur} = \begin{bmatrix}blocDeTexte\\blocDeTexte\\moyen\\=\end{bmatrix} \quad \sigma_{auteur} = \begin{bmatrix}1\\1\\0\\0\end{bmatrix}$$

- classe *titre*

$$p_1^{titre} = \begin{bmatrix} blocDeTexte \\ filet \\ moyen \\ indéfini \end{bmatrix} \quad \sigma_{titre} = \begin{bmatrix} 1 \\ 0 \\ \mathbf{1} \\ 1 \end{bmatrix}$$

- classe *souligneur*

$$p_1^{souligneur} = \begin{bmatrix} filet \\ blocDeTexte \\ indéfini \\ indéfini \end{bmatrix} \quad \sigma_{souligneur} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- **classe *paragraphe***

$$\boldsymbol{p_1^{paragraphe}} = \begin{bmatrix} \mathbf{blocDeTexte} \\ \mathbf{indéfini} \\ \mathbf{petit} \\ \mathbf{indéfini} \end{bmatrix} \quad \boldsymbol{\sigma_{\text{paragraphe}}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix}$$

Cette fois le système classe tous les objets correctement.

4.5 Choix des caractéristiques

Les caractéristiques pertinentes pour une classe sont déterminées par *2(CREM)* : elle est amenée à choisir des caractéristiques pour une classe lors de sa création et lors de sa spécialisation.

Dans les deux situations, il peut exister plusieurs caractéristiques ou ensembles de caractéristiques possibles dont le choix va influencer la qualité du modèle. Un bon ensemble de caractéristiques pour une classe se distingue par les deux critères suivants :

Critère de discrimination. Soit D le nombre d'éléments n'appartenant pas à la classe et ayant la même valeur qu'un pattern de la classe pour l'ensemble de caractéristiques évalué. Un bon ensemble de caractéristiques sur le plan de la discrimination est un ensemble pour lequel D est petit.

Critère d'homogénéité. Soit H le nombre d'ensembles de valeurs différentes au sein des membres de la classe pour l'ensemble des caractéristiques évalué. Un bon ensemble de caractéristiques sur le plan de l'homogénéité est un ensemble pour lequel H est petit.

Dans la phase initiale de l'apprentissage du modèle, on dispose de très peu d'informations et on ne peut garantir le choix de bonnes caractéristiques. Par exemple, lorsqu'on choisit la première caractéristique qui permettra de discriminer les deux premières classes, on ne dispose que d'un unique membre de chaque classe. Si la caractéristique choisie se révèle mauvaise sur le plan de l'homogénéité pour l'une des

classes, la conséquence en sera que les membres de la classe ne seront que rarement reconnus et que la classe devra sans cesse être étendue par l'ajout de patterns. Le système ne prévoit pas la suppression de caractéristiques. C'est pourquoi nous avons prévu la possibilité de remettre en cause les choix réalisés dans la phase initiale de l'apprentissage en recalculant un nouvel ensemble de caractéristiques pour une classe. La solution idéale serait d'essayer toutes les combinaisons de caractéristiques et de choisir celle qui minimise le nombre de patterns nécessaires (plus il y de patterns moins la classe est homogène) tout en n'acceptant que les membres de la classe. Malheureusement on assisterait à une explosion combinatoire. Nous avons implémenté un algorithme qui choisit un ensemble de caractéristiques pour une classe en fonction des patterns de la classe, des membres de la classe et des non membres de la classe. D'une part il évite l'explosion combinatoire par une classification des caractéristiques selon H et D et d'autre part il étend la méthode telle qu'elle est décrite dans la section *Formalisation de* 2(CREM) du chapitre précédent : en effet, l'apprentissage ne se fait plus seulement en fonction d'une correction et de l'état courant du modèle, mais également en fonction de configurations. La classe de tous les objets correspondant aux configurations prises en compte a été donnée ou confirmée par l'opérateur.

Il y a différentes manières d'intégrer cet algorithme au système. Nous en parlons dans le chapitre 5.

4.6 Propriétés de *2(CREM)*

2(CREM) a les particularités suivantes : elle est douée d'apprentissage incrémental et son modèle prend en compte une organisation de l'information à deux dimensions. Ces propriétés sont deux points forts; elles font de *2(CREM)* une méthode souple, bien adaptée à la reconnaissance de documents à structure complexe.

Dans cette section nous allons discuter de deux thèmes qui révèlent des extensions souhaitables : la convergence du modèle et les interactions homme-machine. A noter qu'un troisième élément suscite une extension : la sensibilité aux choix des caractéristiques discriminantes initiales. En effet, le choix d'une caractéristique peu adéquate dans la phase initiale de l'apprentissage du modèle aura des conséquences sur la convergence. Ce point n'a pas été développé dans cette section mais dans le chapitre suivant où diverses solutions sont proposées pour chacune des applications.

4.6.1 Convergence du modèle

2(CREM) est douée d'apprentissage incrémental : le modèle est appris en cours d'exploitation sur la base de corrections externes au système. Au départ, le modèle ne reconnaît aucun objet et dans l'idéal, on aimerait qu'après une exploitation plus

ou moins longue il reconnaisse pratiquement tous les objets rencontrés. On dira que la méthode *converge* si pour un ensemble fini d'objets et moyennant un nombre de corrections fini, elle arrive toujours à construire un modèle capable de classer correctement tous les objets de l'ensemble. Si le choix de caractéristiques de la partie statique est pertinent, notre méthode converge. Cette affirmation ne découle pas d'une preuve, mais de la constatation que pour deux objets qui n'appartiennent pas à la même classe, il existe toujours une caractéristique qui permet de les distinguer. Dans le pire des cas, chaque classe utilise toutes les caractéristiques et chaque objet est représenté dans le modèle par sa propre configuration (devenue pattern de la classe de l'objet). Dans la pratique, notre méthode ne converge pas pour deux raisons :

- on ne dispose pas forcément des caractéristiques qui permettent de distinguer des objets de deux classes différentes;
- même si l'on dispose des caractéristiques nécessaires, on ne peut pas toujours essayer pour chaque classe tous les sous-ensembles de caractéristiques pour garantir le choix d'un sous-ensemble qui mènerait à la convergence. Une telle pratique provoquerait rapidement une explosion combinatoire c'est pourquoi nous avons développé l'heuristique évoquée dans le point précédent.

4.6.2 Interactions homme-machine

2(CREM) prévoit un type d'interaction homme-machine dont le scénario est extrêmement simple : le système propose une solution, l'humain effectue une correction, le système analyse la correction et adapte son modèle en conséquence, c'est-à-dire de manière à ce qu'il classe correctement l'objet sur lequel porte la correction. Pour l'instant l'utilisateur n'a pas la possibilité de confirmer une classe attribuée par le système. Il peut seulement valider la classification faite sur tout un document lorsqu'il estime qu'il n'y a plus d'erreur. Ce procédé n'est pas idéal car il comporte le risque d'un oubli d'erreur de la part de l'utilisateur, mais il est suffisant pour tester notre méthode.

Dans une première manière d'adapter le modèle, nous avions fait un choix qui avait des conséquences importantes sur le fonctionnement du système : le modèle futur était calculé uniquement en fonction du modèle présent et de la correction effectuée. On ne prenait pas en compte les corrections antérieures effectuées par l'opérateur.

Rappelons que l'apprentissage se fait par extension et spécialisation du modèle :

- extension : le système n'a pas associé la vraie classe à l'objet, la configuration de l'objet devient un pattern de la classe et on cherche une caractéristique discriminante de façon à ce qu'aucun pattern des autres classes ne puisse être assimilé à cette classe;
- spécialisation : le système a associé une fausse classe à l'objet, on ajoute une

caractéristique discriminante à la fausse classe de façon à ce qu'elle n'accepte plus l'objet.

L'option adoptée dans un premier temps (modèle futur calculé uniquement en fonction du modèle présent) avait un effet négatif sur le plan de l'interaction homme-machine : il n'y avait aucune garantie qu'un objet dont le classement avait été corrigé par l'opérateur soit par la suite toujours reconnu par le système. Ce mode de fonctionnement nuisait à la convivialité car il est très désagréable pour un opérateur d'effectuer plusieurs fois la même correction.

Nous avons remédié à ce problème par la mémorisation de toutes les corrections effectuées par l'opérateur. Le changement à opérer sur la méthode est minime. Dans la phase d'extension, au lieu de veiller à ce qu'aucun pattern des autres classes ne puisse être assimilé à la classe modifiée on s'assure qu'aucune configuration des objets ayant été corrigés et n'appartenant pas à la classe soit assimilé à la classe. Dans la phase de spécialisation, on s'assure que toutes les configurations des objets corrigés et appartenant à la classe, appartiennent toujours à la classe après l'ajout d'une caractéristique. Si tel n'est pas le cas, la configuration de l'objet en question devient un nouveau pattern de la classe.

Outre la mémorisation des corrections, nous avons imaginé diverses variantes pour prendre en compte les configurations (et non seulement les patterns) dans le choix des caractéristiques. Ces variantes ainsi que la mémorisation ont été développées et sont présentées dans la section 5.3 *Choix des caractéristiques*. Nous allons débattre de diverses autres solutions envisageables dans la conclusion.

Chapitre 5

Application de *2(CREM)* à la reconnaissance d'images de journaux

Le chapitre "Reconnaissance d'images de journaux : utilisation de méthodes simples" a montré que des méthodes simples basées sur des règles sont insuffisantes lorsqu'elles sont appliquées à des documents tels que les journaux. Nous avons appliqué *2(CREM)* à la reconnaissance de la structure physique de haut niveau et à la reconnaissance de la structure logique des journaux. Dans ce chapitre nous allons montrer que *2(CREM)* est beaucoup mieux adaptée à l'analyse de tels documents que les méthodes simples. Nous allons d'abord parler de la spécificité de l'analyse d'images de journaux. Un des traits caractéristiques communs aux documents à structure complexe est que l'information y est organisée de manière beaucoup moins linéaire : la place d'un objet dans la page ainsi que son voisinage sont porteurs de sens, c'est pourquoi dans la section 5.2 nous présenterons l'extraction et la représentation des caractéristiques d'un objet en insistant sur la description du voisinage de cet objet. Dans la section suivante nous parlerons du choix des caractéristiques discriminantes en cours d'apprentissage. *2(CREM)* a été testée sur trois applications différentes : la reconnaissance d'éléments structurants, la fusion des lignes en blocs et l'étiquetage logique. Ces applications seront présentées dans les sections 5.4 à 5.8.

5.1 Spécificité de l'analyse d'images de journaux

Les journaux font partie d'une catégorie de documents que nous avons appelés dans l'introduction *documents à structure complexe*. Ils partagent donc la spécificité de tous les documents complexes, à savoir un ordre de lecture non trivial. Dans ce point, nous allons aussi présenter d'autres aspects plus ou moins spécifiques aux journaux et qui caractérisent leur analyse : la variabilité intra-classe, la découpe en articles, l'utilisation d'objets structurants, les entrefilets, le mode d'intégration

des illustrations au texte et l'organisation des blocs dans la page. Nos expériences ont été menées sur le Los Angeles Times, c'est pourquoi dans un dernier point nous situerons l'analyse de ce quotidien par rapport aux aspects évoqués précédemment.

5.1.1 Ordre de lecture non trivial

Comme vu dans l'introduction, si l'on représente la structure physique de documents complexes sous une forme hiérarchique, on perd l'information qui permet de retrouver l'ordre de lecture. La position des objets les uns par rapport aux autres n'est pas interprétée trivialement, c'est pourquoi elle doit faire partie de la structure physique extraite de manière à donner lieu à une analyse dans l'étape de reconnaissance de la structure logique. Ceci implique donc d'une part un formalisme pour représenter les positions relatives entre objets, et d'autre part des outils d'analyse qui tiennent compte de ces positions relatives.

Par l'analyse de la structure physique, on arrive à retrouver des relations hiérarchiques entre objets, mais pas les relations d'ordre entre objets de même niveau. La structure physique d'un journal est donc représentée par un graphe qui a une structure d'arbre sous-jacente. De plus, un phénomène de *renvoi* complexifie le recouvrement de l'ordre de lecture. Dans le Los Angeles Times, les premières pages sont presque toujours le résumé ou le début d'articles développés sur d'autres pages. La portion d'article située en première page se termine alors par un renvoi spécifiant en principe la page et le *label* de la suite de l'article. De manière symétrique, la suite de l'article peut débuter par une référence au début de l'article. Le système de renvoi est utilisé par beaucoup de journaux.

5.1.2 Variabilité intra-classe

Du non déterminisme de l'ordre de lecture découle une plus grande variabilité des documents appartenant à une classe (par exemple, une classe regroupe les articles IEEE produits sur deux colonnes, une autre les exemplaires du *Los Angeles Times*). Le maquettiste a à sa disposition une deuxième dimension spatiale pour exprimer le contenu de son message et donc plus de choix. Plus de choix en matière de production signifie plus d'incertitude en matière de reconnaissance.

Avec les documents du type "journal", on assiste à plusieurs phénomènes de variabilité intra-classe avec des conséquences diverses sur la reconnaissance.

La diversité des mises en page implique la nécessité d'utiliser un plus grand échantillon d'apprentissage. Cependant, même avec un échantillon très grand, on n'a pas toutes les mises en pages possibles et une mise en page non prévue par le modèle peut survenir en tout temps. Cette situation plaide pour un apprentissage incrémental.

La diversité peut aussi provenir d'une évolution de la mise en page au cours du temps. Cette évolution se fait parfois de manière radicale et implique un changement

de modèle de reconnaissance, mais elle peut aussi se faire imperceptiblement et dans ce cas, seul un modèle incrémental peut y faire face.

Finalement, cette diversité existe aussi entre certaines pages du journal. Il y a des pages spécialisées comme l'éditorial ou le courrier des lecteurs qui mériteraient un modèle propre car elles ont une mise en page qui se différencie assez nettement de celle des autres pages.

5.1.3 Organisation en articles

Au haut niveau, l'objet principal des journaux est l'article. Les articles sont souvent regroupés par thème, mais peuvent être lus totalement indépendemment les uns des autres. On n'a donc pas un ordre de lecture total sur tous les objets du document, mais plusieurs ordres partiels sur des sous-ensembles d'objets. Une des principales tâches de l'analyse de journaux est la découpe en articles.

