Application du raisonnement à partir de cas à l'analyse de documents administratifs

Hatem Hamza

Nancy-Université
Université Nancy 2

UFR mathématiques et informatique **École doctorale IAEM Lorraine**

Département de formation doctorale en informatique

Application du raisonnement à partir de cas à l'analyse de documents administratifs

THÈSE

présentée et soutenue publiquement le 21 Avril 2008

pour l'obtention du

Doctorat de l'université Nancy 2

(spécialité informatique)

par

Hatem HAMZA

Composition du jury

Président :	xxxxxxxx	
Rapporteurs :	Jean Marc Ogier	Professeur, Université de La Rochelle
	Simone Marinai	Professeur assistant, Université de Florence
Examinateurs :	Hubert Emptoz	Professeur, INSA de Lyon
	Jean Lieber	Maître de Conférences, Université Nancy 1
	Yolande Belaïd	Maître de Conférences, Université Nancy 2
Directeur de thèse :	Abdel Belaïd	Professeur, Université Nancy 2
Invité :	Vincent Poulain d'Andecy	Manager département recherche, ITESOFT

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Remerciements

Je dédie cette thèse
à tous ceux que j'aime, et tous ceux qui m'aiment.

Table des matières

Table des figures

1

Introduction générale

1.1 Contexte et problématique

Les administrations publiques telles que les CAF [1], les CPAM[2], les préfectures, les hôpitaux... et les administrations privées comme les banques, les mutuelles, ou les services administratifs des sociétés de vente par correspondance traitent chaque jour des milliers de documents. Par exemple, un grand vépéciste traite 40000 commandes par jour avec leurs paiements. Ces documents doivent donc être traités de manière rapide et efficace. Ceci implique une phase de saisie et d'extraction d'informations, afin de les introduire dans le système informatique de l'entreprise. Pour rendre ce processus rapide et efficace, il faut qu'il soit automatique.

Aujourd'hui, la lecture automatique de documents (LAD) permet d'automatiser ces tâches, mais pas toujours de manière optimale.

ITESOFT est un éditeur de logiciels de capture, traitement et gestion automatique de documents. Ses logiciels permettent la dématérialisation, la rétroconversion ainsi que l'analyse et l'interprétation des documents. ITESOFT est leader sur le marché français et troisième sur le marché européen dans le domaine de dématérialisation des flux d'informations. Cette entreprise est en perpétuelle recherche d'innovations technologiques. Elle investit d'ailleurs massivement afin d'avoir les meilleurs produits. C'est donc grâce à un partenariat avec l'équipe READ du LORIA que cette thèse a été proposée dans le cadre d'un contrat CIFRE.

L'objectif de cette thèse est d'automatiser le traitement des documents administratifs. Ceci permet non seulement un gain de productivité pour les administrations et les services administratifs, mais aussi un gain de réactivité et de temps de réponse pour l'usager.

Nous allons définir ce qu'est un document administratif, ce qu'il contient, ce que nous cherchons à extraire et finalement les contraintes qui nous sont imposées pour la définition de notre système.

1.1.1 Document administratif

Un document administratif est un document textuel qui est reçu et traité par une administration ou un service administratif. Nous classons les différents types de documents administratifs, selon la fonction de l'information qu'ils contiennent :

- documents décrivant une transaction : facture, bon de commande, chèque, ordre de virement, contrat de travail, courriers

[1] Caisse d'Allocations Familiales
[2] Caisse Primaire d'Assurances Maladie

- documents décrivant une prise de renseignement : questionnaire, formulaire, courriers
- justificatifs de situation : carte d'identité, carte de séjour, passeport, permis de conduire, un relevé d'identité bancaire
- annexes : informations non pertinentes pour l'utilisateur, textes juridiques sur le verso du document

Un flux de documents peut être organisé de deux manières. La première est le flux homogène, dans quel cas tous les documents du flux se ressemblent. Par exemple, les bons de commandes que reçoit une société de vente par correspondance constituent un flux de documents homogènes puisque la seule différence entre un bon et un autre est le remplissage effectué par l'utilisateur.

Un document peut aussi faire partie d'un flux de documents hétérogènes, dans quel cas il ne ressemble pas aux documents précédents ou suivants. Par exemple, une CAF reçoit tous les jours des milliers de dossiers clients composés chacun de différents documents : pièce d'identité, relevé bancaire, quittance de loyer. Le traitement de ces documents n'est pas le même que dans le cas d'un flux homogène.

1.1.2 Informations contenues dans un document administratif

L'observation des documents administratifs montre que quelque soit le flux de documents considéré, ceux là partagent les structures d'informations suivantes :

- des informations labellisées, par exemple, le label "total" suivi de l'information "12,56". Le label donne la sémantique de l'information et l'information associée sert à l'expliciter
- des informations non labellisées : ce sont des informations qui peuvent être interprétées sans avoir besoin d'une autre information complémentaire, par exemple : date, nom de l'entreprise, code de sécurité sociale...
- des tableaux simples avec des lignes et des colonnes bien définies, comme le cas de la facture dans figure 1.1
- des listes d'information, comme des détails de vente de plusieurs articles
- des question-réponses : question posée dans le document et réponse de l'utilisateur
- un texte en langage naturel : il n'y a pas d'information explicite à extraire mais il faut une interprétation du texte. Par exemple, dans un courrier de résiliation, il faut pouvoir interpréter la phrase "veuillez résilier mon contrat téléphonique".

1.2 Contraintes

1.2.1 Freemind : logiciel phare d'ITESOFT

Freemind est un logiciel de LAD commercialisé par ITESOFT. Ce logiciel permet de lire des images binaires, de les rétroconvertir via plusieurs moteurs de rétroconversion tels que des OCR("optical character recognition") ou des ICR ("intelligent character recognition"). Il permet aussi de segmenter et nettoyer des images.

Ce logiciel offre deux approches : il permet d'abord de modéliser des documents manuellement. L'utilisateur crée un masque de document dans lequel il peut préciser les endroits où il veut chercher l'information. Ceci est effectué quand les documents traités sont homogènes. Avec une telle modélisation, l'extraction de l'information devient par la suite très rapide. L'inconvénient de cette modélisation est qu'elle est longue (de l'ordre de quelques heures) et fastidieuse à faire. Elle n'est pas non plus adaptée au traitement de flux de documents hétérogènes.

Ce logiciel permet aussi de traiter des documents dans un flux hétérogène. Il en extrait les informations textuelles les plus pertinentes grâce à un autre moteur appelé FullText. Dans ce cas

aussi, une modélisation préalable des informations textuelles présentes dans les documents doit être effectuée. Elle est longue à faire (de l'ordre de quelques jours) et elle consiste à renseigner une base de connaissances.

A travers ces deux possibilités qu'offre le logiciel Freemind, nous avons constaté que l'organisation et l'apprentissage des informations, pour tout type de flux de documents, est la pierre angulaire de tout traitement ultérieur. Nous nous sommes donc fixés comme objectif d'automatiser la modélisation de ces informations, quel que soit le flux considéré.

1.2.2 Qualité de l'information

La qualité de l'information est très importante dans les systèmes d'analyse de documents. Elle dépend d'abord de la qualité du document original. Ainsi, si l'image est dégradée, l'OCR donne un résultat de lecture très dégradé. Elle dépend aussi de la performance de l'OCR ou de l'ICR utilisé. L'OCR utilisé par ITESOFT donne aujourd'hui un résultat de 99% de bons résultats au niveau caractère sur des documents de très bonne qualité, 85% de bons résultats sur des documents de qualité moyenne, et 70% de bons résultats sur des documents de qualité médiocre.

Dans un flux homogène, l'OCR peut être adapté afin de mieux lire les documents en cours. Ceci ne peut, par contre, pas être fait sur un flux de documents hétérogène puisque les documents et leurs qualités sont variables. Il est donc nécessaire de prendre en compte ces informations lors de la conception du système. L'exemple de la figure 1.2 montre du document de qualité très dégradée, sur lequel l'OCR ne sera pas performant.

1.2.3 Approche à proposer

L'approche à proposer doit :

- analyser et interpréter aussi bien les informations labellisées que non labellisées, ainsi que les tableaux et les listes
- ne pas avoir de phase d'apprentissage longue, et ne pas faire appel à l'utilisateur lors de l'apprentissage. Nous avons donc cherché à utiliser un mode de raisonnement qui s'adapte, ou qui adapte son comportement, en fonction du problème posé. Après avoir envisagé les différentes méthodes d'intelligence artificielle, nous avons opté pour une méthode pouvant s'adapter, généraliser, apprendre en continu et ne pas remettre en question ce qui a été appris auparavant, le Raisonnement à Partir de Cas (RàPC).
- s'appuyer sur le socle Freemind. Ce logiciel va d'abord lire les documents, dactylographiés ou manuscrits, et nous fournir le résultat de la lecture. Nous nous plaçons donc en aval de la phase de lecture automatique. Il est cependant à préciser que nous ne nous intéresserons pas aux textes en langage naturel.
- absorber le bruit et la variation des informations dans les documents administratifs afin de minimiser l'erreur en sortie du système

1.2.4 Démarche retenue

Avant de détailler la solution proposée, nous allons d'abord nous intéresser aux travaux existants et proches de notre problématique. Nous allons orienter l'état de l'art vers deux grandes catégories de travaux : l'analyse des documents adminisitratifs et le raisonnement à partir de cas.

La première catégorie de travaux concerne l'analyse et le traitement des documents administratifs. Nous allons explorer les travaux portant sur l'extraction des informations (identifiants, tableaux, logos...), et ce, quel que soit le flux de documents traité (homogène ou hétérogène).

Dans ce cas, ces informations sont extraites indépendamment les unes des autres. Nous notons ici que les travaux sur l'extraction des tableaux sont majoritaires dans la littérature. A notre connaissance, aucun système n'est aujourd'hui capable de traiter toutes ces informations en même temps. Nous appellerons ces systèmes les systèmes basés sur les données.

Nous allons aussi étudier les systèmes qui extraient des informations après une phase de modélisation des documents. Ces systèmes peuvent traiter n'importe quel type de documents administratifs à condition qu'ils soient modélisés au début. Nous distinguerons dans ces travaux les systèmes à modélisation manuelle des systèmes à modélisation automatique. Bien que ces systèmes soient basés sur une modélisation préalable, il n'en reste pas moins qu'ils sont largement dépendants du domaine d'application, et qu'un paramétrage est nécessaire avant toute utilisation. Nous appellerons ces systèmes les systèmes se basant sur un modèle.