5.1.4 Utilisation des objets structurants

Les objets structurants sont des éléments non textuels qui aident le lecteur à retrouver la structure logique d'un document. Comme objets structurants les plus fréquents dans les pages de journaux, nous avons identifié les filets (segments de droite) et les cadres (pourtour d'un rectangle). Ces objets sont également utilisés pour structurer le message dans des documents simples, mais moins fréquemment. Dans les documents simples on trouve des filets dont le rôle est de souligner des titres ou d'isoler le corps du texte d'une note de bas de page. Les cadres servent en principe à mettre en évidence une illustration. Dans la mise en page de journaux, les objets structurants servent avant tout à aider à retrouver l'ordre de lecture : ils mettent en évidence, délimitent les articles ou donnent une certaine indépendance à une portion de document par rapport à ce qui l'entoure. La détection de ces objets ainsi que l'interprétation de leur rôle est une étape indispensable à l'analyse de pages de journaux.

5.1.5 Entrefilets

L'entrefilet est une portion de document qui appartient à un article mais qui conserve une certaine indépendance par rapport aux autres objets de l'article. Il peut être lu à n'importe quel moment de la lecture de l'article. Il peut s'agir, comme le montre les exemples de la figure 5.1, de citations en rapport avec l'article, d'un résumé d'une des idées de l'article ou d'un renseignement sur la nature de l'article. Les entrefilets sont repérables car ils sont rédigés dans une fonte différente de celle du texte qui les entoure, mais aussi parce qu'ils sont souvent isolés à l'aide d'objets structurants.

a) citation

b) résumé

c) nature de l'article

Figure 5.1: Trois types d'entrefilets.

5.1.6 Intégration des illustrations au contenu textuel

Les documents à structure complexe se différencient aussi par leur plus grande fantaisie dans le mode d'intégration des illustration dans le texte. La manière la plus simple d'insérer une illustration est de l'entourer d'un cadre rectangulaire et de lui faire occuper toute la largeur de la colonne de texte. Dans les journaux, on trouve des illustrations non contenues dans un cadre, non rectangulaires, voire non polygonales. De plus, comme le montre l'exemple de la figure 5.2, elles n'occupent pas forcément exactement la largeur d'une colonne, ayant une conséquence sur la forme des blocs de texte. La détection des illustrations et de leur pourtour nécessite des techniques souvent plus évoluées pour l'analyse d'images de journaux.

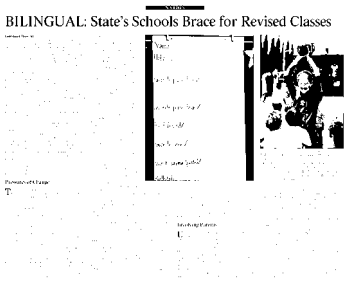


Figure 5.2: Illustration occupant plus que la largeur d'une colonne.

5.1.7 Organisation des blocs dans la page

Dans les documents simples, les blocs sont en principe rectangulaires. Dans les journaux, par contre, on trouve des blocs polygonaux et même des blocs non polygonaux. La segmentation de tels documents est une tâche nettement plus difficile. Sur l'extrait de page de journal de la figure 5.3 on voit des blocs non rectangulaires. Une telle structure de bloc est appelée structure en *mosaïque*.

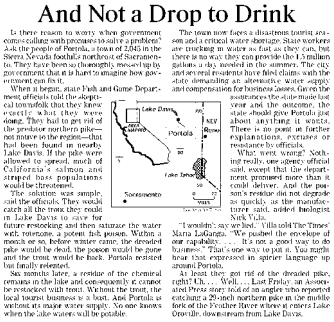


Figure 5.3: Organisation des blocs en mosaïque.

5.1.8 Quelques spécificités du Los Angeles Times

Le LAT présente comme la plupart des quotidiens une grande diversité dans sa mise en page. On y trouve une structure complexe avec des illustrations, des filets et des cadres ainsi que du texte en différentes polices et réparti en colonnes de largeurs inégales. Il s'agit donc d'un document *composite*. L'organisation des blocs est de type mosaïque, les blocs n'étant pas toujours de forme rectangulaire. Les illustrations peuvent être de formes diverses et non contenues dans des cadres. Certaines pages ont une structure qui leur est propre. Sur la figure 5.4, on voit une première page qui a une structure classique pour le LAT, avec une organisation en articles. La deuxième page représentée contient l'éditorial et le courrier des lecteurs. Ici la page n'est pas organisée en articles mais en lettres.

a) mise en page classique pour le LAT

b) mise en page propre à l'éditorial

Figure 5.4: Deux pages du LAT, deux modèles.

L'arbre sous-jacent de la structure physique a été décrit dans le chapitre 3. En annexe on trouve une description de la structure logique sous la forme d'une DTD XML. Le LAT est composé d'articles et d'illustrations. Un article comprend obligatoirement un titre et un contenu et peut comprendre aussi un résumé et un auteur. Dans le contenu d'un article on peut trouver des entrefilets.

5.2 Extraction et représentation des caractéristiques d'un objet

2(CREM) est une méthode de classification dont la première étape est l'extraction de caractéristiques sur les objets à reconnaître. Dans nos quatre applications, les objets à reconnaître sont respectivement des éléments structurants tels que les cadres et les filets (traits ou segments de droite), des couples de lignes de texte et des blocs. Nous avons extrait trois catégories de caractéristiques différentes :

- des caractéristiques propres à l'objet,
- des caractéristiques se rapportant aux objets voisins et
- des caractéristiques basées sur la comparaison entre l'objet et ses voisins ou entre les voisins.

Parmi les caractéristiques propres à l'objet, on trouve le type de l'objet et pour les objets textuels des caractéristiques se rapportant à la fonte (classe et taille de la fonte) et au nombre de lignes. Les couples d'objets sont comparés sur la base de leur position relative et pour les objets textuels, sur la base de leur fonte et de leur nombre de lignes.

Ces caractéristiques impliquant le voisinage d'un objet, nous avons besoin d'une part d'un formalisme pour décrire les relations de voisinage, et d'autre part d'un outil pour extraire les voisins d'un objet. Dans ce point nous parlerons donc de la description des relations de voisinage, de l'extraction de voisins mais aussi de l'extraction de caractéristiques telles que la classe de la taille de la fonte et la classe de la fonte.

5.2.1 Description des relations de voisinage

Allen [1] a étudié la comparaison de deux intervalles de temps. Il a recensé 13 situations différentes. Cette étude peut être étendue pour décrire la position relative entre deux objets dans le plan représentés par leur rectangle englobant. On obtient ainsi les 13x13 situations différentes que Azokly a décrites [3] (cf. figure 5.5).

Nous appellerons les configurations des colonnes 0,1,11 et 12 de l'image 5.5 des *voisinages disjoints sur l'axe des X* et les configurations des lignes 0,1,11 et 12 des

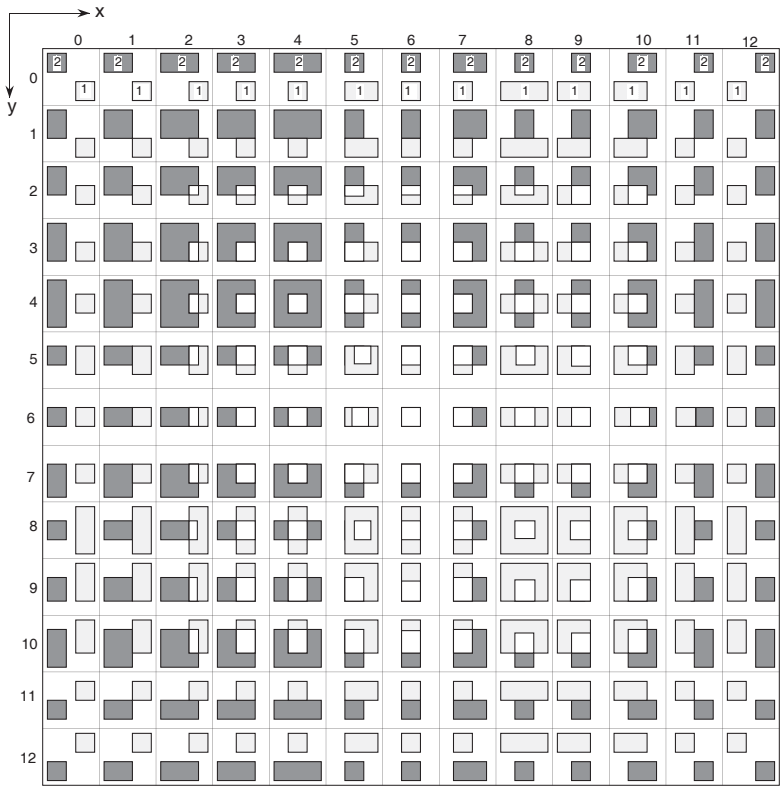


Figure 5.5: Topologies locales de Azokly : les 169 types de positions relatives entre deux objets.

voisinages disjoints sur l'axe des Y. Un voisinage entre deux rectangles est donc dit disjoint, si les projections des rectangles sur l'axe correspondant sont disjointes.

Un rectangle est décrit par les coordonnées de deux sommets opposés comme le montre la figure 5.6.

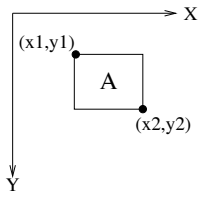


Figure 5.6: Description d'un rectangle : le rectangle A est décrit par le couple de coordonnées $<(x1, y1),(x2, y2)>$.

Nous définissons le *rectangle de voisinage vertical* de deux rectangles A et B par :

$$< (max(A.x1, B.x1), min(A.y2, B.y2)), (min(A.x2, B.x2), max(A.y1, B.y1)) >$$

Le *rectangle de voisinage horizontal* se définit de manière analogue. Si le voisinage des rectangles A et B est disjoint sur l'axe des X, le rectangle de voisinage n'existe pas et on obtient des coordonnées négatives. La figure 5.7 représente un rectangle de voisinage vertical.

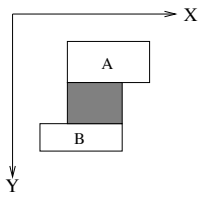


Figure 5.7: Le rectangle gris est le *rectangle de voisinage vertical* de A et B.

Dans nos applications, nous décrivons la position relative entre deux objets dans le plan (deux rectangles) par le *type*, l'*immédiateté* et la *proximité* de leur relation de voisinage. Le *type* d'un voisinage est décrit par l'une des 169 topologies de Azokly.

L'*immédiateté* décrit l'implication d'autres objets dans le voisinage. Un voisinage peut être *direct*, *semi-direct*, ou *indirect* : ces trois situations sont représentées par la figure 5.8.

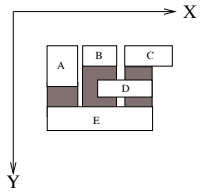


Figure 5.8: L'immédiateté de positions relatives sur l'axe des Y : <A,E> est un voisinage *direct*, <B,E> est un voisinage *semi-direct*, alors que <C,E> est un voisinage *indirect*.

La position relative des rectangles englobants des deux objets est dite :

- *directe* (selon un axe) si le rectangle de voisinage (selon cet axe) n'est intersecté par aucun autre rectangle,
- *semi-directe* (selon un axe), si elle n'est pas directe et s'il existe une droite perpendiculaire à cet axe qui traverse le rectangle de voisinage de ce même axe sans intersecter un rectangle tiers et
- *indirecte* (selon un axe), s'il n'existe pas une droite perpendiculaire à cet axe qui traverse le rectangle de voisinage de ce même axe sans intersecter un rectangle tiers. Cette désignation s'applique aussi aux couples de rectangles sans

rectangle de voisinage, c'est-à-dire avec un voisinage disjoint sur cet axe. Les deux rectangles représentés sur la figure 5.9 ont un voisinage indirect et pas de rectangle de voisinage.

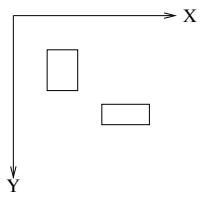


Figure 5.9: Voisinage indirect : les deux rectangles n'ont pas de rectangle de voisinage et ont donc un voisinage indirect.

Finalement, la *proximité* rend compte de l'éloignement entre deux objets en fonction de leur taille respective. Différentes fonctions décrivent la distance horizontale et verticale. Voici la définition des fonctions pour la distance horizontale :

- si les rectangles A et B sont disjoints sur l'axe des X, la fonction M_X est calculée :

$$M_X(A,B) = \frac{max(A.X1, B.X1) - min(A.X2, B.X2)}{max(A.X2 - A.X1, B.X2 - B.X1)}$$

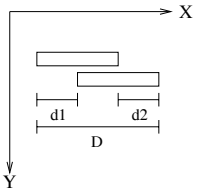
,

- si les rectangles sont non disjoints sur l'axe des X, les mesures $M1_X$ et $M2_X$ sont calculées :

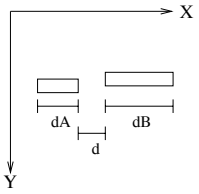
$$M1_X(A,B) = \frac{|A.X1 - B.X1|}{max(A.X2, B.X2) - min(A.X1, B.X1)}$$

$$M2_X(A,B) = \frac{|A.X2 - B.X2|}{max(A.X2, B.X2) - min(A.X1, B.X1)}$$

La figure 5.10 illustre ces fonctions.



a) $M1_X = d1/D$
$M2_X = d2/D$



b) $M_X = d/dB$

Figure 5.10: Proximité de deux rectangles.

Les fonctions qui décrivent la distance verticale se calculent de manière analogue.

5.2.2 Extraction de voisins

Notre outil d'extraction de voisins retourne quatre listes de voisins pour un objet : les voisins supérieurs, les voisins inférieurs, les voisins de gauche et les voisins de droite. Les voisins supérieurs d'un objet sont tous les voisins immédiats et intermédiaires sur l'axe des Y situés en-dessus de l'objet. Ils sont ordonnés par rapport à leur coordonnée $x1$ (de gauche à droite). Le même principe est appliqué pour les autres voisins. La figure 5.11 illustre les quatre listes de voisins extraites.

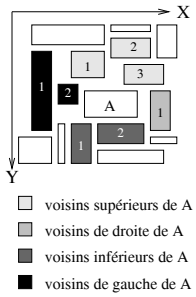


Figure 5.11: Les quatre listes de voisins de A.

Dans les applications, nous utilisons cet outil d'une part pour calculer la valeur de l'attribut d'objet "nombre de voisins supérieurs" ou "nombre de voisins inférieurs" et d'autre part pour accéder aux objets "voisin supérieur gauche", "voisin supérieur droit", "voisin inférieur gauche" et "voisin inférieur droit".

5.2.3 Description de la taille de la fonte

La fonte d'un bloc de texte est sans doute le plus précieux indice sur l'étiquette logique de ce bloc. Dans les documents bien structurés, à chaque fonte est associée une signification précise; de plus, la fonte est un indice plus facile à interpréter que d'autres indices comme les positions relatives. Nous avons extrait deux types d'attribut sur des objets textuels en rapport avec la fonte : la classe de la taille de la fonte et la famille de la fonte. Les tailles de fonte classées en catégories permettent par exemple de repérer les titres principaux. La classe de la fonte donne une indication plus fine : avec certains documents, on peut presque trouver une fonction qui apparie les fontes aux étiquettes logiques de blocs de texte. Dans ce point, nous allons décrire notre outil de classification de taille de fontes alors que la classification des fontes sera présentée dans le point suivant.

La taille de la fonte est un attribut qui se rapporte aux lignes de texte. Ce que nous appelons "taille de fonte" est en fait simplement la hauteur de la ligne de texte. La taille de fonte d'un mot est la hauteur de la ligne à laquelle il appartient et la taille de fonte d'un bloc est la hauteur de la plus haute ligne de texte.

Nous avons conçu une méthode de fixation automatique de seuils pour classer les lignes de texte dans des classes en fonction de la taille de leur fonte. Le nombre de classes peut être choisi en fonction de l'application et les limites de classes sont établies par apprentissage. Un histogramme *hist* de la hauteur des lignes de l'échantillon d'apprentissage est calculé. Cet histogramme est lissé de la manière suivante :

$$hist\_lisse(i) = \frac{\sum_{j=i-3}^{i+3} hist(j)}{7}.$$

On sélectionne ensuite les n maxima les plus élevés de l'histogramme lissé *hist_lisse*, n étant le nombre de classes désiré. Les limites seront les n minima suivant les n maxima sélectionnés. La figure 5.12 illustre la méthode d'apprentissage de la limite des classes.

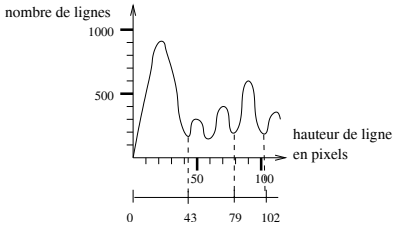


Figure 5.12: Répartition des caractères en 4 classes selon la taille de la fonte : 0-43, 44-79, 80-102 et 103 et plus.

Malheureusement, cette méthode s'est avérée mal adaptée à la reconnaissance d'images de journaux. La répartition des lignes de texte en fonction de la taille n'y est pas régulière : les petites fontes sont beaucoup plus représentées que les grandes fontes, ce qui a pour effet de concentrer les classes dans les petites fontes et de regrouper toutes les autres fontes (moyennes et grandes) dans une seule classe. Nous avons donc fixé nos seuils empiriquement à 44, 80 et 103.

La figure 5.13 est un exemple de la classification des lignes de texte en 3 catégories selon la taille de leur fonte. On remarque que les titres sont toujours dans la classe représentée en gris foncé (grosse taille) et le texte de base des articles est toujours dans la classe représentée en gris clair (petite taille). Dans la classe qui regroupe les lignes de taille intermédiaire, on trouve les résumés d'article.

5.2.4 Classification des fontes par appariement

Notre outil ne fait pas de reconnaissance de fontes : il est capable de regrouper tous les caractères de la même famille typographique (différenciation de fontes), mais ne peut identifier une fonte comme étant par exemple du *times-italique-gras-26pt*. La différenciation de fonte est suffisante pour faire de l'étiquetage logique car l'important

a) image d'une page du Los Angeles Times b) lignes de texte

Figure 5.13: Classification des lignes de texte en 3 catégories : en gris foncé les lignes de grande taille, en gris moyen les lignes de taille moyenne et en gris clair les lignes de petite taille.

n'est pas de connaître le nom d'une fonte, mais de savoir qu'une telle fonte a été utilisée pour écrire le nom de l'auteur d'un article ou le titre d'un paragraphe.

Notre outil est largement inspiré d'une étude typographique proposée par Lebourgeois [43]; il a été développé par David Berthold dans le cadre d'un travail de séminaire [12]. Le but est de regrouper tous les caractères de même taille, même inclinaison, même graisse et écrits dans la même fonte dans une seule famille typographique. L'algorithme se base sur la double hypothèse suivante :

- tous les caractères composant un mot appartiennent à la même famille typographique,
- chaque caractère d'une famille typographique possède un dessin unique au pixel près.

La deuxième hypothèse est vraie sur les images que nous avons analysées : des images idéales générées à partir de documents PDF. Elle ne se vérifie évidemment pas pour n'importe quelle image de documents.

L'idée de la méthode est illustrée par la figure 5.14.

Dans un premier temps, les caractères doivent être regroupés en classes de caractères identiques, appelées *classes de signes*. Pour éviter de devoir comparer pixel par pixel chaque paire de caractère, nous effectuons une pré-classification. Ainsi, la

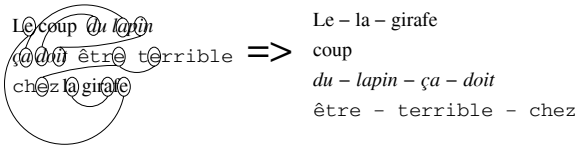


Figure 5.14: Les mots contenant au moins une forme de caractère commune sont classés dans la même famille typographique.

classification se fait en trois étapes : la pré-classification, la mise en correspondance et la classification.

pré-classification Un arbre de tri permet de classer les caractères d'après des critères simples comme la hauteur ou la largeur. La figure 5.15 illustre un tel arbre.

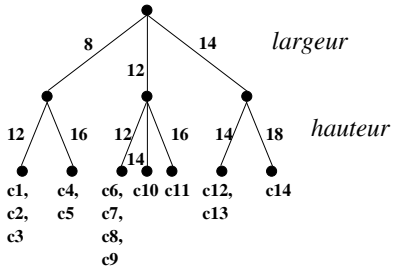


Figure 5.15: Pré-classification des caractères $c1$ à $c14$ selon les critères de la largeur et de la hauteur.

Les feuilles de l'arbre contiennent l'ensemble des caractères avec les mêmes valeurs sur les critères de tri.

mise en correspondance Les caractères contenus dans une feuille de l'arbre de pré-classification sont comparés entre eux pixels par pixels afin de les séparer en classes de signe.

classification Un graphe est construit dont les sommets sont les différentes classes de signe. Pour chaque noeud, on ajoute une arête vers les classes de signes dont au moins un signe est contenu dans un même mot qu'un signe de la classe représentée par le noeud. A chaque sous-graphe connexe correspondra ensuite une classe de fonte.

Cette méthode peut être douée d'apprentissage incrémental. Après la classification, l'arbre de tri est simplifié : on ne garde plus qu'un spécimen par classe de signe auquel on associe sa classe de fonte et son image. Cet arbre devient un modèle de classification de fontes qui peut être enrichi à chaque étape d'analyse.

Pour l'instant, notre méthode d'appariement de fontes ne fonctionne que sur des images idéales de document. La mise en correspondance que nous pratiquons est très rigide puisqu'elle exige que deux caractères soient identiques au pixel près pour appartenir à la même classe. Une mise en correspondance plus souple, par exemple basée sur les masques ternaires de Ingold [34], permettrait d'appliquer la méthode à des documents scannés. En effet, sur de tels documents, deux "a" générés par la même fonte ne sont que rarement identiques au pixel près. Par contre, l'utilisation des masques ternaires risquent d'assimiler un caractère simple comme le 'l' dans deux fontes différentes à la même fonte; de tels caractères ne devraient donc pas être considérés pour effectuer le recoupement.

5.3 Choix des caractéristiques

Le principe de l'algorithme de choix des caractéristiques par le système a été abordé dans le chapitre précédent.

Rappelons que dans *2(CREM)* on a deux ensembles de caractéristiques. Un ensemble commun à toutes les classes et déterminé lors de l'implémentation d'une application et un ensemble dynamique pour chaque classe, sous-ensemble de l'ensemble statique qui comprend les caractéristiques discriminantes pour la classe. L'ensemble dynamique est désigné par le sélecteur associé à chaque classe et est construit au cours de l'apprentissage du système. Le système est amené à choisir des caractéristiques discriminantes pour une classe lors de sa création ainsi que lors de sa spécialisation. Rappelons que la spécialisation d'une classe survient chaque fois qu'une configuration a été classée à tort dans la classe.

En situation de création de classe, le système choisit parmi les caractéristiques de l'ensemble statique un sous-ensemble minimal de manière à ce que aucun pattern du modèle n'ait les mêmes valeurs pour toutes les caractéristiques choisies que le premier pattern de la nouvelle classe.

Une configuration est classée à tort dans une classe lorsqu'un pattern de la classe a les mêmes valeurs que la configuration pour toutes les caractéristiques désignées par le sélecteur de la classe. Dans ce cas, le système choisit une caractéristique pour laquelle le pattern et la configuration n'ont pas la même valeur et il l'ajoute aux caractéristiques discriminantes de la classe.

Dans les deux situations, il peut exister plusieurs caractéristiques ou ensembles de caractéristiques possibles. Le but de l'algorithme de choix de caractéristiques présenté dans le chapitre précédent (section 4.5) est de déterminer, par une heuristique, la meilleure caractéristique (ou ensemble de caractéristiques) possible.

Nous avons implémenté différentes manières d'utiliser cet algorithme, qui combinées entre elles peuvent donner lieu à plusieurs réalisations de *2(CREM)* :

1. L'opérateur a la possibilité de valider une classification lorsqu'il considère que tous les éléments d'un document sont classés correctement. Le système constitue alors pour chaque classe l'ensemble de ses membres et de ses non membres et recalcule l'ensemble de caractéristiques discriminantes.

2. A chaque apprentissage, on prend en compte toutes les configurations classées par l'opérateur durant la session courante, le système les ayant toutes mémorisées.

3. L'opérateur a aussi la possibilité de fournir au système un ensemble de documents correctement classés constituant un échantillon d'apprentissage dont le système extrait pour chaque classe l'ensemble des membres et des non membres.

4. En cours d'apprentissage, le système fait des mises à jour locales (pour une classe) de l'ensemble des caractéristiques discriminantes. Si les configurations membres (respectivement non membres) de la classe n'ont pas été mémorisées, elles sont remplacées par les patterns du modèle décrivant (respectivement ne décrivant pas) la classe. La masse de connaissances est évidemment moins grande, mais ce procédé permet toutefois d'augmenter l'homogénéité sur les valeurs des caractéristiques discriminantes d'une classe et par là de diminuer le nombre de patterns nécessaires à la description de la classe. Une mise à jour locale est lancée pour une classe lorsque le nombre de ses patterns dépasse pour la première fois un seuil donné (nous avons empiriquement fixé le seuil à 6).

L'algorithme proposé n'est pas le résultat de recherches poussées et est probablement loin d'être idéal. Cependant il permet déjà d'améliorer sensiblement le modèle.

5.4 Démarche d'expérimentation

Le premier prototype de *2(CREM)* a été testé sur quatre applications différentes : la reconnaissance de filets, la reconnaissance de cadres, la fusion des lignes en blocs et l'étiquetage logique. Dans les sections suivantes, nous décrirons chacune des applications ainsi que les résultats des tests effectués. La présente section présente les applications de manière globale puis explique la stratégie de tests.

5.4.1 Description des applications

Le prototype correspond à la formalisation de base de *2(CREM)* à laquelle on a ajouté l'implémentation décrite au numéro 4 de la section 5.3. et basée sur l'algorithme présenté dans la section 4.5. Par cette implémentation on élimine l'effet

pervers du choix initial d'une mauvaise caractéristique, mais non le désagrément causé à l'opérateur par la modification de classes d'objets qu'il a lui-même attribuées. Chaque application sera décrite en utilisant le formalisme présenté dans la section 4.4. D'une application à l'autre, la partie variable de *2(CREM)* est le n-uplet C du modèle (C, Σ, Π) (ensemble de caractéristiques statiques parmi lesquelles on choisira les caractéristiques discriminantes pour une classe) et la structure physique extraite sur les documents à analyser. Rappelons qu'une structure physique est décrite par un graphe étiqueté et attribué (Σ_S, Σ_A) avec :

- $\Sigma_S = (S, \mu_S, \nu_S)$ étiqueté par E_S et attribué par $T_S \equiv D_S \rightarrow V_S$ et
- $\Sigma_A = (A, \mu_A, \nu_A)$ étiqueté par E_A et attribué par $T_A \equiv D_A \rightarrow V_A$.

Les ensembles E_S, E_A, D_S et D_A sont, pour une application donnée, communs à toutes les structures physiques. Ils déterminent les attributs qui seront extraits sur chaque document. Pour spécifier les applications, nous allons chaque fois énumérer le contenu de ces ensembles. Rappelons que E_S est l'ensemble des étiquettes associées aux sommets et qu'une étiquette correspond au type de l'objet représenté par le sommet. Les sommets des graphes de chaque structure physique correspondront aux objets du document dont le type est énuméré dans E_S. E_A est l'ensemble des étiquettes associées aux arcs (une étiquette correspond aux relations de voisinage qui existent entre deux objets). Les arcs des graphes correspondent aux paires d'objets du document qui ont une relation de voisinage énumérée dans E_A. D_S est l'ensemble des attributs extraits sur les objets, et D_A est l'ensemble des attributs extraits sur une paire d'objets.

Pour décrire une application, nous allons donc définir les ensembles E_S, E_A, D_S, D_A et C.

5.4.2 Stratégie de tests

Nous avons testé *2(CREM)* sur des pages du Los Angeles Times. De ces pages, nous avons exclu les éditoriaux qui ont une structure sensiblement différente des autres pages : ils ne sont pas composés d'articles, mais de lettres. De telles pages nécessiteraient un modèle séparé. Pour chaque application décrite dans le point précédent, nous avons effectué deux séries de tests. La première a pour but d'évaluer le processus d'apprentissage incrémental : elle met en correspondance le nombre de manipulations (corrections) effectuées par l'opérateur avec le taux de reconnaissance. La seconde série de tests est une évaluation du modèle construit par des tests "batch" (non interactifs).