La deuxième catégorie des travaux étudiés concerne les travaux du raisonnement à partir de cas (RàPC). En effet, ce paradigme de raisonnement permet non seulement de modéliser les informations automatiquement, mais aussi de pouvoir généraliser à partir de peu d'exemples . Nous nous intéresserons plus particulièrement aux travaux du RàPC textuel, puisque les informations que nous traitons sont aussi textuelles, ainsi qu'au RàPC en traitement d'images. Ces deux domaines du RàPC sont ceux qui s'approchent le plus de notre application.

L'approche que nous proposons essaie donc de prendre en compte les contraintes imposées par l'application, l'état de l'art de l'analyse de documents administratifs et celui du RàPC. Cette approche est guidée par le modèle et fonctionne selon deux cycles de raisonnement à partir de cas.

Le premier cycle est actionné lorsque le système traite des documents déja modélisés. Il permet non seulement de traiter des informations labellisées et non labellisées, mais aussi de traiter des tableaux ou des zones tabulaires.

Le deuxième cycle traite des documents dont le modèle est inconnu du système. Ce deuxième cycle est très avantageux puisqu'il permet de traiter des documents complètement nouveaux au système, en les analysant partie par partie. Dans ce cas aussi, les informations labellisées et non labellisées sont traitées, ainsi que les zones tabulaires.

1.3 Plan de la thèse

Ce mémoire est organisé comme suit. Dans le deuxième chapitre, nous présentons l'état de l'art des travaux en analyse de documents. Cet état de l'art est nécessaire pour savoir où se situent les vrais défis en analyse de documents administratifs, où se placer pour apporter une contribution à ce domaine, quels sont les problèmes non encore ou partiellement résolus ?

Dans le troisième chapitre, nous présentons un état de l'art des travaux en RàPC. Nous ne ferons pas un état de l'art général sur le RàPC mais nous nous focaliserons uniquement sur deux applications du RàPC, à savoir le RàPC en traitement et analyse d'images, et le RàPC textuel. Notre application utilisant aussi bien le texte que l'image du document, nous nous plaçons parfaitement au milieu de ces applications.

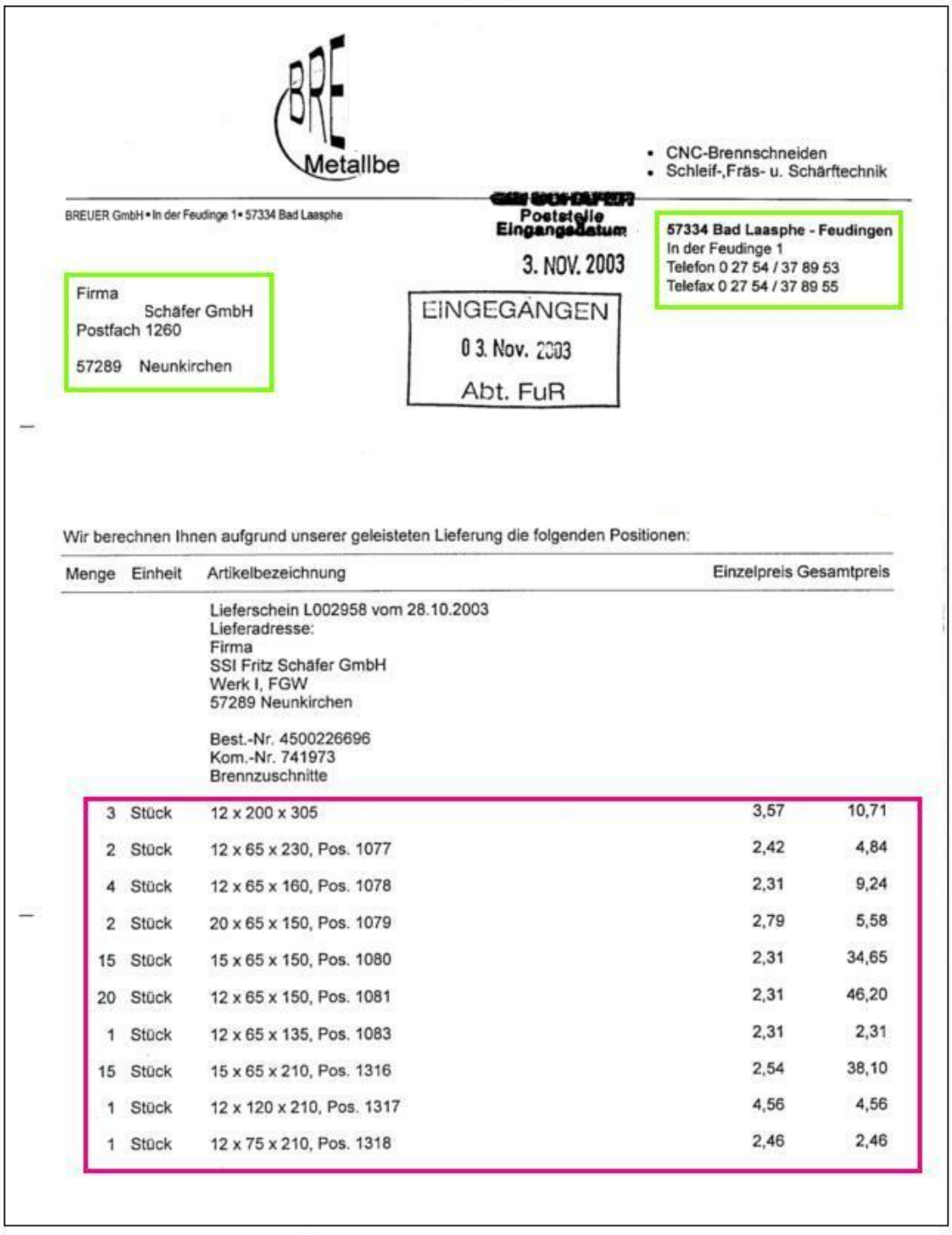
Nous introduisons notre approche d'analyse de documents utilisant le RàPC dans le quatrième chapitre. Nous y présentons les grandes lignes de notre système que nous avons appelée CBRDA (Case-based Reasoning for Document Analysis).

Notre approche nécessite une modélisation préalable du document qu'on appelle élaboration du problème en RàPC. Nous détaillons cette étape dans le cinquième chapitre. Cette élaboration permet d'extraire les caractéristiques physiques et logiques du document afin de constituer un document "requête" à traiter par le système. Ce document est modélisé par un graphe.

Dans le sixième chapitre, nous présentons la recherche des cas proches ainsi que l'adaptation, au sens RàPC. Cette phase est celle qui va nous permettre de comprendre le contenu du document. Elle se base sur la comparaison avec des documents déja traités et interprétés et l'utilisation des informations associées aux documents les plus proches. Nous aborderons aussi dans ce chapitre l'utilisation de mesures de similarité entre graphes et nous expliquerons notre choix.

Finalement, nous exposerons dans le dernier chapitre la méthode utilisée pour classer les documents. Partant du constat que le système proposé traite de plus en plus de documents et que ceux-là doivent être retenus et mémorisés lorsqu'ils sont complètement nouveaux, il devient alors inévitable de classer la base contenant tous ces documents de manière à optimiser son accès et sa recherche. Nous proposons dans cette thèse d'utiliser les réseaux de neurones non supervisés incrémentaux afin de classer les bases de documents. Nous adaptons donc un algorithme déja existant pour l'appliquer à la classification incrémentale de graphes.

Nous terminons ce mémoire par une conclusion générale et des perspectives.



BRE Metallbe

- CNC-Brennschneiden
- Schleif-,Fräs- u. Schärftechnik

BREUER GmbH • In der Feudinge 1 • 57334 Bad Laasphe

Poststelle
Eingangsdatum
3. NOV. 2003

57334 Bad Laasphe - Feudingen
In der Feudinge 1
Telefon 0 27 54 / 37 89 53
Telefax 0 27 54 / 37 89 55

Firma
Schäfer GmbH
Postfach 1260
57289 Neunkirchen

EINGEGANGEN
03. Nov. 2003
Abt. FuR

Wir berechnen Ihnen aufgrund unserer geleisteten Lieferung die folgenden Positionen:

Menge	Einheit	Artikelbezeichnung	Einzelpreis	Gesamtpreis
		Lieferschein L002958 vom 28.10.2003 Lieferadresse: Firma SSI Fritz Schäfer GmbH Werk I, FGW 57289 Neunkirchen Best.-Nr. 4500226696 Kom.-Nr. 741973 Brennzuschnitte		
3	Stück	12 x 200 x 305	3,57	10,71
2	Stück	12 x 65 x 230, Pos. 1077	2,42	4,84
4	Stück	12 x 65 x 160, Pos. 1078	2,31	9,24
2	Stück	20 x 65 x 150, Pos. 1079	2,79	5,58
15	Stück	15 x 65 x 150, Pos. 1080	2,31	34,65
20	Stück	12 x 65 x 150, Pos. 1081	2,31	46,20
1	Stück	12 x 65 x 135, Pos. 1083	2,31	2,31
15	Stück	15 x 65 x 210, Pos. 1316	2,54	38,10
1	Stück	12 x 120 x 210, Pos. 1317	4,56	4,56
1	Stück	12 x 75 x 210, Pos. 1318	2,46	2,46

FIG. 1.1 – Exemple d'une facture classique, avec des zones d'adresses (encadrées en vert), et un tableau (encadré en rose)

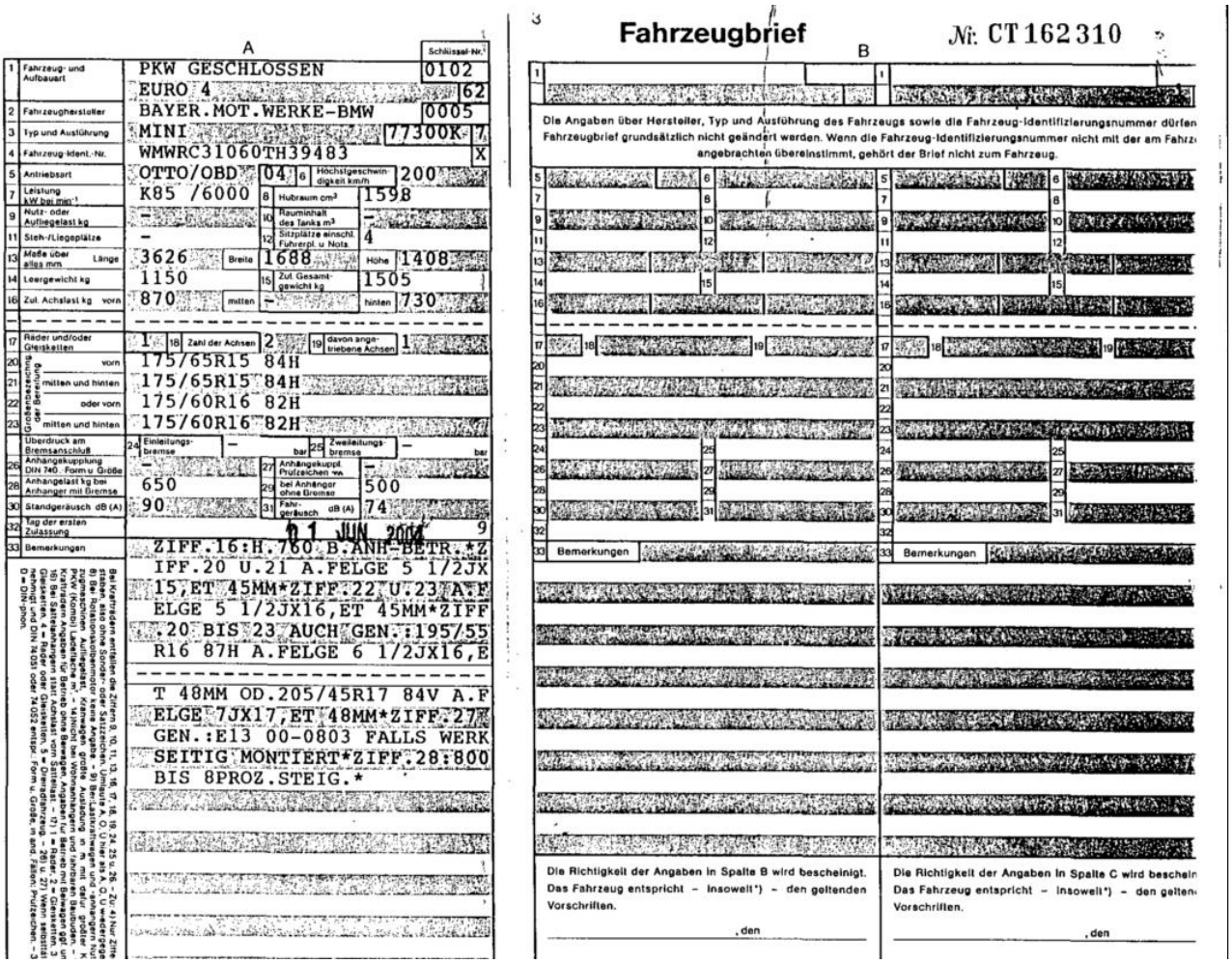


FIG. 1.2 – Exemple d'un formulaire avec une qualité du document est assez dégradée.

2

Analyse de documents administratifs

Sommaire

L'analyse de documents administratifs débute toujours par l'opération de capture d'image, suivie de la reconnaissance optique de caractères. Ces deux opérations sont indispensables pour préparer le traitement ultérieur. La capture d'image est faite par des scanners qui ont aujourd'hui des performances très élevées en termes de vitesse de défilement, de vision des pages et de qualité de numérisation. La lecture optique de caractères est effectuée par des OCR qui, bien que très performants, restent très tributaires de la qualité du support papier et de la qualité de l'image produite par le scanner.

Tous les systèmes rencontrés dans la littérature passent donc par ces deux étapes (capture d'image et OCR). Il s'ensuit plusieurs traitements, chacun orienté selon le besoin. D'après la littérature, on peut distinguer deux voies de recherche dominantes :

- La première est celle qui consiste à traiter les documents à la volée, sans se baser sur un modèle de connaissances prédéfini, mais en prenant en compte uniquement les caractéristiques du document (analyse guidée par les données). Ce type d'approches peut être adopté dans des cas où le flux de documents en entrée du système est hétèrogène. On n'a donc aucune information sur le document suivant le document en cours de traitement. Dans le cadre des systèmes développant cette voie de recherche, nous incluons les systèmes d'extraction de tableaux, d'adresses et de mots-clés.
- La deuxième voie de recherche identifiée est celle qui consiste à modéliser les documents ainsi que les informations liées aux domaines d'application, en amont de tout traitement (approche dirigée par le modèle). Dans ce cas, non seulement les informations propres aux documents traités sont utilisées, mais celles des documents de la même classe le sont aussi.

Quel que soit le document traité et comme précisé dans le chapitre précédent, certaines informations doivent être extraites. La figure 2.1 montre plusieurs exemples d'informations à extraire d'un document administratif. Les mots encadrés en vert correspondent à des mots-clés. Ils sont des repères surtout dans les factures et les formulaires. Ils permettent de trouver des

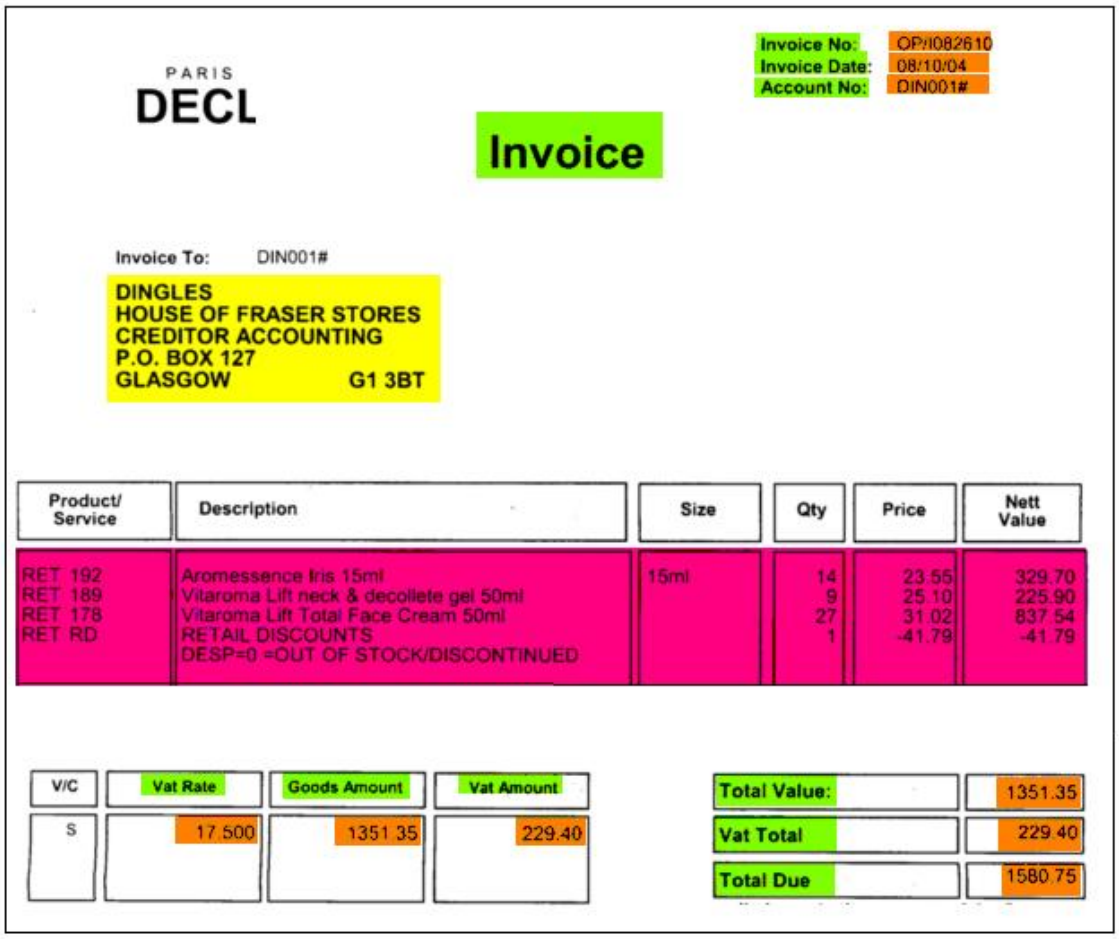


FIG. 2.1 – Exemples d'informations à extraire par les systèmes d'analyse de factures et de formulaires

informations telles que le montant total associé à une facture ou la date de livraison d'un client. Les adresses (en jaune) sont souvent des éléments très recherchés dans un document administratif car elles permettent de trouver l'expéditeur et le destinataire du document. Le tableau est aussi un élément crucial dans un document administratif.

Dans la suite, nous présenterons d'abord les travaux en analyse de documents administratifs se basant sur les données, puis les travaux se basant sur les modèles.

2.1 Systèmes guidés par les données

Ces systèmes cherchent à analyser chaque document en étudiant ses caractéristiques, mais sans utiliser un modèle. Ils possèdent l'avantage de ne pas dépendre d'une classe de documents, et de pouvoir analyser et traiter des documents très différents. Concernant ces systèmes, nous avons constaté que les travaux sur la localisation et l'extraction des tableaux restent majoritaires dans la littérature.

2.1.1 Systèmes utilisant des données génériques

Les systèmes basés sur les données peuvent aussi utiliser des données génériques communes à un ensemble de documents, ou à un domaine d'application.

Mao [MLM97] a proposé un système de lecture de coupons de vols se basant sur un OCR spécifique qui reconnaît les mots des coupons. Ce système utilise ensuite une base de mots (liée au domaine du voyage) pour corriger les mots mal reconnus ainsi que les champs d'informations associés. Il n'y a pas dans ce système de modèle général du coupon, mais plutôt un ensemble d'informations liées aux mots qui peuvent y être présents.

Srihari [SSRL95] a, quant à lui, proposé un système d'extraction et d'interprétation de blocs d'adresses dans les fiches d'impôts. Ce système passe par deux phases : la première consiste à extraire les blocs d'adresse. La deuxième consiste à reconnaître le contenu de ces blocs. La reconnaissance du contenu des blocs est effectuée grâce à une base de noms de villes et de codes postaux. L'adresse extraite est comparée à une base d'adresses afin de la vérifier.

La différence entre les deux systèmes précédents réside dans le fait que le premier extrait des informations isolées (relatives à des mots-clés), alors que le deuxième extrait des informations relatives à des structures de documents.