Rappelons que le prototype de *2(CREM)* utilisé pour nos tests comprend l'extension proposée dans la section précédente *Choix des caractéristiques* qui permet au système de faire des remises à jour locales. Nous n'avons pas utilisé les trois autres extensions proposées : validation d'un document par l'opérateur, mémorisation des

configurations classées par l'opérateur et utilisation d'un ensemble de documents déjà reconnus pour la constitution du modèle. Il serait pourtant intéressant de comparer l'évolution du taux de reconnaissance avec et sans l'utilisation des extensions. C'est pourquoi dans l'application *étiquetage logique*, nous avons effectué une deuxième série de tests avec un deuxième prototype de *2(CREM)*. Ici nous avons combiné les extensions deux et trois : au cours de la session, l'apprentissage se fait en prenant en compte toutes les configurations classées par l'opérateur et à la fin de chaque session, les documents déjà reconnus sont fournis au système de manière à ce qu'il fasse une mise à jour de son modèle.

Évaluation du processus d'apprentissage

Pour évaluer le processus d'apprentissage nous testons le modèle après chaque manipulation de l'utilisateur. Le modèle est testé sur un minimum d'une quinzaine de pages qui constituent l'échantillon de test. Pour chaque page nous avons construit un *fond de vérité* ("groundtruth" en anglais) qui est le document reconnu (c'est-à-dire contenant la classe correcte de chaque objet). *2(CREM)* permet la constitution de fonds de vérité à peu de frais car il n'est pas nécessaire de les saisir manuellement. A l'aide d'un modèle rudimentaire (construit avec un minimum de manipulations), on opère une première classification qui est ensuite corrigée interactivement à travers l'interface *XMIllum*.

Dans un premier temps nous avons construit un modèle en sauvant son état après chaque manipulation. Nous avons obtenu ainsi autant de modèles que nous avons fait de manipulations. Un outil calcule ensuite l'évolution du taux de reconnaissance en regard des manipulations effectuées. Avec chaque modèle construit, il effectue une reconnaissance des pages de tests puis compare les résultats et les fonds de vérité.

Évaluation d'un modèle

Bien que *2(CREM)* n'ait pas été conçue pour de la reconnaissance entièrement automatique, nous avons évalué la pertinence des modèles construits par des tests "batch". Le modèle a été entraîné sur des pages différentes de celles de l'échantillon de test jusqu'à-ce qu'il atteigne une certaine stabilité dans les taux de reconnaissance, c'est-à-dire que de nouvelles manipulations n'améliorent pas sensiblement la classification. La stabilité peut se détecter en utilisant le même procédé que pour l'évaluation du processus d'apprentissage.

Soulignons que les résultats présentés sont liés au modèle utilisé. Un autre modèle construit par une personne différente ou non, donnera d'autres résultats, surtout si le nombre de classes et de caractéristiques est élevé. La différence ne réside pas tellement dans le taux de reconnaissance global (toute classe confondue), mais plutôt dans les classes qui sont confondues. Par exemple, pour l'application *étiquetage logique des blocs*, un modèle aura peut-être tendance à confondre les titres et les

sous-titres alors qu'un autre modèle confondra les résumés et le texte de base.

5.5 Reconnaissance de filets

Les filets sont, avec les cadres, les principaux objets structurant les pages de journaux. Dans l'application, nous cherchons à classer les filets en fonction de leur rôle de structuration.

Pour les filets, nous avons identifié trois rôles de structuration qui correspondent aux classes auxquelles un filet peut appartenir :

- Certains filets permettent de mettre en évidence un objet par rapport aux autres; nous les appellerons les *filets souligneurs*. La plupart du temps, ils indiquent un titre, mais ils peuvent aussi indiquer, comme le montre l'exemple de la figure 5.16, un nom de rubrique ou un sujet.
- Les filets appelés *séparateurs externes* délimitent les articles.
- Les filets appelés *séparateurs internes* donnent une certaine indépendance à une partie du contenu d'un article par rapport au reste du contenu. Il arrive fréquemment que les filets séparateurs internes existent par paire (voire triplet)(cf. exemple de la figure 5.16), alignés verticalement de part et d'autre d'une portion du document : nous appelons cette portion du document *entre-filet*. Le texte situé de part et d'autre des deux filets s'enchaîne comme si les filets et l'entrefilet n'existaient pas.

La figure 5.16 montre les différents rôles de structuration que peut avoir un filet.

Dans la suite de l'analyse, la connaissance du type des filets va être utile à divers stades.

Les filets sont utiles pour *l'étiquetage logique des blocs*. Un objet avec un filet souligneur situé en dessous de lui sera facilement reconnu comme étant un titre d'article, un titre de rubrique, un sujet d'article ou quelque chose de similaire et l'objet situé entre deux filets séparateurs comme entrefilet. De même les filets séparateurs externes permettent de désigner les objets situés en dessus d'eux comme étant des objets localisés à la fin des articles et les objets situés en dessous d'eux comme étant des objets localisés au début des articles, réduisant ainsi le spectre des possibilités.

Les filets séparateurs externes marquent la frontière entre deux articles et peuvent donc s'avérer d'une grande aide dans la *délimitation des articles*.

La connaissance du rôle d'un séparateur interne est indispensable, notamment pour détecter l'enchaînement qui existe entre les blocs de texte situés de part et d'autre d'un entrefilet, sous-problème du *recouvrement de l'ordre de lecture*.

a) b)

a) rôle d'un filet séparateur: (1), (2) et (3) sont des séparateurs internes alors que (4) est un séparateur externe.
b) rôle d'un filet souligneur: (1) met en évidence le nom de la rubrique et (2) met en évidence le sujet traité.

Figure 5.16: Rôles des filets.

5.5.1 Description de l'application

Dans l'application *reconnaissance de filets*, le graphe (Σ_S, Σ_A) de la structure physique est formé par $\Sigma_S = (S, \mu_S, \nu_S)$ et $\Sigma_A = (A, \mu_A, \nu_A)$ où :

$\mathbf{E_S}$ = {*filet, blocDeTexte, illustration, cadre*} est l'ensemble des types d'objet considérés;

$\mathbf{E_A} = \{ID, V_{ID}, V_{IG}, V_{SD}, V_{SG}\}$ est l'ensemble des types de positions relatives considérées; pour classer un objet *filet*, on va considérer l'objet lui-même (ID), son voisin inférieur droit (V_{ID}), son voisin inférieur gauche (V_{IG}), son voisin supérieur droit (V_{SD}) et son voisin supérieur gauche (V_{SG});

$\mathbf{D_S}$ = {*type, nbvs, nbvi, tailleDeFonte*} est l'ensemble des attributs extraits sur les objets, à savoir leur type, le nombre de leurs voisins supérieurs (*nbvs*), le nombre de leurs voisins inférieurs (*nbvi*) et la taille de la fonte des objets textuels ($tailleDeFonte$);

$$\begin{aligned}\mathbf{T_S} = \quad \{ & t_{type} : type && \rightarrow \quad \{filet, blocDeTexte, \\ & && \quad\ illustration, cadre\}, \\ & t_{nbvs} : nbvs && \rightarrow \quad \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\ & t_{nbvi} : nbvi && \rightarrow \quad \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\ & t_{tailleDeFonte} : tailleDeFonte && \rightarrow \quad \{petit, moyen, grand\}\}\end{aligned}$$

est le domaine des valeurs de chaque attribut de D_S;

$\mathbf{D_A} = \{position\}$ est une description de la position relative, attribut extrait sur un couple d'objets;

$\mathbf{T_A} = \{t_{position} : position \rightarrow (N \times N) \times (I \times I) \times (P \times P)\}$ est le domaine des valeurs de l'attribut *position* de D_A où $N = \{0, \ldots, 12\}$ représente le type de la position relative par rapport à un axe par les treize intervalles de Allen, $I = \{dir, midir, indir\}$ représente l'immédiateté de la relation (directe, semi-directe ou indirecte) par rapport à un axe et P représente la proximité de la relation par rapport à un axe. P correspond aux nombres réels si la relation est non disjointe et aux couples de nombres réels si la relation est disjointe.

Les classes à disposition sont :

$\mathbf{\Omega} = \{souligneur,\ séparateurInterne,\ séparateurExterne\}$.

La partie statique du modèle (C, Σ, Π) comprend la liste de caractéristiques suivante :

$$\begin{aligned}\mathbf{C} = \quad & \{< type, V_{IG} >, < type, V_{ID} >, < type, V_{SG} >, < type, V_{SD} >, \\ & < nbvs, ID >, < nbvi, ID >, < tailleDeFonte, V_{IG} >, \\ & < tailleDeFonte, V_{ID} >, < tailleDeFonte, V_{SG} >, \\ & < tailleDeFonte, V_{SD} >, < position, V_{SG} >, \\ & < position, V_{SD} >, < position, V_{IG} >, < position, V_{ID} >\},\end{aligned}$$

les caractéristiques extraites sur les objets à classer.

5.5.2 Tests et évaluation

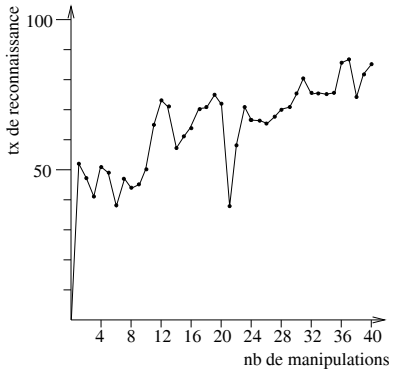


Figure 5.17: Évolution du modèle au cours de l'apprentissage : taux de reconnaissance par rapport au nombre de manipulations de l'opérateur.

La figure 5.17 montre l'évolution du modèle au cours de l'apprentissage en mettant en correspondance le nombre de manipulations de l'opérateur et le taux de recon-

naissance. Après une manipulation, on a déjà un taux de reconnaissance de 50%. Ça peut paraître curieux. En fait, la première correction (manipulation) consistait à classer un filet inconnu dans la classe "séparateur externe". Comme "séparateur externe" devient alors la seule classe du modèle et que le sélecteur de caractéristiques correspondant ne contient que des 0 (généralité maximale), tous les filets sont associés à la classe "séparateur externe". Dans l'échantillon de test, il y a 50% des filets qui sont des séparateurs externes, ce qui explique le résultat obtenu. A noter que les 22 pages testées contiennent une centaine de filets.

Globalement on remarque une progression dans le taux de reconnaissance, mais localement on assiste fréquemment à une décroissance. Ce phénomène survient lorsque l'on spécialise une classe : l'effet escompté est que la classe n'accepte plus les filets ne lui appartenant pas, l'effet de bord est que la classe rejette des filets lui appartenant et acceptés avant la manipulation. Vers 90% de reconnaissance, le modèle cesse de progresser : nous en déduisons que les caractéristiques extraites sont insuffisantes.

Les tests "batchs" ont été effectués sur 81 pages du Los Angeles Times. Le tableau 5.1 donne le taux de reconnaissance en spécifiant si les échecs sont le résultat d'une erreur, d'un manque de connaissance (colonne "inconnu") ou d'un conflit. On compte dans la colonne "conflit" les filets qui ont été assignés à plusieurs classes, dont la classe correcte; si aucune classe n'est trouvée, on incrémente la colonne "inconnu" et si on obtient une ou plusieurs classes fausses, on incrémente la colonne "erreur".

classes	nombre total	reconnus	inconnus	en conflit	mal classés
séparateurs externes	271	90%	9%	0%	1%
séparateurs internes	110	81%	18%	0%	1%
souligneurs	173	90%	5%	0%	5%

Tableau 5.1: Tests "batchs" pour la reconnaissance des filets effectués sur 81 pages du Los Angeles Times; les résultats sont exprimés en pourcentage du nombre total de filets d'un type donné.

On remarque que les taux de reconnaissance sont bons pour les filets séparateurs externes et souligneurs. Les filets séparateurs internes donnent de moins bons résultats. La cause de certains échecs est due à l'insuffisance des caractéristiques de base comme le montrent les tests d'apprentissage. Cependant, on peut attribuer une partie des échecs au manque de connaissance (situations non apparues dans l'échantillon d'apprentissage) et espérer qu'en mode interactif le modèle s'améliorerait.

5.6 Reconnaissance des cadres

Pour les cadres, nous avons identifié deux rôles de structuration correspondant aux classes auxquelles un cadre peut appartenir :

- Les cadres appelés *séparateurs externes* mettent en évidence un article par rapport aux autres. Ils sont parfois utilisés systématiquement pour encadrer certaines rubriques.
- Les cadres appelés *séparateurs internes* ont souvent le même rôle qu'une paire de filets et la portion de document qu'ils isolent est de même nature qu'un entrefilet.

L'utilisation du rôle des cadres pour l'étiquetage logique est comparable à celle des filets séparateurs internes et externes.

5.6.1 Description de l'application

Ici, le graphe (Σ_S, Σ_A) de la structure physique est formé par $\Sigma_S = (S, \mu_S, \nu_S)$ et $\Sigma_A = (A, \mu_A, \nu_A)$ où :

$\mathbf{E_S}$ = {*filet, blocDeTexte, illustration, cadre*} est l'ensemble des types d'objet considérés;

$\mathbf{E_A} = \{ID, V_{ID}, V_{IG}, V_{SD}, V_{SG}\}$ est l'ensemble des types de positions relatives considérées; pour classer un objet *cadre*, on va considérer l'objet lui-même (ID), son voisin inférieur droit (V_{ID}), son voisin inférieur gauche (V_{IG}), son voisin supérieur droit (V_{SD}) et son voisin supérieur gauche (V_{SG});

$\mathbf{D_S}$ = {*type, nbvs, nbvi, tailleDeFonte, structure*} est l'ensemble des attributs extraits sur les objets, à savoir leur type, le nombre de leurs voisins supérieurs (*nbvs*), le nombre de leurs voisins inférieurs (*nbvi*), la taille de la fonte des objets textuels ($tailleDeFonte$) et la structure des objets qui sont des cadres (*structure*), c'est-à-dire une indication sur le nombre de blocs contenus dans le cadre;

$$\begin{array}{lll} \mathbf{T_S} = \{t_{type} : type & \rightarrow & \{filet, blocDeTexte, \\ & & illustration, cadre\}, \\ t_{nbvs} : nbvs & \rightarrow & \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\ t_{nbvi} : nbvi & \rightarrow & \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\ t_{tailleDeFonte} : tailleDeFonte & \rightarrow & \{petit, moyen, grand\}, \\ t_{structure} : structure & \rightarrow & \{unibloc, multiblocs\}\} \end{array}$$

est le domaine des valeurs de chaque attribut de D_S;

$\mathbf{D_A}$ = {*position*} est une description de la position relative, attribut extrait sur un couple d'objets,

$\mathbf{T_A} = \{t_{position} : position \rightarrow (N \times N) \times (I \times I) \times (P \times P)\}$ est le domaine de l'attribut *position* de D_A où $N = \{0, \ldots, 12\}$ représente le type, I l'immédiateté et P la proximité de la relation.