Cesarini [CFGS03] traite des factures de différentes origines en utilisant un domaine de connaissances indépendant des classes de documents traitées. Ces connaissances sont appelées "Class Independant Domain Knowledge" (CIDK). CIDK est composé d'informations générales sur les structures logiques du document. Par exemple, pour le mot-clé "total", on trouve tous ses synonymes, ses fréquences d'apparition dans des zones du document, ainsi que des informations relatives à sa taille moyenne, sa position moyenne, et la position de l'interprétation qui lui est associée. Les informations du CIDK restent très générales, bien qu'elles soient synthétisées à partir d'informations relatives à des classes de documents. Elles ne donnent par exemple pas de positions absolues pour les informations modélisées.

Le système proposé par Belaïd [BB98] utilise des perceptrons multi-couches afin de classer les cellules d'un document. Il commence d'abord par extraire les lignes horizontales et verticales ainsi que les cellules. Les composantes connexes à l'intérieur des cellules ainsi que les caractéristiques des cellules sont ensuite extraites et utilisées pour la classification. Le premier étage de classification utilise un perceptron à une couche permettant de séparer les cellules en plusieurs classes : cellule vide, cellule grisée, cellule noire, cellule contenant du texte aligné horizontalement, cellule contenant du texte aligné verticalement... Les deux étages de classification suivants permettent de séparer ces données en utilisant les caractéristiques des composantes connexes. Parmi les caractéristiques utilisées, nous citons : la densité des pixels dans une cellule, la hauteur moyenne des CC dans une cellule, le nombre de CC...Même si une phase d'apprentissage est nécessaire, chaque document est traité indépendamment des autres sans utilisation de modèle général de document.

Ce dernier système ne convient pas à l'approche que nous voulons proposer dans cette thèse. En effet, même si les résultats de reconnaissance produits par un réseau de neurones peuvent être excellents, cela nécessite un temps d'apprentissage long. De plus, si un nouveau type d'information apparaît dans les documents en cours de traitement, tout l'apprentissage doit être refait. Cette solution ne répond donc pas à notre problématique.

2.1.2 Localisation et segmentation de tableaux

Les tableaux sont des structures très fréquentes dans les documents administratifs. Ils décrivent des transactions entre un fournisseur et un acheteur, des articles achetés, leurs prix

CODE ARTICLE	DESIGNATION	QTE	QTE TOTALE	P.V. UNITAIRE	P.V. TOTAL	CODE TVA
902990956	CAPSORB 35 L	3	105,00	0,790	82,95	1
907990470	TKO 3.75 L	8	2,00	60,140	120,28	1
902830309	MOTORECLAT 1 AEROSOL	12	1,00	56,780	56,78	1
900911100	DP 1000 CHAMOIS: ROULEAU	6	6,00	5,480	32,88	1

Qty	Item no.	Description	EAN-Code	Unit Price	Disc. %	Amount
5	3092	Sponge Chief	4973167030929	4.14		20.70
5	3319	Cleansing Cream, 125ml	4973167033197	17.16		85.80
6	3397	Creamy Soap, 125ml	4973167033975	17.16		102.96
1	3438	Sensai EX La Lotion, 150 ml	4973167034385	83.34		83.34
6	3601	IB 10 Seconds Awakening Essence, 40ml	4973167036013	20.81		124.86
7	96582	SCP Advanced Recovery Concentrate, 40ml	4973167965825	75.52		528.64
10	96850	SCP Advanced Recovery Concentrate for Eyes, 15ml	4973167968505	49.48		494.80
4	96927	Sensai CP ARC Cell-Refining Cream, 40ml	4973167969274	75.52		302.08

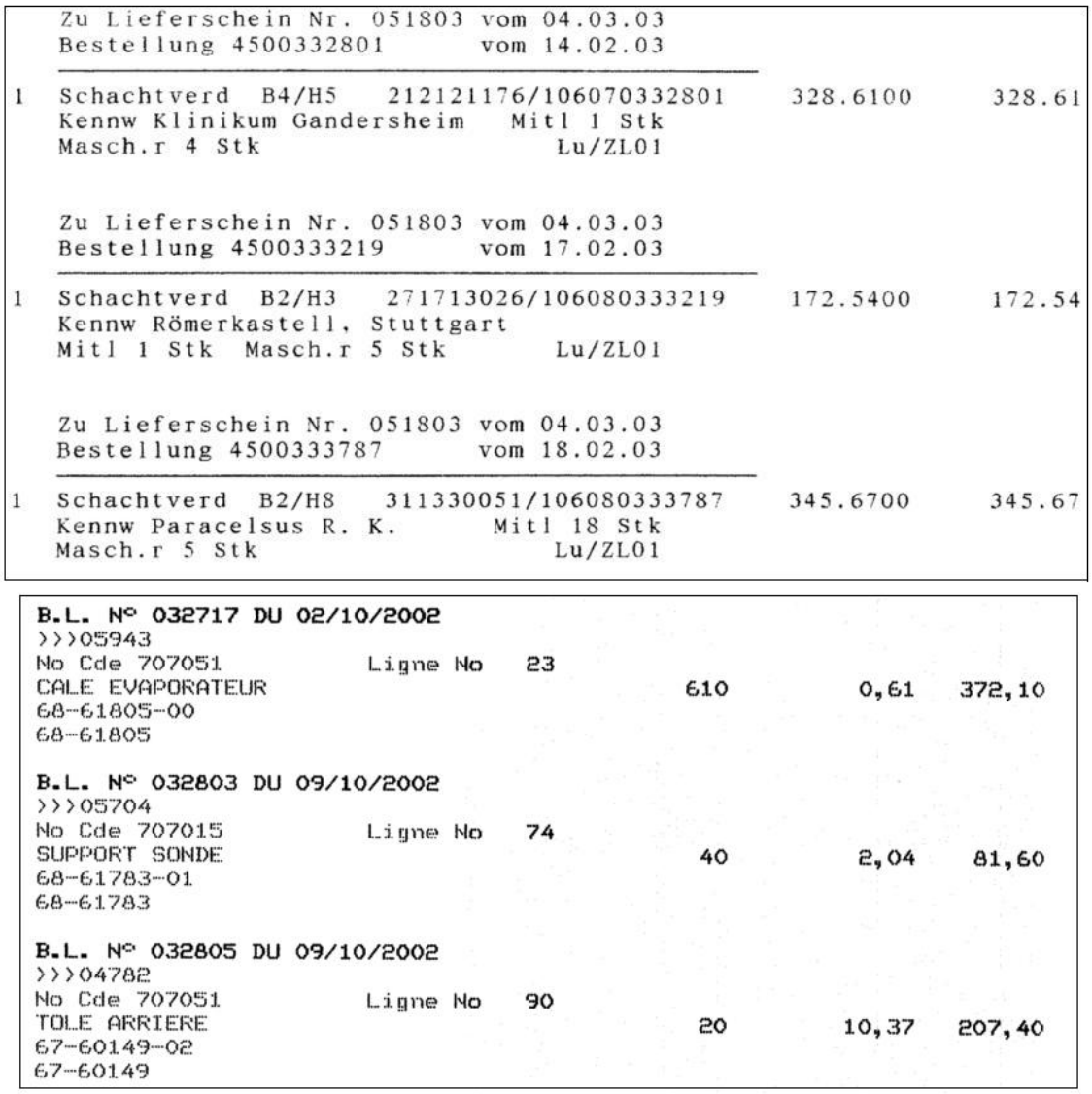
N° Article	Désignation du produit / EAN produit / BL	Prix Brut / PCE / Code article client	R1 et R2 en %		Prix Net / PCE / Quantité carton	Qté PCE	Prix Total / Prix net au carton
N° Commande Client : 86085606		N° Commande Eurodough: 57839					
EAN : 3564700215005	BL : 80067993				Qté Car : 7		Prix net/Car : 4,80
300080	TABLIER BLANC BLOC BRISE MGV 300G 12CT	4,00 / 10	0,00	0,00	4,00 / 10	36	14,40
EAN : 3564700215012	BL : 80067993				Qté Car : 3		Prix net/Car : 4,80
300173	TABLIER BLANC ROUL SABLEE MGV 230G 12CT	6,22 / 10	0,00	0,00	6,22 / 10	96	59,71
EAN : 3564700214992	BL : 80067993				Qté Car : 8		Prix net/Car : 7,46

FIG. 2.2 – Des exemples de tableaux

unitaires et totaux... De ce fait, il est indispensable de les traiter dans toute chaîne d'analyse de documents administratifs. La variation des représentations des tableaux les rend assez difficiles à extraire ou à interpréter. Dans les figures 2.2 et 2.3, nous montrons quelques exemples de tableaux. Dans la figure 2.2, le premier tableau est très simple avec une seule ligne d'article qui se répète. Le deuxième tableau comporte des lignes d'articles qui peuvent s'écrire sur une ou deux lignes. Enfin, les lignes d'article du troisième tableau s'étendent sur trois lignes. Il est beaucoup plus difficile de localiser les lignes d'articles du troisième tableau que celles du premier.

Dans la figure 2.3, il y a deux zones tabulaires qui sont très différentes des tableaux vus dans la figure 2.2. Non seulement il n'existe pas de lignes ou de colonnes physiques permettant de délimiter les zones du tableau, mais en plus, les lignes d'articles sont étalées sur 5 lignes. Nous appelons ces zones des structures à motifs composites.

La plupart des travaux en analyse de tableaux sont des méthodes descendantes (top down) qui partent de la zone du tableau avant de trouver ses lignes et cellules. Peu de méthodes sont ascendantes. La notre, présentée dans le chapitre 3 en est une. Dans [eSJT06], les différentes tâches dans le domaine de l'analyse des tableaux sont identifiées. Contrairement à [eSJT06], nous nous intéresserons uniquement aux documents de type administratif dans la description des



	Zu Lieferschein Nr. 051803 vom 04.03.03 Bestellung 4500332801 vom 14.02.03		
1	Schachtverd B4/H5 212121176/106070332801 Kennw Klinikum Gandersheim Mitl 1 Stk Masch.r 4 Stk Lu/ZL01	328.6100	328.61
	Zu Lieferschein Nr. 051803 vom 04.03.03 Bestellung 4500333219 vom 17.02.03		
1	Schachtverd B2/H3 271713026/106080333219 Kennw Römerkastell, Stuttgart Mitl 1 Stk Masch.r 5 Stk Lu/ZL01	172.5400	172.54
	Zu Lieferschein Nr. 051803 vom 04.03.03 Bestellung 4500333787 vom 18.02.03		
1	Schachtverd B2/H8 311330051/106080333787 Kennw Paracelsus R. K. Mitl 18 Stk Masch.r 5 Stk Lu/ZL01	345.6700	345.67

B.L. N° 032717 DU 02/10/2002
>>>05943
No Cde 707051 Ligne No 23
CALE EVAPORATEUR 610 0,61 372,10
68-61805-00
68-61805

B.L. N° 032803 DU 09/10/2002
>>>05704
No Cde 707015 Ligne No 74
SUPPORT SONDE 40 2,04 81,60
68-61783-01
68-61783

B.L. N° 032805 DU 09/10/2002
>>>04782
No Cde 707051 Ligne No 90
TOLE ARRIERE 20 10,37 207,40
67-60149-02
67-60149

FIG. 2.3 – Des exemples de tableaux, plus difficiles à extraire

travaux de localisation et de segmentation de tableaux.

La localisation de tableaux consiste, à partir d'un document, à détecter le début et la fin du tableau dans le document. C'est en général la première tâche qui est effectuée avant tout autre traitement. Deux approches existent dans la littérature :

- celles qui utilisent les indices physiques caractérisant les tableaux : lignes horizontales, lignes verticales, étude des intersections entre ces lignes ;
- celles qui utilisent les indices logiques associés aux tableaux tels que leurs entêtes, ou des relations entre les lignes ou les colonnes des tableaux

Systèmes utilisant les indices physiques

Ce premier type d'approches utilise les caractéristiques physiques de l'image : pixels, projections, lignes horizontales, verticales, points d'intersections entre les lignes... pour retrouver les tableau dans les documents administratifs.

La transformée de Hough est utilisée pour la détection de lignes de tableaux dès 1995 par [TBB95b] et [RSE+98]. Le principe de [TBB95b] est le suivant : les lignes horizontales et verticales sont cherchées par transformée de Hough. Les intersections de ces lignes sont par la suite recherchées. Un graphe est constitué à partir de ces intersections (les intersections des lignes étant les noeuds, et les arcs les segments liant les intersections). Les cellules du tableau correspondent aux cycles minimaux de ce graphe.

Une autre méthode est proposée dans [GDPP05]. Elle utilise des opérateurs de morphologie mathématique (ouverture, fermeture, érosion, dilatation) sur une image binaire afin d'extraire les lignes. Ces opérateurs permettent de connecter des lignes parfois interrompues ou discontinues. L'extraction des lignes est affinée par la suppression des zones textes et/ou image. Il ne reste donc à la fin que les vraies lignes verticales et horizontales caractérisant le tableau.

D'autres approches utilisent une grammaire pour retrouver la structure tabulaire du document. Par exemple, Couasnon [Coü06] propose de localiser des tableaux en utilisant une grammaire (EPF) et un analyseur associé. La grammaire EPF (Enhanced Position Formalism) décrit le document. Il est nécessaire d'avoir un analyseur, qui à partir des informations données par la grammaire, va extraire les tableaux se trouvant dans les documents. Bien que cette approche ait été validée sur un très grand nombre de documents anciens, elle n'a pas, à notre connaissance, été testée sur des documents administratifs (à part visuellement). De plus, bien qu'elle soit générique et pouvant s'adapter à tout nouveau type de documents, son inconvénient majeur réside dans le fait que l'utilisateur doit lui même formaliser la grammaire relative au type de documents avant de commencer l'extraction des informations. Ceci n'est pas une tâche facile surtout pour un utilisateur novice.

Les arbres M-X-Y sont utilisés dans [CMSS02] et nécessitent un apprentissage. En partant de la description du document par un arbre M-X-Y (traduisant la présence de lignes horizontales, verticales, de lignes blanches séparant le texte), l'algorithme proposé parcourt l'arbre et essaie de trouver les tableaux présents dans le document. Les paramètres de cet algorithme sont optimisés grâce à un apprentissage sur un corpus de documents étiquetés. Lorsque ce processus est achevé, il devient beaucoup plus facile d'extraire les tableaux dans le corpus de test. Les résultats de cette méthode sont meilleurs que ceux données par deux OCR commerciaux.

Les approches proposées dans ce paragraphe ne peuvent cependant pas être utilisées directement dans le cadre du système que nous proposons. Nous sommes, en effet, contraints d'utiliser uniquement les informations textuelles extraites à partir de l'image. Nous ne pouvons pas utiliser les informations au niveau pixel. Nous montrons dans le chapitre 5 la nature des informations dont nous disoposons.

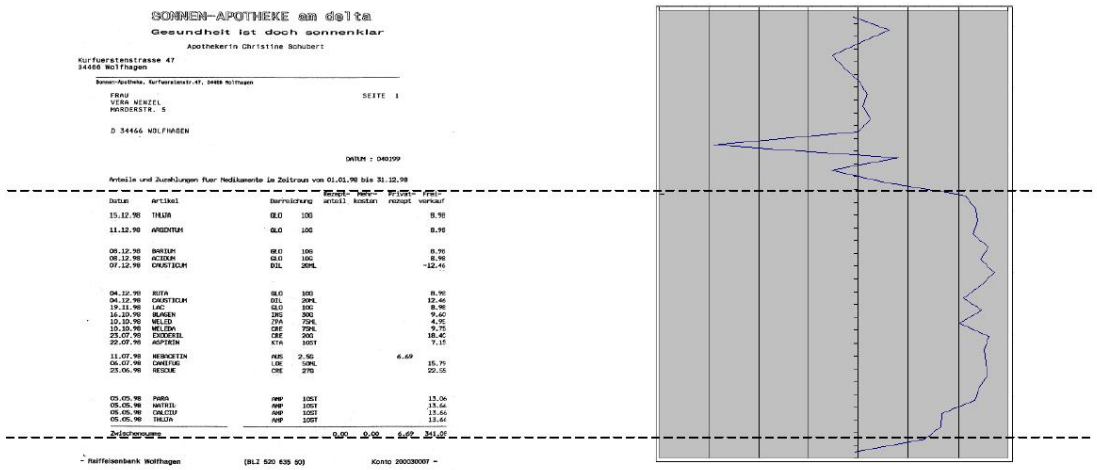


FIG. 2.4 – Exemple d'un document avec le signal de corrélation associé. Source [KGKD01]

Systèmes utilisant des indices logiques

Le deuxième type d'approches consiste à utiliser des indices d'un plus haut niveau telles que les entêtes des tableaux, utilisés aussi bien dans [KGKD01] que dans [WT98]. Les entêtes des tableaux sont des éléments très discriminants dans la phase de recherche de tableaux. S'il n'est pas toujours facile de les trouver, il n'en demeure pas moins que s'ils sont bien détectés, nous pouvons être sûrs de localiser toutes les lignes du tableau.

Dans [KGKD01], Klein utilise les entêtes comme première solution pour la localisation des tableaux. En fait, l'extraction des entêtes fonctionne comme dans [WT98] uniquement si des entêtes similaires existent dans une base d'entêtes liée au système d'extraction. Cela veut dire que les entêtes extraits sont comparés avec une base d'entêtes de tableaux validée.

La deuxième approche de Klein consiste à considérer toutes les lignes du document et à essayer de trouver celles qui ont une forte corrélation entre elles. Ceci est effectué en considérant que chaque ligne est représentée par un signal : un mot représentant un signal positif, et un espace représentant un signal négatif. Cette approche cherche à trouver les signaux les plus corrélés et à les étudier pour en extraire les tableaux. La figure 2.4 montre l'exemple d'un document, où chaque ligne est modélisée par un signal, et où la corrélation entre deux lignes successives est affichée sous la forme d'un signal (figure à droite). Nous pouvons constater que la zone tablulaire est celle qui a la plus grande corrélation entre ses lignes.

2.1.3 Systèmes de segmentation et d'interprétation des tableaux

L'interprétation d'un tableau consiste à donner une sémantique à chaque colonne ou ligne du tableau (colonne des prix, colonne des totaux...).

L'un des rares travaux ayant traité ce problème est un travail de l'équipe READ [BB04] portant sur les factures. A partir de fichiers ASCII contenant les mots présents dans un tableau déja extrait, le but est de retrouver les lignes, colonnes et champs d'articles du tableau. La technique utilisée est basée sur l'étiquetage par partie de discours. Après un étiquetage primaire des mots par leur catégorie morphologique, des champs sont regroupés et identifiés. L'étude de leur régularité et redondance permet de segmenter les factures en articles et de structurer ces articles. Le modèle extrait est utilisé pour corriger les articles erronés.

Le but de cette approche donc est de délimiter les lignes d'articles et de donner une sémantique à chaque élément dans le tableau. Le pourcentage d'articles reconnus est de 91.02% et celui des champs bien étiquetés (quantité, prix unitaire, désignation, ...) de 92.56%. Cette approche est donc prometteuse pour une interprétation de tableaux. Cependant, son inconvénient majeur réside dans le fait que le tableau doit déja être extrait.

2.1.4 Bilan

Vu l'importance du facteur temps dans les applications industrielles, il serait, à notre avis, préférable de combiner une approche basée sur les propriétés physiques de l'image avec une autre utilisant les propriétés logiques. En effet, l'extraction par utilisation des lignes horizontales, verticales et des intersections peut permettre de bien extraire les tableaux les plus simples, ceux qui sont délimités naturellement par une "grille ".

Si les tableaux présents dans le document ont des structures complexes, d'autres approches prenant en compte le résultat de l'OCR (mots) et la caractérisation des mots (mots-clés, entêtes, motifs de lignes...) peuvent être utilisées. Ce deuxième type d'approche fonctionnerait alors en remplacement ou en complément du premier type. Il serait en mesure de localiser des tableaux plus complexes, que la géométrie seule n'arrive pas à retrouver. D'ailleurs, Klein [KGKD01] développe ce type d'argument pour son application industrielle.

Dans la littérature, nous avons remarqué que la plupart des travaux font de la localisation de tableaux en même temps qu'ils font de la segmentation de cellules et de lignes dans les tableaux. Ceci est particulièrement vrai dans les approches ascendantes (qui partent du pixel pour arriver au tableau, avant de redescendre au niveau des cellules). Certains travaux font au contraire de la localisation de cellules de tableaux avant de localiser le tableau en entier. C'est le cas de [Ito93].