Les classes à disposition sont :

$\mathbf{\Omega} = \{séparateurInterne,\ séparateurExterne\}$.

La partie statique du modèle (C, Σ, Π) est :

$$\begin{aligned}\mathbf{C} = \quad & \{< type, V_{IG} >, < type, V_{ID} >, < type, V_{SG} >, < type, V_{SD} >, \\ & < nbvs, ID >, < nbvi, ID >, < tailleDeFonte, V_{IG} >, \\ & < tailleDeFonte, V_{ID} >, < tailleDeFonte, V_{SG} >, \\ & < tailleDeFonte, V_{SD} >, < structure, ID >, \\ & < position, V_{SG} >, < position, V_{SD} >, < position, V_{IG} >, \\ & < position, V_{ID} >\},\end{aligned}$$

les caractéristiques extraites sur les objets à classer.

5.6.2 Tests et évaluation

Le nombre de cadres séparateurs externes étant nettement plus élevé que le nombre de cadres séparateurs internes, l'apprentissage converge rapidement.

Pour les tests "batchs" nous avons utilisé 114 pages du Los Angeles Times car le nombre de cadres par pages est très faible. Les résultats sont présentés dans le tableau 5.2.

classes	nombre total	reconnus	inconnus i	en conflit	mal classés
séparateurs externes	130	98%	0%	2%	0%
séparateurs internes	40	83%	2%	0%	15%

Tableau 5.2: Tests "batchs" pour la reconnaissance des cadres effectués sur 114 pages du Los Angeles Times; les résultats sont exprimés en pourcentage du nombre total de cadres d'un type donné.

5.7 Fusion des lignes de texte en blocs

Nous avons appliqué *2(CREM)* à de la segmentation de haut niveau. Dans le chapitre 3, nous avons vu qu'un des problèmes de segmentation les plus spécifiques aux documents à structure complexe était la détection des blocs de texte. Rappelons

qu'une colonne est la plus grande portion texte de document à l'intérieur de laquelle les lignes de texte sont alignées verticalement et l'ordre de lecture se fait trivialement de gauche à droite et de haut en bas. Les colonnes peuvent être composées de plusieurs blocs repérables à la taille de l'interligne.

Dans les documents à structure complexe, le nombre, la hauteur et la largeur des colonnes est variable, ce qui rend la détection des blocs beaucoup moins aisée. Comme la détection des lignes de texte de documents complexes est relativement similaire à celle de documents simples, il était naturel d'adopter une approche ascendante. Le chapitre 3 a montré les faiblesses d'une approche basée sur des règles fixes pour la fusion des lignes en blocs. L'idée de l'algorithme de fusion était d'analyser tous les couples de lignes voisines sur l'axe vertical et de décider s'ils devaient ou non fusionner en se référant à un ensemble de règles. Cette approche avait une double faiblesse : elle ne prenait pas en compte le contexte et nécessitait pour l'établissement des règles une énumération exhaustive de toutes les configurations possibles. La méthode des patterns appliquées à la fusion des lignes de texte est basée sur le même principe. Les objets à classer sont des couples de lignes de texte voisines sur l'axe vertical et les classes sont *fusion/pasDeFusion*. Le système constitue par apprentissage l'ensemble de toutes les configurations possibles. La méthode des patterns ne souffre pas des faiblesses de la méthode basée sur des règles fixes. La figure 5.18 compare le résultat de segmentation en blocs obtenu par les deux méthodes sur un exemple typique de structure en mosaïque. A noter que dans cette application, l'ensemble de caractéristiques discriminantes pour une classe n'est pas appris : il correspond pour les deux classes à l'ensemble C des caractéristiques statiques. Nous émettons l'hypothèse que dans le domaine physique, les caractéristiques discriminantes pour une classe varient moins d'une classe de documents à l'autre que dans le domaine logique.

a) méthode basée sur des règles

b) méthode basée sur les patterns

Figure 5.18: Fusion des lignes de texte en blocs.

5.7.1 Description de l'application

Le graphe (Σ_S, Σ_A) de la structure physique est formé par $\Sigma_S = (S, \mu_S, \nu_S)$ et $\Sigma_A = (A, \mu_A, \nu_A)$ où :

$\mathbf{E_S} = \{$*filet, ligneDeTexte, illustration, cadre*$\}$ est l'ensemble des types d'objet considérés;

$\mathbf{E_A} = \{ID, V_{ID}, V_{IG}, V_{SD}, V_{SG}\}$ est l'ensemble des types de positions relatives considérées; pour classer un objet *couple de lignes de texte*, on va considérer l'objet lui-même (ID), son voisin inférieur droit (V_{ID}), son voisin inférieur gauche (V_{IG}), son voisin supérieur droit (V_{SD}) et son voisin supérieur gauche (V_{SG});

$\mathbf{D_S} = \{$*type, nbvs, nbvi, positionInterne*$\}$ est l'ensemble des attributs extraits sur les objets, à savoir leur type, le nombre de leurs voisins supérieurs (*nbvs*), le nombre de leurs voisins inférieurs (*nbvi*) et la position relative des deux lignes de texte composant l'objet;

$$\begin{array}{lll} \mathbf{T_S} = \{t_{type} : type & \rightarrow & \{filet, ligneDeTexte, \\ & & illustration, cadre\}, \\ t_{nbvs} : nbvs & \rightarrow & \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\ t_{nbvi} : nbvi & \rightarrow & \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\ t_{positionInterne} : positionInterne & \rightarrow & (N \times N) \times (I \times I) \times \\ & & (P \times P)\} \end{array}$$

est le domaine de chaque attribut de D_S où $N = \{0, \ldots, 12\}$ représente le type, I l'immédiateté et P la proximité de la relation;

$\mathbf{D_A} = \{$*position*$\}$ est une description de la position relative, attribut extrait sur un couple d'objets;

$\mathbf{T_A} = \{t_{position} : position \rightarrow (N \times N) \times (I \times I) \times (P \times P)\}$ est le domaine de l'attribut *position* de D_A où $N = \{0, \ldots, 12\}$ représente le type, I l'immédiateté et P la proximité de la relation.

Les classes à disposition sont :

$\mathbf{\Omega} = \{$*fusion, pasDeFusion*$\}$.

La partie statique du modèle (C, Σ, Π) est :

$$\begin{array}{ll} \mathbf{C} = & \{< positionInterne, ID >, < type, V_{IG} >, < type, V_{ID} >, \\ & < type, V_{SG} >, < type, V_{SD} >, < nbvs, ID >, < nbvi, ID >, \\ & < position, V_{SG} >, < position, V_{SD} >, < position, V_{IG} >, \\ & < position, V_{ID} >\}, \end{array}$$

caractéristiques extraites sur les objets à classer.

5.7.2 Tests et évaluation

L'évolution du modèle au cours de l'apprentissage interactif est représenté par la figure 5.19.

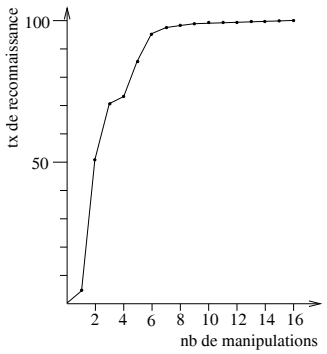


Figure 5.19: Évolution du modèle au cours de l'apprentissage : taux de reconnaissance par rapport au nombre de manipulations de l'opérateur.

Ici l'évolution est progressive : comme les caractéristiques pertinentes ne sont pas apprises, on n'assiste pas à l'effet de bord de la spécialisation. L'apprentissage illustré par la figure 5.19 a été mené sur une seule page de document car le nombre de couples de lignes par page est suffisament grand (environ 400) pour avoir des données représentatives. Par contre l'apprentissage du modèle utilisé pour conduire les tests "batchs" s'est fait sur l'échantillon d'apprentissage habituel qui comprend 22 pages du Los Angeles Times.

Pour rendre compte de la pertinence du modèle, nous avons également effectué des tests "batchs" sur 29 pages du Los Angeles Times. Ici la présentation des résultats diffère. Au lieu de compter le nombre de couples de lignes classés correctement, nous avons compté le nombre de blocs segmentés correctement. Les résultats sont très satisfaisants : nous atteignons 99.2% de réussite puisque sur 752 blocs, seuls six blocs ont été mal segmentés. Ces erreurs correspondent à des configurations non rencontrées durant l'apprentissage.

5.8 Étiquetage logique

Finalement, la méthode des patterns a été appliquée à l'étiquetage logique des blocs de texte. La variété des étiquettes est très grande sur des documents complexes tels que les journaux. Nous avons classé les blocs de texte en quatorze classes différentes : texte de base, titre, sous-titre, auteur, fonction de l'auteur, résumé (bref curriculum-vitae de l'auteur), source, légende, ancres (ancreVers et ancreDe), rubrique, sujet,

entre-filets (entrefiletTitre et entrefiletContenu). La figure 5.21 illustre une partie de ces classes sur une page complète du Los Angeles Times alors que des exemples des autres classes sont donnés par la figure 5.20.

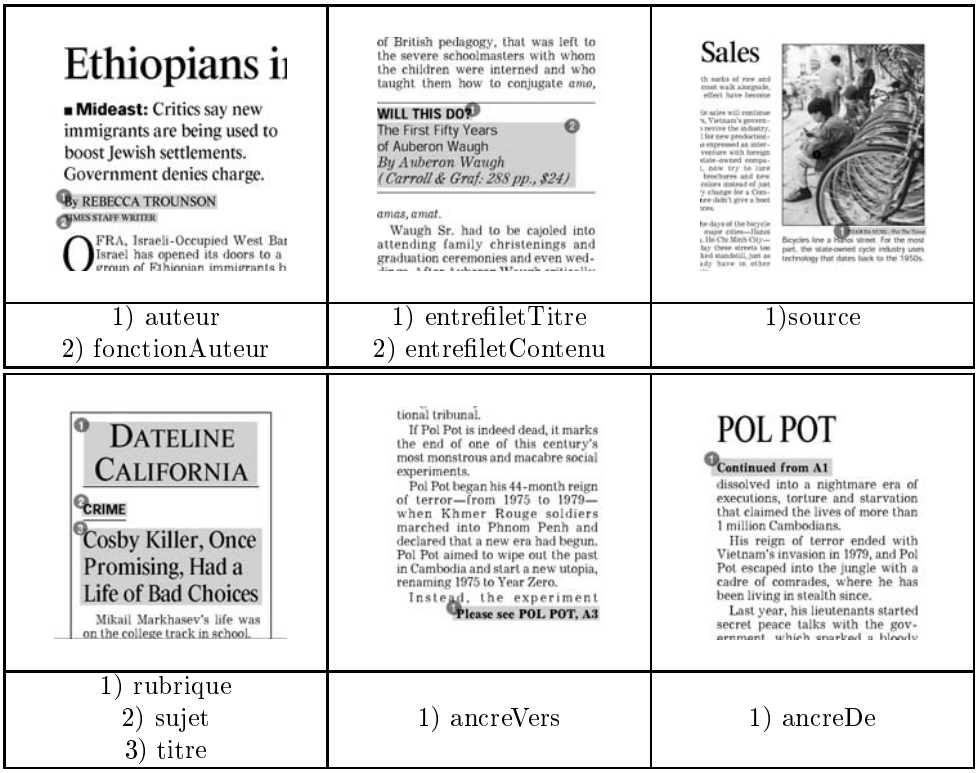


Figure 5.20: Classes de blocs de texte; à noter la relation entre l'*ancreVers* de la cinquième image et l'*ancreDe* de la dernière image.

La classe de la fonte est un indice précieux sur l'étiquette à attribuer à un bloc de texte. Malheureusement, la fonte utilisée pour éditer par exemple le texte de base peut être identique sur dix documents, puis changer. La méthode des patterns est particulièrement bien adaptée à ce genre de situation : la classe de la fonte reste une caractéristique déterminante, mais pour chaque nouvelle fonte, on a un pattern différent dans la classe *texteDeBase*. Dans le cas des titres, la classe de la fonte varie trop pour être une caractéristique déterminante, la taille de la fonte sera certainement préférée par le système.

Figure 5.21: Quelques étiquettes pour les blocs de texte d'une page du Los Angeles Times.

5.8.1 Description de l'application

Le graphe (Σ_S, Σ_A) de la structure physique est formé par $\Sigma_S = (S, \mu_S, \nu_S)$ et $\Sigma_A = (A, \mu_A, \nu_A)$ où :

$\mathbf{E_S}$ = {*filet, blocDeTexte, illustration, cadre*} est l'ensemble des types d'objet considérés;

$\mathbf{E_A} = \{ID, V_{ID}, V_{IG}, V_{SD}, V_{SG}\}$ est l'ensemble des types de positions relatives considérées; pour classer un objet *blocDeTexte*, on va considérer l'objet lui-même (*ID*), son voisin inférieur droit (V_{ID}), son voisin inférieur gauche (V_{IG}), son voisin supérieur droit (V_{SD}) et son voisin supérieur gauche (V_{SG});

$\mathbf{D_S}$ = {*type, nbvs, nbvi, tailleDeFonte, classeDeFonte, nblignes, rôle*} est l'ensemble des attributs extraits sur les objets, à savoir leur type, le nombre de leurs voisins supérieurs (*nbvs*), le nombre de leurs voisins inférieurs (*nbvi*), la taille de la fonte (*tailleDeFonte*), la classe de la fonte (*classeDeFonte*), le nombre de lignes pour un objet textuel (*nblignes*) et le rôle pour un filet ou un cadre (*rôle*);

$$
\begin{array}{lll}
\mathbf{T_S} = \{t_{type} : type & \rightarrow & \{filet, ligneDeTexte, \\
 & & illustration, cadre\}, \\
t_{nbvs} : nbvs & \rightarrow & \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\
t_{nbvi} : nbvi & \rightarrow & \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\
t_{tailleDeFonte} : t_{tailleDeFonte} & \rightarrow & \{petit, moyen, grand\}, \\
t_{nblignes} : t_{nblignes} & \rightarrow & \{0, 1, 2\text{-}5, 5\text{-}\infty\}, \\
t_{rôle} : t_{rôle} & \rightarrow & \{souligneur, \\
 & & séparateurInterne \\
 & & séparateurExterne\}
\end{array}
$$

est le domaine des valeurs de chaque attribut de D_S;

$\mathbf{D_A}$ = {*position*,*compTailleDeFonte*} est une description de la position relative (*position*) et une comparaison de la taille de la fonte (*compTailleDeFonte*), attributs extrait sur un couple d'objets;

$$
\begin{array}{lll}
\mathbf{T_A} = & & \\
\{t_{compTailleDeFonte} : compTailleDeFonte & \rightarrow & \{=, <, >\}, \\
t_{position} : position & \rightarrow & (N \times N) \times (I \times I) \times \\
 & & (P \times P)\}
\end{array}
$$

est le domaine des valeurs de chaque attribut de D_A où $N = \{0, \ldots, 12\}$ représente le type, I l'immédiateté et P la proximité de la relation.