2.2 Systèmes guidés par les modèles

Ces systèmes, contrairement aux précédents, se basent sur un modèle du document afin d'extraire les informations. Utiliser un modèle permet d'accélérer les traitements puisque le système est guidé par le modèle pour extraire l'information. Le modèle peut être construit automatiquement ou manuellement, sachant que la majorité des approches de la littérature sont manuelles. Si plusieurs modèles sont possibles, il faut classer le document à traiter pour connaître le modèle à utiliser.

Modélisation des documents

La modélisation des documents consiste à extraire un modèle pour chaque document ou lot de documents. Ce modèle est établi à partir d'observations effectuées sur un ensemble de documents très similaires et couvre donc au maximum les variations, voire distorsions, qui existent d'un document à l'autre. Cette modélisation, si elle est manuelle, est toujours faite grâce à une interface qui est fournie à l'utilisateur, et dans laquelle celui-ci va modéliser les informations à extraire en utilisant leurs coordonnées et natures dans le document. Si par contre, la modélisation est automatique, elle est faite grâce aux caractéristiques physiques et logiques qui sont extraites du document.

La modélisation automatique de documents utilise une extraction d'indices physiques à partir de l'image, puis un modèle est construit à partir de ces indices. C'est le cas de plusieurs travaux tels que celui de Shimotsuji [SA96] qui extrait automatiquement des cellules de tableaux présentes

dans le document et utilise les coordonnées des centres de ces cellules pour former un vecteur représentatif de chaque document.

Mao [MAM96] extrait les lignes horizontales et verticales, les intersections entre ces lignes, les logos, le texte et les cellules de tableaux et forme un modèle grâce à tous ces éléments. Finalement, il trouve que les informations relatives aux lignes horizontales et verticales sont suffisantes pour trouver la classe du document.

Ting [TL96] propose une autre approche de modélisation automatique, qui le conduit à représenter le modèle du document par une chaîne de caractères. Il commence par extraire les éléments suivants : le texte, les lignes, les distances verticales entre deux éléments du document, les indentations, et les intersections (surtout les coins). Chaque élément extrait est représenté par un caractère, ce qui fait que le document est finalement représenté par une chaîne de caractères qui suit l'ordre de lecture humaine (de gauche à droite et de haut en bas). Le modèle d'un document contenant deux blocs de texte (notés T) séparés par un espace (noté E) serait donc représenté par la chaîne "TET".

Héroux [PSAE98] propose deux modélisations : la première utilise une décomposition pyramidale de l'image par calcul de densité de pixels. Cette décomposition produit un vecteur de taille 341 qui décrit la densité en pixels sur plusieurs niveaux de découpage. La deuxième est un arbre dans lequel on transcrit les informations de l'image d'une manière descendante (Méta blocs→ blocs→ lignes→ mots...).

Duygulu [DA02] extrait récursivement des blocs jusqu'à atteindre le niveau des cellules dans un formulaire. Il utilise des lignes horizontales et verticales pour retrouver ces blocs. Le modèle final du document est une représentation hiérarchique en arbre fortement inspirée de la représentation en arbre X-Y.

D'autres auteurs n'utilisent pas la structure physique pour représenter les documents, mais ils utilisent plutôt les mots-clés (qui sont parfois assimilés à des structures logiques). C'est le cas de Sako [Ish01] qui extrait les mots-clés à partir du document puis considère que le modèle du document est l'ensemble de ces mots-clés. Notons ici que les relations entre les mots-clés ne sont pas prises en compte dans le modèle du document, seule leur présence est considérée.

Nous retrouvons aussi l'utilisation des mots-clés dans le cadre de la modélisation manuelle de documents. C'est le cas de Cesarini [CGMS98], qui utilise en plus des mots-clés, d'autres zones d'intérêts comme les logos et les lignes horizontales et verticales. Tous ces éléments sont extraits manuellement. Le modèle final proposé est un graphe étiqueté et orienté. Ishitani [Ish01] utilise aussi les mêmes éléments (à part le logo) pour modéliser ses documents puis les classer.

Finalement, l'approche de modélisation manuelle la plus connue consiste à extraire le modèle du document à partir d'un document vierge (c'est surtout le cas des formulaires). Nous ne pouvons pas citer dans ce mémoire tous les systèmes utilisant cette approches. A titre d'exemples, celles de Arai [AO97] et Tang [TL97] extraient les lignes et les cellules du document en essayant d'être le plus robuste possible aux variations de l'inclinaison. Bien que ces approches semblent dépassées, il n'en demeure pas moins qu'elles continuent à être utilisées dans l'industrie.

Les approches de modélisation manuelle se basent sur des interfaces qui permettent de sélectionner les zones d'intérêt dans le document, et de les associer avec les autres zones déja extraites. La modélisation manuelle est une tâche très fastidieuse et longue, surtout si on traite des documents provenant de classes très variées. L'utilisateur a la possibilité de corriger, d'ajouter des informations, de commenter...

Les approches automatiques sont plus intéressantes car elles sont indépendantes d'un utilisateur. Elles essaient d'extraire les informations liées aux structures physiques (lignes horizontales, verticales, intersections) ou logiques (mots-clés) afin de proposer un modèle final du document. L'avantage de ces méthodes est certainement la rapidité, mais l'inconvénient majeur est le risque

de proposer un modèle erroné à cause de l'absence d'une vérification par l'utilisateur.

L'utilisation des informations liées aux lignes horizontales, verticales, intersections...(structure physique du document) peut être avantageuse si le document est bien scanné. Par contre, dès qu'une variation de ces lignes se produit, la reconnaissance de la classe de document peut facilement échouer. Une structuration de plus haut niveau peut donc être utile pour absorber ces variations, d'où le recours à l'utilisation d'une description hiérarchique des blocs [DA02] au lieu d'utiliser uniquement les lignes. L'autre niveau de description consiste à utiliser des informations telles que les mots-clés. Ils sont facilement repérables et une fois reconnus, constituent un moyen très apprécié pour modéliser un document. Leur utilisation a d'ailleurs fait ses preuves dans beaucoup de travaux [CGMS98], [Ish01]...

La représentation du modèle du document est en soi un domaine assez vaste. Certains auteurs se contentent d'utiliser un vecteur ou une liste pour stocker les informations présentes dans un document, mais de telles représentations et surtout dans les documents administratifs (où le texte ne couvre jamais plus de 50% de la page), ne prennent pas en compte l'agencement des informations dans le document, qui est une information très importante.

Une fois le modèle du document extrait, les systèmes avec modèles cherchent à l'associer à un modèle déjà traité. La finalité de cette étape dépend vraiment de l'application du système. Certains systèmes se contentent de classer le document comme c'est le cas des applications cherchant à séparer les documents (séparer factures, formulaires, lettres), mais la plupart des applications cherchent à extraire des informations de ces documents en fonction de leurs classes.

On trouve ainsi dans la littérature plusieurs manières de classer un document administratif :

- le kppv, un classifieur classique utilisé par [PSAE98] avec des documents représentés en vecteurs décrivant une pyramide de densité de pixels dans l'image, ou celui de [SMF+03] qui décrit ses documents en fonction des mots-clés qui y sont présents. Une autre forme de kppv utilisant une distance entre chaînes est proposée dans [TL96]. En effet, comme le document est modélisé par une chaîne de caractères, il suffit de comparer les chaînes représentant les documents pour trouver la classe la plus proche. Le kppv a aussi été utilisé sur des arbres [PSAE98] modélisant des documents en utilisant la distance d'édition puis le plus grand graphe commun (que nous détaillons dans le chapitre 6).
- le Perceptron multicouches, utilisé par [PSAE98]. Une phase d'apprentissage des classes de documents est d'abord effectuée sur une base de documents puis le PMC est utilisé pour identifier la classe d'un document. Pour ce type de classification, le système doit disposer d'un temps d'apprentissage, ce qui n'est pas toujours possible. De plus, pour que ce type de classification fonctionne et ne produise pas que du rejet, soit le nombre de classes est figé dès le début, soit il faut refaire l'apprentissage dès le début à chaque fois que le système rencontre une nouvelle classe de documents.

Interprétation des documents

L'interprétation des documents consiste à donner une sémantique à chaque élément du document. Par exemple, le mot clé "rue" peut être un bon indicateur pour trouver l'adresse. Cependant, cette information est insuffisante, car il faut aussi associer le numéro de rue et le nom de la rue. Si ceci n'est pas fait, l'extraction de "rue" n'a plus d'intérêt.

A travers la littérature, nous avons pu identifier deux directions dans l'interprétation de documents :

- il y a d'abord les méthodes qui traitent les documents dont le modèle est construit à partir d'informations logiques (mots-clés). Ce modèle indique en général l'endroit ou la méthode à employer pour interpréter une information. Par exemple, le modèle indique

qu'il faut chercher l'interprétation de "total" à droite et après une distance de 20 pixels entre "total" et son interprétation. Les systèmes cherchent donc cette interprétation, en prenant en compte des paramètres comme : l'inclinaison de la page, la distance entre le champ d'information à extraire, l'information à interpréter et la nature de l'information recherchée. Ceci est l'exemple des travaux de Cesarini [CGMS98].

– il y a ensuite les méthodes qui utilisent les modèles basés sur la structure physique du document. C'est le cas des modèles produits à partir de documents pré imprimés. Les méthodes déployées essaient d'extraire les informations d'un document rempli [AO97]. Ces informations, qui ne sont toujours pas lisibles par l'OCR lors de leurs extractions, doivent être restaurées par les systèmes avant d'être passées à l'OCR.

La tendance actuelle dans la recherche en analyse de documents administratifs est vers une confiance de plus en plus grande dans les OCRs (vu que les scanners sont plus performants, de même que les OCRs), et un plus grand intérêt dans la manière de laquelle on doit modéliser un document ou le classer. L'interprétation n'est évoquée de manière très précise que dans quelques publications telle que [CGMS98] [CFGS03].

Dans [CFGS03], l'interprétation des éléments logiques du document (mots-clés) est effectuée grâce à des connaissances :

– relatives à la classe du document en cours de traitement
– relatives au domaine des factures. Ce sont des informations très génériques (évoquées dans le paragraphe 2.1)

Lorsque la classe du document est connue, il est naturellement plus facile d'extraire aussi bien les mots-clés que leurs interprétations grâce aux informations spécifiques liées à cette classe. Cependant, si cette extraction échoue, ou que le document traité est inconnu, alors seules les connaissances génériques peuvent permettre d'extraire aussi bien les mots-clés que leurs interprétations.

Ce travail [CFGS03] est très intéressant. Il propose ainsi une solution à l'analyse et l'interprétation de documents de types connus et inconnus. Cependant, il possède quelques limites :

1. la capacité de généralisation de ce système n'est pas vraiment démontrée. Les tests établis sur 138 documents sont insuffisants pour pouvoir aspirer à une grande généralisation. De plus, si un cas de facture complètement nouveau se présente et que les informations présentes dans les bases de connaissances (spécifique et générique) ne couvrent pas ce cas précis, il devient dès lors très difficile de trouver une interprétation aux mots-clés de cette facture. Ceci est dû au fait que chaque mot-clé est analysé indépendamment des autres. Une prise en compte du contexte de chaque mot-clé (l'ensemble des mots-clés qui lui sont associés ou proches), devrait lui permettre de mieux généraliser. C'est ce que nous proposons de faire dans notre travail.

2. les domaines de connaissances ne semblent pas être mis à jour automatiquement. D'ailleurs, il faut une base d'apprentissage pour les constituer. Par la suite, il faudrait normalement prendre en compte les nouvelles classes de documents qui sont traitées, incorporer les connaissances qu'ils contiennent et ce, afin de rendre le système capable de traiter des documents similaires dans le futur. Un apprentissage incrémental est nécessaire pour mettre à jour les bases de connaissances (ou domaines de connaissances). Nous nous sommes aussi proposé de faire ce type d'apprentissage dans cette thèse.

3. ce système ne traite pas les tableaux. Ceci est presque indispensable dans tout traitement de factures.

2.3 Conclusion

Nous avons présenté dans ce chapitre les approches les plus courantes dans le domaine de l'analyse et de l'interprétation du document administratif. Nous avons d'abord détaillé les travaux se basant sur les données, puis les travaux se basant sur les modèles.

Nous avons décelé, à travers l'état de l'art effectué sur les méthodes présentées (et d'autres que nous ne pouvons pas citer dans ce mémoire à cause du manque de relation directe avec le sujet de thèse) que pour qu'un système d'analyse, de reconnaissance et d'interprétation de documents administratifs fonctionne bien et soit le plus général possible, il faut que plusieurs étages de traitements existent :

- une extraction des structures physiques du document, ces structures physiques incluant les tableaux, les blocs, les lignes horizontales et/ou verticales.
- une modélisation du document en entrée. Cette modélisation peut être automatique ou manuelle, mais il est préférable de la faire automatiquement.
- une reconnaissance de la classe du document à analyser.
- l'interprétation du document grâce aux informations intrinsèques du document ou aux informations liées à la classe du document.

Nous estimons, pour notre part, qu'une approche utilisant un modèle est plus efficace dans la mesure où elle permet une analyse plus rapide des documents dont la classe est connue. Nous pensons aussi qu'une approche de modélisation automatique est meilleure qu'une approche de modélisation manuelle.

Si les documents ne sont pas classés, et qu'ils sont tous hétérogènes, une solution doit aussi être proposée pour leur analyse et interprétation. C'est ce que propose Cesarini dans [CFGS03] avec l'utilisation d'informations générales et/ou spécifiques. Nous pensons aussi qu'un système combinant analyse de documents homogènes et analyse de documents hétérogènes permettrait de faire face à la variabilité des documents à laquelle un système d'analyse de documents peut être confronté. C'est la solution que nous proposons dans cette thèse.

3

Raisonnement à partir de cas

Sommaire

3.1 Qu'est ce que le RàPC

Le raisonnement à partir de cas (RàPC) est un paradigme de raisonnement qui utilise les expériences précédentes pour résoudre de nouveaux problèmes [AP94]. Par le terme "expériences précédentes", nous voulons dire : expériences d'analyse, de planification, de stratégie... Le champ d'application du RàPC est très vaste aujourd'hui. Il est appliqué dans tous les domaines où on a besoin d'utiliser ou de synthétiser les expériences passées afin de proposer de nouvelles solutions.

Les premiers travaux en RàPC ont été proposés en 1983 par Kolodner [Kol83], qui a créé un système de questions-réponses en utilisant une base de connaissances déjà établies. Plusieurs systèmes ont été depuis créés et on peut aujourd'hui trouver les applications du RàPC dans de nombreux domaines tels que :

- le design de systèmes [GGC07]
- le traitement et l'analyse de documents textuels [WAB06]
- l'analyse d'image [PHR06]
- la recherche médicale [LdB+03]
- les jeux vidéos, la robotique [BB07]

Nous nous référons dans la suite de ce chapitre, aux travaux effectués dans les domaines d'analyse de documents textuels et de traitement et d'analyse d'images (TAI). Ce choix est motivé par le fait que les documents administratifs, une fois dématérialisés, ne sont plus que du texte. L'OCR, en effet, renvoie des informations textuelles accompagnées d'informations de position. De plus, le travail effectué sur ce texte correspond aussi à la tâche d'interprétation d'images que l'on retrouve dans les travaux de RàPC en TAI.

Le RàPC repose sur plusieurs étapes que nous allons détailler au fur et à mesure de ce chapitre 3.1. Plusieurs modélisations ont été proposées pour définir le RàPC, mais la plus courante chez les

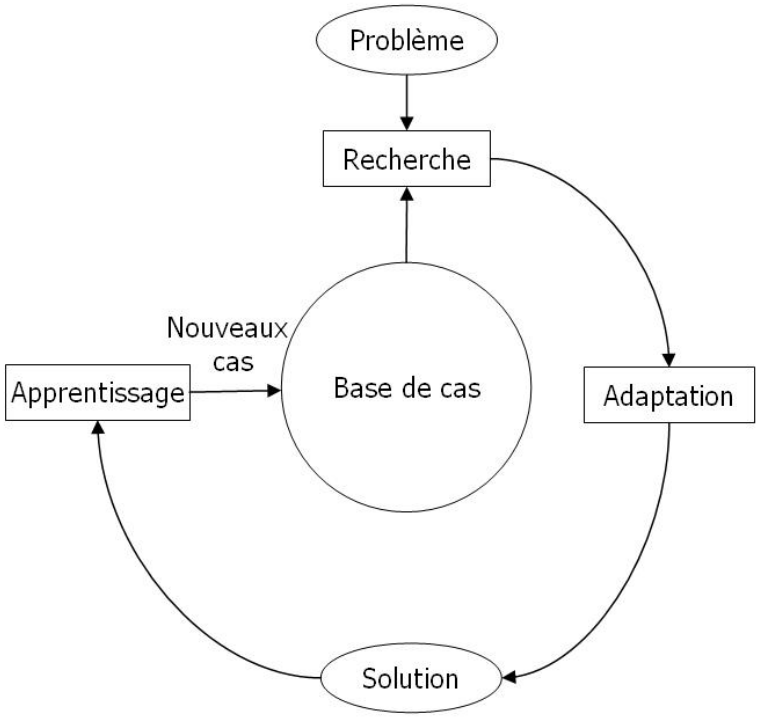


FIG. 3.1 – RàPC : les différentes étapes

chercheurs en RàPC est celle de Aamodt et Plaza [AP94]. Elle comporte 4 étapes : la recherche, la réutilisation, la révision, la mémorisation (ou l'apprentissage). Une étape préliminaire est également indispensable, c'est l'élaboration du problème. Dans cette thèse, nous avons fusionné les étapes de réutilisation et de révision en une étape appelée classiquement "adaptation". Ceci se fait d'ailleurs dans la plupart des travaux en RàPC.

Terminologie du RàPC

En RàPC, un problème est posé par l'utilisateur ou par le système. C'est la partie à résoudre. Un cas est l'ensemble formé par le problème et sa solution.

$$cas = \{Probleme, Solution\}$$

Dans cette thèse, nous ferons l'abus de langage suivant (qui est courant) : "résoudre un problème" c'est "résoudre un cas". De plus, nous considérons que le problème le plus proche correspond au cas le plus proche.

On appelle cas cible le cas à résoudre et cas source le cas de la base de cas servant à résoudre le cas cible.

La base de cas est l'ensemble des cas du système. Ceux-ci sont soit donnés par l'utilisateur, soit par le système qui s'enirchit automatiquement.

Elaboration du problème

Cette phase consiste en l'extraction d'indices, de descripteurs du problème posé et à leur représentation. Elle peut être faite soit par l'utilisateur qui, à partir de ses données, va constituer

un problème à résoudre, soit par le système qui sait lui même élaborer un problème en fonction des données dont il dispose.

Par exemple, dans le système FAQFinder [BHK95], la donnée du système est une question posée par l'utilisateur. La question est par la suite traitée par le système qui en extrait le problème (extraction des termes les plus significatifs, élimination des termes les moins informatifs). Le problème final dans FAQFinder est donc un vecteur représentatif de la requête.

Recherche de cas proches

La recherche de cas proches est une des parties les plus importantes du RàPC. Elle implique de prendre en compte la représentation des cas, une mesure de similarité entre cas et le choix du ou des cas proches. La mesure de similarité entre cas se réduit le plus souvent à une mesure de similarité entre problèmes. Elle dépend directement de la représentation des données et de leur complexité. Une fois la mesure de similarité adéquate au problème est choisie, il s'agit alors, pour chaque nouveau problème, de retrouver les cas proches dans la base de cas.

Adaptation

L'adaptation consiste à trouver une solution au problème cible à partir d'une solution donnée au problème source. La solution du problème source doit être gardée ou modifiée pour permettre la proposition d'une nouvelle solution au problème cible. Deux grandes catégories d'adaptation existent dans la littérature :

- l'adaptation structurelle : elle essaie d'appliquer la solution du cas source via certains changements prenant en compte les différences entre les problèmes source et cible, sur le problème cible.
- l'adaptation dérivationnelle, qui retrace la manière dont la solution source a été proposée pour produire la solution pour cible. Ceci est particulièrement vrai pour les applications de planification par exemple.

Ces deux grandes catégories peuvent être déclinées en plusieurs types d'autres adaptations [WM94]. La plus simple, et c'est celle que nous allons utiliser, est l'adaptation nulle. Elle consiste, à partir de la solution du problème source, à "coller" la solution sur le problème cible. Cette adaptation, même si elle est très simple, permet de résoudre des problèmes, comme le cas de documents par exemple.