Les classes à disposition sont :

$$\begin{aligned}\mathbf{C} = \quad & \{< type, V_{IG} >, < type, V_{ID} >, < type, V_{SG} >, < type, V_{SD} >, \\ & < nbvs, ID >, < nbvi, ID >, < tailleDeFonte, ID >, \\ & < tailleDeFonte, V_{SG} >, < tailleDeFonte, V_{SD} >, \\ & < tailleDeFonte, V_{IG} >, < tailleDeFonte, V_{ID} >, \\ & < classeDeFonte, ID >, < classeDeFonte, V_{SG} >, \\ & < classeDeFonte, V_{SD} >, < classeDeFonte, V_{IG} >, \\ & < classeDeFonte, V_{ID} >, < nbligne, ID >, < nbligne, V_{SG} >, \\ & < nbligne, V_{SD} >, < nbligne, V_{IG} >, < nbligne, V_{ID} >, < rôle, ID >, \\ & < rôle, V_{SG} >, < rôle, V_{SD} >, < rôle, V_{IG} >, < rôle, V_{ID} >, \\ & < position, V_{SG} >, < position, V_{SD} >, < position, V_{IG} >, \\ & < position, V_{ID} >, < compTailleDeFonte, V_{SG} >, \\ & < compTailleDeFonte, V_{SD} >, < compTailleDeFonte, V_{IG} >, \\ & < compTailleDeFonte, V_{ID} >\},\end{aligned}$$

Tableau 5.3: Caractéristiques extraites sur les objets à classer

$$\begin{aligned}\mathbf{\Omega} = \quad & \{texteDeBase, titre, sousTitre, auteur, fonctionAuteur, \\ & résumé, source, légende, ancreVers, ancreDe, rubrique, \\ & sujet, entrefiletTitre, entrefiletContenu\}.\end{aligned}$$

La partie statique du modèle (C, Σ, Π) est décrite dans la table 5.3. Le sélecteur de caractéristiques $\sigma_\omega \in \Sigma$ d'une classe $\omega \in \Omega$ est construit automatiquement par apprentissage. Afin d'illustrer ce processus, nous allons énumérer les caractéristiques choisies parmi les 34 caractéristiques disponibles pour les classes *texteDeBase* et *titre*. Ces sélecteurs sont le résultat d'un apprentissage sur 22 pages du Los Angeles Times.

Classe *texteDeBase*

- <*tailleDeFonte*,V_{SG}>: la taille de fonte du voisin supérieur gauche;
- <*tailleDeFonte*,V_{IG}>: la taille de fonte du voisin inférieur gauche;
- <*classeDeFonte*,*ID*>: la classe de la fonte de l'objet;
- <*nbligne*,*ID*>: le nombre de lignes de l'objet;
- <*nbligne*,V_{SD}>: le nombre de lignes du voisin supérieur gauche.

Classe *title*

- <*type*,V_{SG}>: le type du voisin supérieur gauche;
- <*type*,V_{ID}>: le type du voisin inférieur droit;
- <*nbvs*,*ID*>: le nombre de voisin supérieurs de l'objet;
- <*tailleDeFonte*,*ID*>: la taille de fonte de l'objet;

- $<$*nbligne,ID*$>$: le nombre de lignes de l'objet;
- $<$*nbligne,*$V_{SD}>$: le nombre de lignes du voisin supérieur droit.

On remarque par exemple que la classe de la fonte est déterminante pour la classe *texteDeBase* alors que pour la classe *title*, c'est plutôt la taille de la fonte. La taille de la fonte des objets voisins d'un objet *texteDeBase* est déterminante, alors que pour la classe *title*, c'est plutôt le type des objets voisins. Si l'on se réfère à la figure 5.21, on constate que les blocs entourant le *texteDeBase* ont une fonte de petite taille alors que le type du voisin supérieur gauche d'un *titre* est souvent un filet.

5.8.2 Tests et évaluation

La figure 5.22 montre l'évolution du modèle en cours d'apprentissage. Le nombre de manipulations nécessaire est beaucoup plus grand que dans les applications précédentes car le nombre de classes est nettement plus élevé (14). Là aussi, l'effet de bord de la spécialisation est visible par la chute du taux de reconnaissance.

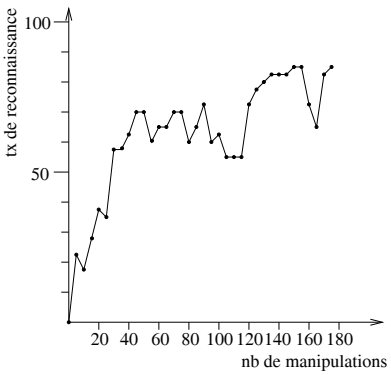


Figure 5.22: Évolution du modèle au cours de l'apprentissage : taux de reconnaissance par rapport au nombre de manipulations de l'opérateur.

L'évaluation du modèle pour l'étiquetage logique a été faite sur 29 documents. Étant donné le nombre de classes, nous avons choisi de détailler le résultat seulement pour les classes les plus représentées, à savoir "texte de base", "titre", "auteur", "résumé" et "source". Le résultat des autres classes est synthétisé sous la ligne "autres".

Sur le tableau 5.4 qui présente les résultats, on remarque que les principaux échecs sont dus d'abord à des conflits, puis à un manque de connaissance. Il est très rare que des blocs soient classés faux. On peut expliquer que le nombre de conflits soit nettement plus élevé que dans les autres applications par le fait que le nombre de classes est élevé. Une confusion typique est "titre" et "rubrique", ce qui n'est pas vraiment étonnant puisque les objets de la classe "rubrique" ont physiquement les

classes	nombre total	reconnus	inconnus	en conflit	mal classés
textes de base	442	90%	5%	5%	0%
titres	111	69%	18%	6%	7%
auteurs	72	96%	1%	0%	3%
résumés	37	84%	0%	14%	2%
sources	63	94%	3%	0%	3%
autres	252	75%	15%	8%	2%

Tableau 5.4: Tests "batchs" pour l'étiquetage des blocs effectués sur 29 pages du Los Angeles Times; les résultats sont exprimés en pourcentage du nombre total de blocs d'un type donné.

caractéristiques d'un titre: ils se trouvent en début d'article, sont composés de caractères de taille plus grande que les éléments voisins et sont la plupart du temps soulignés.

Les mêmes tests ont été menés avec le deuxième prototype de *2(CREM)*. Dans ce prototype on prend en compte à chaque apprentissage toutes les configurations de la session (correspond en principe à une page de document) classées par l'opérateur. A la fin de chaque session, une mise à jour du modèle est faite avec tous les documents qui ont été correctement classés.

Les tests d'apprentissage ont également été ménés avec le deuxième prototype. Dans cette réalisation de *2(CREM)* , on prend en compte à chaque apprentissage toutes les configurations de la session classées par l'opérateur (une session correspond à une page de document). A la fin de chaque session, une mise à jour du modèle est faite avec tous les documents qui ont été reconnus. Les résultats de ces tests sont représentés sur la figure 5.23.

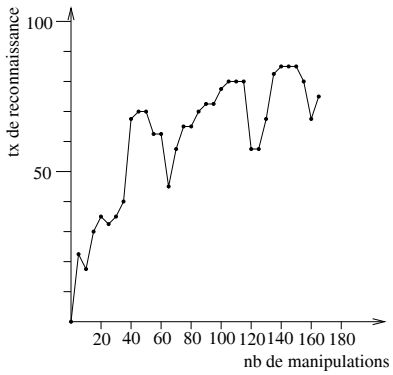


Figure 5.23: Évolution du modèle au cours de l'apprentissage avec le deuxième prototype de *2(CREM)* : taux de reconnaissance par rapport au nombre de manipulations de l'opérateur.

On remarque que les résultats sont comparables à ceux obtenus avec le premier prototype. Ici aussi il n'y a pas convergence du modèle. Nous pensons que ceci est dû au nombre élevé de classes et à la non pertinence des caractéristiques choisies au niveau de la spécification de l'application. Elles ne permettent pas de distinguer toutes les classes et des conflits apparaissent. L'avantage du deuxième prototype se manifeste surtout sur le plan de la convivialité du système.

Chapitre 6

Conclusion

Ce chapitre tire un bilan sur le travail réalisé. Un bref résumé permet de rappeler le contexte de l'étude, les objectifs et la démarche adoptée. Les principales contributions sont exposées dans la section suivante et le travail se termine par une discussion sur les voies qui pourraient être approfondies dans l'idée d'une continuation.

6.1 Résumé

Le but de cette étude est de mettre au point une méthode de reconnaissance d'images de *documents complexes* qui soit douée d'*apprentissage incrémental*. Le thème de la reconnaissance de la structure physique de bas niveau ayant déjà donné lieu à beaucoup de recherches, nous nous sommes concentrés sur la reconnaissance de la structure physique de haut niveau ainsi que sur la reconnaissance de la structure logique, domaines nettement moins traités dans la littérature. Notre démarche a été la suivante : nous avons travaillé sur des images idéales synthétisées à partir de documents PDF, nous affranchissant par là des problèmes de biais ou de bruit liés à la saisie du document par un scanner. Dans un premier temps, nous avons développé des algorithmes simples de segmentation, ce qui nous a permis d'une part d'obtenir les données pour faire de la reconnaissance de haut niveau, et d'autre part de nous familiariser avec les documents à structure complexe afin d'en saisir la spécificité et d'en déduire les besoins particuliers en matière de reconnaissance. La deuxième étape a été la mise au point de *2(CREM)*, une méthode de reconnaissance pour les documents à structure complexe douée d'apprentissage incrémental.

6.2 Contributions

Notre contribution va de l'observation à l'expérimentation en passant par la conception et la réalisation d'un outil logiciel. Dans cette section nous allons résumer quelles ont été nos principales contributions.

6.2.1 Étude de la spécificité des documents à structure complexe

Une étude de la spécificité des documents à structure complexe a été menée. Son but était de déterminer si de tels documents pouvaient être analysés avec les méthodes développées pour les documents simples. Dans la négative, nous désirions connaître quels aspects de ces documents changeaient la nature de la tâche de reconnaissance et quelle en était l'implication dans la conception d'un outil de reconnaissance adapté. L'étude conclut que les méthodes doivent être revues : l'ordre de lecture des documents complexes n'est pas trivial et l'information ne peut être facilement linéarisée. On peut alors imaginer deux manières de reconnaître la structure logique d'un document complexe : le recouvrement de l'ordre de lecture peut précéder ou suivre l'étiquetage logique des objets du document. Nous préconisons un étiquetage logique préalable au recouvrement de l'ordre de lecture car nous pensons que la structure physique ne suffit pas à déterminer de manière univoque un ordre de lecture. En conséquence, les outils de reconnaissance doivent pouvoir représenter et interpréter un voisinage en deux dimensions : un objet ne comporte pas un précédent et un suivant comme pour les documents à structure simple, mais des voisins dans toutes les directions.

6.2.2 Mise au point d'un concept pour la description de la position relative de deux objets

Le formalisme utilisé par Azokly [3] pour l'expression des positions relatives entre les enveloppes rectangulaires de deux objets a été étendu de manière à exprimer l'interférence d'un tiers objet et la distance séparant les deux objets. La description du type de position relative par le formalisme de Azokly a été nommée *type* de la position relative. Une position relative entre deux objets peut être *directe*, *semi-directe* ou *indirecte* sur chacun des axes horizontaux et verticaux en fonction de la présence et de la position d'un tiers objet entre les deux objets : cette description est appelée *immédiateté* de la position relative. Finalement, la *proximité* est une mesure de l'éloignement de deux objets sur un axe en fonction de leurs tailles respectives. Le type, l'immédiateté et la proximité permettent de rendre compte de la position relative de l'objet et de chacun de ses voisins et ainsi de décrire le voisinage d'un objet dans un espace à deux dimensions.

6.2.3 Conception d'une méthode de classification

Une méthode générale de classification a été conçue. Elle a été baptisée *2(CREM)* et a les propriétés suivantes : 1) elle tient compte des caractéristiques intrinsèques de l'objet à classer aussi bien que des caractéristiques des objets qui forment son voisinage; 2) elle prévoit l'expression et l'interprétation du voisinage d'un objet dans

les deux dimensions; 3) elle est douée d'apprentissage incrémental. L'apprentissage du modèle se fait à deux niveaux : ce sont non seulement les valeurs que prennent des caractéristiques pour une classe donnée qui sont apprises, mais aussi les caractéristiques pertinentes pour une classe. La liste des valeurs des caractéristiques d'un objet et de son voisinage est appelée *configuration* de l'objet. Le modèle est constitué pour chaque classe d'un ensemble de configurations de références appelées *patterns*. Un objet est attribué à une classe si sa configuration correspond à un des patterns de la classe.

6.2.4 Implémentation de *2(CREM)*

2(CREM) a été implémentée puis appliquée à la reconnaissance d'images de journaux dans quatre applications : la reconnaissance de filets, la reconnaissance de cadres, la fusion des lignes de texte en blocs et l'étiquetage logique des blocs de texte. La diversité de ces quatre applications montre le caractère général de la méthode. La différence entre ces quatre applications réside uniquement dans l'extraction de certaines caractéristiques.

Le modèle, les données et les résultats intermédiaires et finaux ont été représentés en XML. Nos expériences ont confirmé que le choix de XML comme standard de représentation des données dans le domaine de la reconnaissance de document est tout à fait justifié. La visualisation et la manipulation des données dans le cadre de l'apprentissage interactif a été possible grâce à l'utilisation de l'environnement *XMIllum* développé par Oliver Hitz. L'adaptation des données au format accepté par *XMIllum* se fait tout simplement par une transformation spécifiée par une feuille de style XSLT.

6.2.5 Évaluation de *2(CREM)*

2(CREM) a été testée pour les quatre applications décrites dans le chapitre 5. Deux types de tests ont été effectués sur des exemplaires du Los Angeles Times. Les résultats des tests "batchs" montrent la capacité de reconnaissance du modèle. Les résultats sont suffisamment bons pour conclure à la pertinence de la méthode. Les autres tests évaluent le processus d'apprentissage du modèle en mettant en parallèle le nombre de manipulations effectuées par l'opérateur et le taux de reconnaissance. Le nombre de manipulations nécessaires pour atteindre un taux de reconnaissance satisfaisant est tout à fait raisonnable. L'apprentissage incrémental a donc le double avantage de permettre l'adaptation du modèle à de légères modifications du document à analyser et d'éviter la saisie manuelle d'échantillons de tests, travail long et fastidieux.

6.2.6 *2(CREM)*, un outil pour la constitution de fonds de vérité

Un des grands problèmes non encore résolus de la reconnaissance d'images de documents est la constitution de fonds de vérité. Les fonds de vérité sont des documents déjà reconnus : ils sont nécessaires d'une part pour entraîner certains types de systèmes de reconnaissance, d'autre part pour évaluer les résultats de reconnaissance. *2(CREM)* est un système d'apprentissage incrémental : on peut faire de la reconnaissance même lorsque le système n'a aucune connaissance sur la classe de documents. Le modèle est alors vide et est construit petit à petit par la collaboration de l'utilisateur et du système. Ainsi, la constitution de fonds de vérité est grandement facilitée : au lieu d'étiqueter manuellement l'ensemble des documents constituant un échantillon, l'utilisateur fait quelques manipulations sur les premiers documents de l'échantillon.

6.2.7 Implémentation d'un outil de classification de fontes

Un outil de classification de fontes par appariement a été implémenté selon l'idée de Lebourgeois [43]. Cet outil est basé sur le principe suivant : deux caractères présents dans le même mot appartiennent à la même classe. Les classes de fonte sont ainsi reconstituées par recoupement. L'outil fonctionne sur les images idéales de documents car deux caractères sont considérés comme semblables seulement s'ils sont identiques au pixel près. Il est également doué d'apprentissage puisque les spécimens des différentes classes sont conservés dans un modèle. Il a permis l'extraction d'une caractéristique essentielle pour l'étiquetage logique. En effet, la fonte (par son type et sa taille) est un des principaux attributs qui véhiculent la structure logique implicite des documents à structure complexe.

6.3 Extensions

Bien que *2(CREM)* ait été implémentée et testée, elle n'est pas intégrée à un système complet de reconnaissance. Dans cette section, nous allons énumérer divers points qui mériteraient un approfondissement.

6.3.1 Développement d'un système complet de reconnaissance

L'extension la plus intéressante du point de vue pratique nous semble le développement d'un système complet de reconnaissance de la structure logique des images de journaux. Bien que les quatre applications développées sur la base de *2(CREM)* fassent un très grand pas dans cette direction, une étape de recouvrement de l'ordre

de lecture et de découpe en articles doit suivre l'étiquetage des blocs de texte. Nous pensons que là aussi *2(CREM)* pourrait être appliquée, un peu de la même manière qu'elle a été utilisée pour la fusion des lignes de texte en blocs. Chaque couple de blocs voisins (texte ou illustration) pourraient être classé dans une des catégories décrivant les trois situations suivantes : 1) les deux blocs se suivent dans l'ordre de lecture et donc appartiennent au même article; 2) les deux blocs appartiennent au même article mais ne se suivent pas; 3) les deux blocs n'appartiennent pas au même article. Un post-traitement devrait permettre de réunir deux blocs qui se suivent dans l'ordre de lecture mais ne sont pas physiquement voisins (cas plutôt rare).

6.3.2 Choix des caractéristiques

Dans la section *Choix des caractéristiques* du chapitre 5, nous avons parlé du choix par le système des caractéristiques discriminantes pour chaque classe. Rappelons que le seul critère de *discrimination* pour le choix d'une caractéristique supplémentaire pose le problème du choix d'une mauvaise caractéristique (c'est-à-dire peu homogène pour la classe ou peu discriminante) dans la phase initiale de l'apprentissage. C'est pourquoi nous avons introduit un algorithme de mise à jour qui recalcule l'ensemble des caractéristiques discriminantes lorsque le modèle est suffisamment riche en connaissance. Les critères de choix sont la *discrimination* et l'*homogénéité*. Avec cet algorithme, le système ne trouve pas forcément l'ensemble idéal en regard de ces deux critères : en effet, on ne peut se permettre de tester tous les ensembles possibles car il y aurait explosion combinatoire. Nous avons adopté une solution qui consiste à effectuer un classement entre les caractéristiques, toujours en fonction des critères de discrimination et d'homogénéité. Peu d'effort a été investi dans la mise au point de l'algorithme de choix et nous sommes convaincus qu'une réflexion plus poussée permettrait d'améliorer encore les résultats.

6.3.3 Révision des interactions homme-machine

Dans le premier prototype de *2(CREM)*, la seule action proposée à l'opérateur est la correction des résultats erronés. Le système interprète les corrections pour enrichir son modèle de reconnaissance. L'apprentissage du modèle se fait en fonction de la dernière correction et de l'état courant du modèle sans prendre en compte les corrections antérieures. Rappelons qu'ainsi on ne peut garantir que le système ne modifie pas la classe d'un objet qui avait été attribuée par l'opérateur, ce qui nuit gravement à la convivialité du système. Dans cette section nous envisageons deux solutions à ce problème.

Une première solution a été implémentée dans le deuxième prototype : les configurations classées par l'opérateur sont mémorisées durant la session et à la fin de chaque session une mise à jour du modèle est lancée.

Une deuxième solution permet d'éviter la mémorisation des configurations. Dans le document en voie de reconnaissance, on fait simplement la distinction entre les classes attribuées par le système et celles attribuées par l'opérateur. Ainsi, lors de la classification, le système peut éviter de changer une classe attribuée par l'opérateur. Par contre, si ce procédé permet de garantir que la classe attribuée par l'opérateur à un objet ne change pas au cours d'une session, lors de l'apprentissage le modèle est modifié sans prise en compte des corrections antérieures. Si on relance le système sur le même document, il est possible que certains objets classés par l'opérateur ne soient plus reconnus par le système. L'utilisation de l'algorithme que nous avons présenté dans la section *Choix des caractéristiques* du chapitre 5 permet de remédier à cette faiblesse. Il suffit, après chaque session, de lancer une remise à jour du modèle en lui fournissant tous les documents reconnus antérieurement. Cette méthode devrait même être plus efficace que la précédente puisque le système peut ainsi garantir que non seulement tous les objets classés par l'opérateur seront reconnus, mais que également les autres objets (dont la classe n'a pas été corrigée) se verront toujours attribuer la même classe. Par contre le risque d'erreurs de la part de l'utilisateur est plus grand : il peut ne pas avoir repéré une erreur dans la classification proposée par le système. Ce point est délicat car ainsi le modèle est construit sur la base de données erronées.

6.3.4 Remise en cause des résultats

Durant la classification, lorsqu'une classe est attribuée à un objet, elle l'est définitivement, à moins d'une intervention externe au système. Certaines méthodes, comme celles de Héroux [24] et de Lebourgeois [43] prévoient une remise en cause en fonction de la classification des objets physiquement proches; la classification est itérée jusqu'à l'obtention d'une certaine stabilité. Dans notre cas, la remise en cause pourrait être intéressante, notamment pour l'étiquetage logique des blocs de texte. L'étiquette d'un bloc est fortement prévisible si l'on connaît l'étiquette des blocs voisins. L'ajout des étiquettes logiques des blocs voisins dans l'ensemble des caractéristiques à disposition du système implique une révision de l'architecture du système de manière à ce qu'il prévoie la remise en cause de ses résultats. Un modèle contenant des données probabilistes permettrait d'évaluer globalement une solution. Un tel modèle a été proposé par Rolf Brugger [13] : les données du modèle sont représentées sous la forme de n-grams généralisés. Elles expriment les probabilités d'apparitions d'un événement qui ne dépendent pas uniquement d'événements précédents (frères dans la hiérarchie d'un document), mais aussi d'événements englobant ou contenus (parents et enfants dans la hiérarchie). Son modèle ne permet par contre pas d'exprimer des relations entre objets s'organisant dans un espace à deux dimensions.

Notre modèle pourrait assez facilement être adapté de manière à ce qu'il contienne des données probabilistes. Rappelons que le processus de classification d'un objet consiste à mettre la configuration de l'objet en correspondance avec les patterns des

classes à disposition. Un objet est assimilé à une classe si pour toutes les caractéristiques discriminantes de la classe, la configuration a les mêmes valeurs qu'un pattern de la classe. Au lieu de cela on pourrait retourner pour chaque classe un indice de plausibilité qui dépendrait du nombre de valeurs identiques par rapport au nombre de caractéristiques discriminantes. L'indice de plausibilité global pour un document ou une portion de document serait alors la somme des indices de plausibilité des classes choisies pour chaque objet. L'idée est de trouver le choix de classes qui maximise l'indice de plausibilité global.

6.3.5 Classification des fontes applicables à des documents non-idéaux

L'outil de classification de fontes que nous avons implémenté ne s'applique qu'à des images idéales de documents. L'algorithme adopté pour la mise en correspondance de deux caractères est très rigide. Deux signes sont considérés comme étant le même caractère s'ils sont identiques au pixel près. Une mise en correspondance plus souple, basée par exemple sur les masques ternaires de Ingold [34], permettrait d'appliquer la méthode à des documents scannés.

6.3.6 Tests et application de *2(CREM)*

Il serait intéressant de tester la méthode sur d'autres journaux que le Los Angeles Times. On pourrait par exemple observer dans quelle mesure un modèle appris sur un journal peut être réutilisé et adapté pour un autre journal. La méthode pourrait aussi être testée sur d'autres types de documents complexes comme des catalogues.

Nous pensons que *2(CREM)* peut servir à toute reconnaissance d'objets et qu'elle est particulièrement bien adaptée lorsque les objets sont organisés en deux dimensions. On peut aussi imaginer étendre *2(CREM)* pour qu'elle prenne en compte une troisième dimension si l'on définit un formalisme capable d'exprimer des relations de voisinage dans un tel contexte.

6.4 Bilan général

Cette thèse a étudié le problème de la réingénierie des documents à structure complexe. Elle a établi que l'organisation bidimensionnelle de tels documents a des conséquences déterminantes sur le processus de reconnaissance et nécessite des outils spécialisés c'est pourquoi j'ai inventé *2(CREM)* , une méthode générale de classification douée d'apprentissage incrémental et adaptée à la reconnaissance d'objets dans un contexte bidimensionnel. Avec l'amélioration des techniques de reconnaissance, il est probable que l'utilisation des systèmes de réingénierie de documents se banalise un jour et j'espère que mon travail a fait un pas dans cette direction.

Bibliographie

[1] J. Allen. Maintaining Knowledge about Temporal Intervals. *ACM*, 26(11): 832–843, 1983.

[2] A. Antonacopoulos. Page Segmentation Using the Description of the Background. *Computer Vision and Image Understanding*, 70(3): 350–369, June 1998.

[3] A. Azokly. *Une approche générique pour la reconnaissance de la structure physique de documents composites.* PhD thesis, IIUF-Université de Fribourg, 1995.

[4] F. Bapst. *Reconnaissance de documents assistée: architecture logicielle et intégration de savoir-faire.* PhD thesis, IIUF-University of Fribourg, Fribourg-Switzerland, 1998.

[5] F. Bapst, R. Brugger, and R. Ingold. Towards an Interactive Document Recognition System. Internal working paper 95-09, IIUF-University of Fribourg, Fribourg-Switzerland, March 1995.

[6] F. Bapst, R. Brugger, A. Zramdini, and R. Ingold. Integrated Multi-Agent Architecture for Assisted Document Recognition. In *DAS'96*, pages 172–188, Malvern, Pennsylvania, October 1996. Reprinted in: Document Analysis II, J. J. Hull and S. L. Taylor (Eds), World Scientific, 1998, pages 301-317.

[7] F. Bapst, R. Brugger, A. Zramdini, and R. Ingold. L'intégration des données dans un système de reconnaissance de documents assistée. In *CNED'96*, Nantes (France), 1996.

[8] F. Bapst, A. Zramdini, and R. Ingold. A Scenario Model Advocating User-driven Adaptive Recognition Systems. In *ICDAR'97*, pages 745–748, Ulm-Germany, August 1997.

[9] Frédéric Bapst and Rolf Ingold. Using Typography in Document Image Analysis. In Roger D. Hersch, Jacques André, and Heather Brown, editors, *Electronic Publishing, Artistic Imaging and Digital Typography (RIDT'98)*, number 1375 in Lecture notes in computer science, pages 240–251, March 1998.

[10] A. Belaïd. Conception automatisée de modèles de page en vue de leur utilisation en reconnaissance de documents. In *LAMPE'97*, Lausanne (Suisse), September 1997.

[11] A. Belaïd and Y. Toussaint. Une méthode d'étiquetage morpho-syntaxique pour la reconnaissance de tables de matières. In *CIFED'2000*, pages 51–60, Lyon (France), July 2000.

[12] D. Berthold. Klassifizierung von Fonts. Master's thesis, IIUF-Université de Fribourg, September 2000.

[13] R. Brugger. *Eine statistische Methode zur Erkennung von Dokumentstrukturen.* PhD thesis, IIUF-University of Fribourg, Fribourg-Switzerland, 1999.

[14] R. Brugger, F. Bapst, and R. Ingold. A DTD Extension for Document Structure Recognition. In Roger D. Hersch, Jacques André, and Heather Brown, editors, *Electronic Publishing, Artistic Imaging and Digital Typography (RIDT'98)*, number 1375 in Lecture Notes in Computer Science, pages 343–354, St-Malo, France, March 1998.

[15] R. Brugger, A. Zramdini, and R. Ingold. Modeling Documents for Structure Recognition Using Generalized n-grams. In *ICDAR'97*, pages 56–60, Ulm-Germany, August 1997.

[16] F. Chang, K. Liang, T. Tan, and W. Hwang. Binarization of Document Images using Hadamard Multiresolution Analysis. In *ICDAR'99*, pages 374–377, Bangalore (India), September 1999.

[17] L. Cinque, L. Lombardi, and G. Manzini. A Multiresolution Approach For Page Segmentation. *Pattern Recognition Letters*, 19: 217–225, October 1998.

[18] Aldus Corporation. TIFF Revision 6.0. http: //www.adobe.com: 80/support/salesdocs/2596.htm, 1992.

[19] F. Esposito, D. Malerba, and G. Semeraro. Automated Acquisition of Rules for Document Understanding. In *ICDAR'93*, pages 650–654, Tsukuba (Japan), 1993.

[20] K. Fan, C. Liu, and Y. Wang. Segmentation and Classification of Mixed Text/Graphics/Image Documents. *Pattern Recognition Letters*, 15: 1201–1209, December 1994.

[21] J. Fisher, S. Hinds, and D. D'Amato. A Rule-based System For Document Image Segmentation. In *10th International Conference on Pattern Recognition*, pages 567–570, Atlantic City (USA), 1990.

[22] B. Gatos, S.L. Mantzaris, K.V. Chandrinos, A. Tsigris, and S.J. Perantonis. Integrated Algorithms for Newspaper Page Decomposition and Article Tracking. In *ICDAR'99*, pages 559–562, Bangalore (India), september 1999.

[23] V. Govindaraju, S. W. Lam, D. Niyogi, D. B. Sher, R. Srihari, S. N. Srihari, and D. Wang. Newspaper Image Understanding. In *Lecture Notes in Artificial Intelligence 444*, pages 375–384, Atlantic City (USA), june 1989.

[24] P. Héroux, E. Trupin, and Y. Lecourtier. Modélisation et classification pour la rétroconversion des documents. In *CIFED'00*, pages 413–421, Lyon (France), July 2000.

[25] E. Herwijnen. Practical SGML. Kluwer Academic Publisher, 1990.

[26] O. Hitz and R. Ingold. Visualization of Document Recognition Results using XML Technology. In *CIDE'2000*, Lyon (France), July 2000.

[27] O. Hitz, L. Robadey, and R. Ingold. Analysis of Synthetic Document Images. In *ICDAR'99*, pages 374–377, Bangalore (India), September 1999.

[28] O. Hitz, L. Robadey, and R. Ingold. Using XML in Document Recognition. In *DLIA'99*, Bangalore (India), September 1999.

[29] O. Hitz, L. Robadey, and R. Ingold. An Architecture for Editing Document Recognition Results using XML Technology. In *DAS'2000*, pages 385–396, Rio de Janeiro (Brazil), December 2000.

[30] P. Héroux, S. Diana, A. Ribert, and E. Trupin. Etude de méthodes de classification pour l'identification automatique de classes de formulaires. In *CIFED'98*, pages 286–295, Québec (Canada), Mai 1998.

[31] T. Hu. *New Methods for Robust and Efficient Recognition of the Logical Structures in Documents.* PhD thesis, IIUF-Université de Fribourg, 1994. n. 1076.

[32] Adobe Systems Incorporated. PostScript Language. Addison-Wesley, 1986.

[33] Adobe Systems Incorporated. Portable Document Format Reference Manual. Addison-Wesley, 1993.

[34] R. Ingold. *Structures de documents et lecture optique: une nouvelle approche.* Presses Polytechniques Romandes, 1990.

[35] Y. Ishitani. Logical Structure Analysis of Document Images Based on Emergent Computation. In *ICDAR'99*, pages 189–192, Bangalore (India), September 1999.

[36] A. Jain, R. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1): 4–37, January 2000.

[37] H. Kamada and K. Fujimoto. High-speed, High-accuracy Binarization Method for Recognizing Text in Images of Low Spatial Resolutions. In *ICDAR'99*, pages 139–142, Bangalore (India), September 1999.

[38] S. Klink, A. Dengel, and T. Kieninger. Document Structure Analysis Based on Layout and Textual Features. In *DAS'2000*, pages 41–50, Rio de Janeiro (Brazil), December 2000.

[39] D. Knuth. The Texbook. *Computers and Typesetting*, 1986.

[40] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan. Syntactic Segmentation and Labeling of Digitized Pages from Technical Journals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7): 737–747, July 1993.

[41] S. Lam, D.Wang, and S. Srihari. Reading Newspaper Text. In *10th International Conference on Pattern Recognition*, pages 703–705, Atlantic City (USA), 1990.

[42] L. Lamport. LaTeX, a Document Preparation System. Addison-Wesley, 1994.

[43] F. Lebourgeois, H. Emptoz, and H. Vigne. Rasade: Reconnaissance automatique des structures associées aux documents écrits. In *CIFED'2000*, pages 281–294, Lyon (France), July 2000.

[44] P. Lefèvre and F. Reynaud. ODIL: an SGML Description Language of the Layout of Documents. In *ICDAR'95*, pages 480–488, Montreal (Canada), 1995.

[45] U. Mahadevan and S. Srihari. Parsing and Recognition of City, State, and ZIPCodes in Handwritten Addresses. In *ICDAR'99*, pages 325–328, Bangalore (India), September 1999.

[46] U. Miletzki, T. Bayer, and H. Schafer. Continuous Learning Systems Postal Address Readers with Built-in Learning Capability. In *ICDAR'99*, pages 329–332, Bangalore (India), September 1999.

[47] C. Musciano and B. Kennedy. *HTML (Hyper-Text Markup Language) – The Definitive Guide.* O'Reilly, 1997.

[48] G. Nagy, S. Seth, and M. Viswanathan. A Prototype Document Image Analysis System for Technical Journals. *Computer*, 25(7): 10–22, July 1992.

[49] D. Niyogi. *A Knowledge-based Approach to Deriving Logical Structure from Document Images.* PhD thesis, Faculty of the Graduate School of State University of New York at Buffalo, Buffalo-USA, 1994.

[50] J. Ogier, R. Mullot, J. Labiche, and Y. Lecourtier. Interprétation de documents par cycles perceptifs de construction d'objets cohérents. Application aux données cadastrales. *Traitement du signal*, 12(6): 627–637, 1995.

[51] L. O'Gorman. The Document Spectrum for Page Layout Analysis. *IEEE Trans. pattern analysis and Machine Intelligence*, 15(11): 1162–1173, November 1993.

[52] V. Quint, H. Richy, C. Roisin, and I. Vatton. Thot - Manuel utilisateur. Imag - INRIA, November 1995.

[53] L. Robadey, O. Hitz, and R. Ingold. Segmentation de documents idéaux à structure complexe. In *CIFED'00*, pages 383–392, Lyon (France), July 2000.

[54] L. Robadey, O. Hitz, and R. Ingold. A Pattern-Based Method for Document Structure Recognition. In *DLIA'01*, Seattle (USA), September 2001.

[55] L. Robadey and R. Ingold. Détection automatique de pages vides dans un système d'archivage, premières experiences. In *CIFED'98*, pages 286–295, Québec (Canada), Mai 1998.

[56] A. Rosenfeld, R. Hummel, and S. Zucker. At the Frontiers of OCR. *IEEE trans SMC*, 6(6): 420–433, June 1976.

[57] N. Roussel, O. Hitz, and R. Ingold. Web-based Cooperative Document Understanding. In *ICDAR'01*, pages 368–373, Seattle (USA), September 2001.

[58] R. Sivaramakrishnan, I. Phillips, J. Subramanium, and R. Haralick. Zone Classification in a Document using the Method of Feature Vector Generation. In *ICDAR'95*, pages 541–544, Montréal (Canada), 1995.

[59] L. Spitz. Style-Directed Document Recognition. In *Document Layout Interpretation and its Application - DLIA'99*, Bangalore (India), September 1999.

[60] T. Steinherz, N. Intrator, and E. Rivlin. Skew Detection via Principal Components Analysis. In *ICDAR'99*, pages 153–156, Bangalore (India), September 1999.

[61] RAF Technology. DAFS Library, Programmer's Guide and Reference, August 1995.

[62] S. Tsujimoto and H. Asada. Understanding Multi-articled Documents. In *10th International Conference on Pattern Recognition*, pages 551–556, Atlantic City (USA), 1990.

[63] World Wide Web Consortium (W3C). XML-QL: A Query Language for XML. http: //www.w3.org/TR/NOTE-xml-ql, August 1998.

[64] World Wide Web Consortium (W3C). XSL Transformations (XSLT) 1.0. http: //www.w3.org/TR/WD-xslt, November 1999.

[65] World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0 (Second Edition). http: //www.w3.org/TR/REC-xml, October 2000.

[66] World Wide Web Consortium (W3C). Extensible Stylesheet Language (XSL) 1.0. http: //www.w3.org/TR/WD-xsl, November 2000.

[67] H. Walischewski. Automatic Knowledge Acquisition for Spatial Document Interpretation. In *ICDAR'97*, pages 243–247, Ulm (Germany), 1997.

[68] H. Walischewski. Learning Regions of Interest in Postal Automation. In *ICDAR'99*, pages 317–320, Bangalore (India), September 1999.

[69] D. Wang and S. N. Srihari. Classification of Newspaper Image Blocks Using Texture Analysis. In *Computer Vision, Graphics, and Image Processing*, volume 47, pages 327–352, 1989.

[70] P. S. Williams and M. D. Alder. Generic Texture Analysis Applied to Newspaper Segmentation. In *ICNN'96*, volume 3, pages 1664–1669, Washington DC (USA), June 1996.

[71] K.Y. Wong, R.G. Casey, and F.H. Whal. Document Analysis System. *IBM J. Res. Dev.*, 26(6): 647–656, 1982.

[72] L. Xingyuan, D. Doermann, W. Oh, and W. Gao. A Robust Method for Unknown Forms Analysis. In *ICDAR'99*, pages 531–534, Bangalore (India), September 1999.

[73] M. Yamaoka, O. Iwaki, N. Babaguchi, and T. Kitahashi. Interactive Approach to the Extraction of Logical Structures from Unformatted Document Images Using a Substructure Model. In *ICDAR'99*, pages 185–188, Bangalore (India), September 1999.

Annexe A

DTDs pour structures de documents

Dans cette annexe se trouve une DTD XML de structure physique qui se veut suffisamment générale pour convenir à la plupart des documents et une DTD spécifique à la structure logique du Los Angeles Times.

A.1 DTD de la structure physique de documents

```
<!ELEMENT  document  (page+)>
<!ELEMENT  page      (block+)>
<!ELEMENT  block     ((column|block|frame|graphic)+)>
<!ELEMENT  frame     (block+)>
<!ELEMENT  column    (paragraph+)>
<!ELEMENT  paragraph (initial?,line+)>
<!ELEMENT  line      (word+)>
<!ELEMENT  word      (glyph+)>
<!ELEMENT  initial   (#PCDATA)>
<!ELEMENT  glyph     (#PCDATA)>

<!ATTLIST    document
  reference  CDATA           #IMPLIED
  resolution CDATA           #IMPLIED
  size       CDATA           #IMPLIED
>

<!ATTLIST    page
  number     CDATA           #IMPLIED
>

<!ATTLIST    block
  type       (rect|polygon)  "rect"
  pos        CDATA           #REQUIRED
```

```
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>
<!-- position and shape of the block -->

<!ELEMENT  graphic          EMPTY>
<!ATTLIST  graphic
  type       (thread|other)  "thread"
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>

<!ATTLIST  frame
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>

<!ATTLIST  column
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>

<!ATTLIST  paragraph
  font       CDATA           #REQUIRED
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>

<!ATTLIST  line
  font       CDATA           #REQUIRED
  pos        CDATA           #REQUIRED
  dim        CDATA           REQUIRED
  polyline   CDATA           #IMPLIED
>

<!ATTLIST  word
  font       CDATA           #REQUIRED
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>

<!ATTLIST  initial
  font       CDATA           #REQUIRED
```

```
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>

<!ATTLIST  glyph
  font       CDATA           #REQUIRED
  pos        CDATA           #REQUIRED
  dim        CDATA           #REQUIRED
  polyline   CDATA           #IMPLIED
>
```

A.2 DTD de la structure logique du Los Angeles Times

```
<!ELEMENT LAT ((article|illustration)+, editorial, letters)>
<!ATTLIST LAT day     CDATA #REQUIRED>
<!ATTLIST LAT weekday CDATA #REQUIRED>
<!ATTLIST LAT month   CDATA #REQUIRED>
<!ATTLIST LAT year    CDATA #REQUIRED>

<!ELEMENT article (title, summary? author? content,
                   illustration* citation*)>
<!ATTLIST article column  CDATA #IMPLIED>
<!ATTLIST article subject CDATA #IMPLIED>
<!ATTLIST article note    CDATA #IMPLIED>

<!ELEMENT illustration EMPTY>
<!ATTLIST illustration illustration CDATA #REQUIRED>
<!ATTLIST illustration title        CDATA #IMPLIED>
<!ATTLIST illustration source       CDATA #IMPLIED>
<!ATTLIST illustration caption      CDATA #IMPLIED>

<!ELEMENT title (#PCDATA)>

<!ELEMENT summary (#PCDATA)>

<!ELEMENT author EMPTY>
<!ATTLIST author firstname CDATA #REQUIRED>
<!ATTLIST author résumé    CDATA #REQUIRED>
<!ATTLIST author position  CDATA #IMPLIED>
<!ATTLIST author location  CDATA #IMPLIED>

<!ELEMENT summary (#PCDATA)>
<!ATTLIST summary keyword CDATA #IMPLIED>
```

```
<!ELEMENT citation   (text, author?)>

<!ELEMENT text       (#PCDATA)>

<!ELEMENT content    (title? text interfilet*)+>

<!ELEMENT interfilet (title? text (source|author)?)>

<!ELEMENT editorial  (title, content)+ author+)>

<!ELEMENT letters    ((title, (text, author)*)*)>
```