Apprentissage

L'apprentissage consiste à intégrer les cas nouvellement résolus et révisés dans la base de cas. Ce processus doit être conçu pour éviter de remplir la base avec chaque nouveau cas résolu (la redondance peut être dangereuse pour le système dans la mesure ou elle peut le ralentir considérablement). Il permet au système de gérer ses nouvelles connaissances. Nous proposons dans cette thèse une solution intéressante de classification de la base de cas. Nous nous plaçons dans le cas où la base de cas est constamment alimentée par de nouveaux cas résolus différents, et nous essayons de construire un système capable de classer ces cas de manière à avoir toujours un accès rapide à la base.

Dans la suite de ce chapitre, nous allons présenter certains travaux de RàPC. Nous nous intéressons au RàPC textuel et au RàPC en analyse et traitement d'images. Nous détaillerons au fur et à mesure les trois étapes citées (recherche, adaptation et apprentissage).

3.2 RàPC textuel et RàPC en traitement et analyse d'images

Le RàPC textuel est une branche du RàPC qui s'intéresse particulièrement aux données textuelles (représentées sous forme de textes). Il se base donc sur des cas décrits dans des textes, le problème et la solution sont donc du texte. Il est facile au lecteur d'imaginer que la plupart des applications de traitement de l'information ainsi que celles de la fouille de texte peuvent aussi se retrouver dans des travaux de RàPC textuel.

Le RàPC en traitement et analyse d'images (TAI) a été, pour sa part, un domaine de recherche très étudié à partir de 1995. Le traitement et l'analyse d'images existent depuis longtemps, certes, mais l'utilisation de méthodes de RàPC n'a commencé à voir le jour qu'à partir de 1995 avec le système de Macura [MM95] qui a étudié des images de radiologie.

Les RàPC textuel et le RàPC en TAI comme tous les autres systèmes utilisant le RàPC, traitent les problèmes suivants :

- la représentation des données. Ici, les données sont des textes ou des images qui peuvent être traitées dans leur état brut ou sous forme de vecteurs, de matrices de pixels, de graphes, de résumés...
- la mesure de similarité entre cas, qui dépend de la représentation des données
- l'adaptation des solutions des cas sources aux cas cibles
- l'apprentissage et la maintenance de la base de cas.

Dans le RàPC textuel, les applications sont nombreuses, nous en citons quelques unes qui nous ont paru être très intéressantes :

- FAQFinder [BHK95] est un système de questions réponses utilisant une FAQ. L'utilisateur entre une requête sous forme de questions, et le système répond en donnant les éléments textuels qui satisfont la requête.
- PRUDENTIA [Web99] est un système de recherche dans une base de cas légaux. Les cas juridiques similaires à un cas sont proposés à l'utilisateur.
- ECUE [DB07] est un filtre anti SPAM. L'entrée du système est un courrier électronique. La sortie est une réponse : Spam, non Spam.
- SMILE [BA05] est un système de réponse à des cas de conflits judiciaires. Une base de cas est utilisée pour proposer une réponse aux cas de conflits posés au système.

La richesse des documents varie en fonction de l'application, des textes de loi dans SMILE ou PRUDENTIA sont beaucoup plus riches et complexes que des courriers électroniques.

Le RàPC en TAI peut être appliqué à tous les étages du traitement et d'analyse d'images : à partir de l'étape de la capture jusqu'à l'étape d'interprétation de l'image. Petra Perner [Per01] a publié des travaux dans chacune de ces phases de traitement. C'est d'ailleurs l'auteur qui a le plus publié sur le RàPC en TAI. Des applications du RàPC en TAI sont proposées dans la capture de l'image (choix des meilleurs paramètres de capture), la binarisation et/ou la segmentation des couleurs d'une image et pour arriver finalement à l'interprétation des différentes parties d'une image.

3.2.1 Représentation des données

Dans le RàPC textuel

On part toujours d'un texte qui, dans ce cas, peut être étudié de plusieurs manières. La représentation des données se confond ici avec l'élaboration du problème. L'approche la plus courante consiste à extraire, à partir du texte en entrée (question, texte de loi...), un vecteur représentatif de ce texte. Ce vecteur n'est autre que la représentation tf*idf (term frequency, inverse document frequency) du document en entrée. Cette représentation prend en compte la

fréquence d'un terme dans le document (tf), pondérée par le nombre de documents contenant ce terme par rapport au nombre de documents total. Cette représentation, très utilisée aussi dans le domaine de la recherche d'informations, est surtout utilisée en RàPC textuel quand il s'agit de résumer une requête ou un texte par ses mots-clés. Ceci permet de connaître a priori le thème du texte, à condition que la représentation tf*idf prenne en compte tous les mots-clés présents dans le texte. On retrouve cette représentation dans les systèmes FAQFinder [BHK95] et Drama[LW99]. Utiliser une représentation vectorielle peut être très efficace si l'ordre des mots dans le document n'a pas d'importance, ce qui n'est pas le cas de tous les documents textuels.

D'autres approches de représentation des cas par des graphes ou des arbres existent aussi mais sont moins fréquentes. Une des premières tentatives a été [CWP+04] qui a proposé d'utiliser des graphes pour représenter les cas. D'ailleurs, la représentation en graphes a permis de donner de meilleurs résultats que la représentation vectorielle lors de la phase de recherche.

Nous ne détaillerons pas d'autres modèles de représentations, puisque les seuls qui nous intéressent sont les deux modèles cités.

Dans le RàPC en TAI

La représentation des cas se fait en fonction de l'application étudiée. On remarque l'existence de trois types de représentations :

- une représentation prenant l'image au niveau pixel en entrée du système. Cette représentation est aussi couplée avec d'autres informations telles que la manière dont l'image a été capturée, le matériel et le paramétrage nécessaire à l'obtention de cette image. On dispose donc d'informations image et d'informations non-image qui donnent une richesse certaine à l'information en entrée. Ce type de représentation peut être adéquat si l'application vise à traiter directement les pixels de l'image (segmentation, binarisation). Un exemple d'utilisation de cette représentation est l'application développée par Perner pour la binarisation d'images [Per00]. Un autre exemple d'application utilisant les pixels de l'image comme représentation des cas est [PB04] qui utilise les contours des objets pour les reconnaître.
- une représentation qui ne prend en compte que des descripteurs statistiques de l'image tels que la moyenne (des couleurs ou niveau de gris), l'écart type, la variance,... Ce type de représentation peut aussi être utile pour des applications de segmentation ou de binarisation, voire de classification. Le contenu de l'image (la nature des objets contenus dans l'image) importe peu ici, seuls les descripteurs comptent. Frucci [FPdB07] a récemment proposé un système de segmentation (par la méthode watershed) d'images par utilisation de descripteurs statistiques. Perner a proposé de coupler ces paramètres statistiques avec des informations sur la manière dont l'image a été capturée [Per99] (L'application étudiée est la recherche de paramètres de segmentation d'images). De telles informations permettent d'enrichir la description des problèmes.
- une représentation de plus haut niveau, modélisant le contenu de l'image par des graphes attribués, ou des arbres. Ce type de représentation suppose que le contenu de l'image est la partie la plus importante. Dans ce cas, les descripteurs de type couleur, prise de l'image, appareil, ne sont pas pris en compte. Ceci est surtout utilisé quand on cherche à interpréter le contenu l'image en fonction d'autres images de la base. C'est le cas de [Per98a] qui représente les parties segmentées de l'image par un graphe afin de pouvoir l'interpréter par la suite.

D'après ce qui précède, la représentation des données est fonction de l'application. Perner dit dans [Per01] qu'après avoir testé les représentations à base de pixels et la représentation utilisant les descripteurs statistiques, dans un but de segmentation d'images, les résultats donnés par la

première approche sont meilleurs que ceux donnés par la deuxième. Ceci peut s'expliquer par le fait que dans le cas de la segmentation ou de la binarisation, on peut avoir deux images ayant exactement les mêmes descripteurs (moyenne, écart type, entropie, inclinaison...) et nécessitant deux segmentations différentes. Par contre, si on choisit la représentation par pixels, l'utilisateur peut être certain du résultat puisque aucune information n'est perdue et que les mesures de similarité utilisées sont relatives à une information de plus bas niveau (pixel, couleur, position).

Lorsqu'il s'agit de tâches d'interprétation (comme dans le cas de notre application), la première étape consiste à extraire des objets à partir de l'image, puis à les représenter sous forme de graphe, d'arbre ou de réseau. Comme l'interprétation utilise généralement les objets présents dans l'image et n'utilise pas directement les pixels, l'obligation de représenter uniquement les objets de plus haut niveau s'impose naturellement.

Un autre critère de choix de représentation est la capacité de stockage de la base de cas. Il est évident qu'une représentation de plus haut niveau requiert moins d'espace qu'une représentation ou chaque image doit être stockée intégralement dans la base de cas.

3.2.2 Mesure de similarité

Le choix de la mesure de similarité entre problèmes dépend essentiellement de la méthode de représentation. Une mesure de similarité couramment utilisée dans le RàPC textuel est la distance Cosinus. Etant donné deux vecteurs X et Y, cette mesure s'écrit :

$$d(X,Y) = \frac{X \cdot Y}{\|X\| \, \|Y\|}$$

Ici, c'est l'angle entre les deux vecteurs qui est étudié. Si deux vecteurs sont orthogonaux, alors cette mesure est nulle, si les deux vecteurs sont opposés, alors cette mesure donne -1.

Dans les systèmes utilisant les représentations en graphe, des mesures de similarités inter-graphes sont utilisées. Certaines de ces mesures seront détaillées dans le chapitre 4.

Perner dans [Per01] a classé les différentes mesures de similarité qu'on peut utiliser pour comparer des cas en RàPC appliqué au TAI. Si les cas sont les pixels des images, alors la proximité est mesurée par des distances entre images (distance de haussdorff par exemple). Si l'image est représentée par ses descripteurs (statistiques, capture ...), alors une distance entre vecteurs s'impose. Finalement, si les cas représentent des relations entre objets de l'image (représentation en arbres ou en graphes), des distances inter-graphes ou inter-arbres seront retenues (voir chapitre 4).

3.2.3 Adaptation

L'adaptation dans le RàPC textuel se base essentiellement sur la solution apportée par le cas source. Parmi les nombreuses applications de RàPC textuel, nous citons deux exemples d'adaptation :

- répondre à une requête. Cette réponse est un texte entièrement produit par le système (combinaison de plusieurs textes de la base, ou texte entier de la base). Ceci est le cas de FAQFinder, SMILE, PRUDENTIA. L'adaptation consiste à synthétiser des réponses à des requêtes similaires pour pouvoir répondre à la requête en cours ;
- répondre à un courriel, comme c'est le cas du système proposé dans [LL04]. Ce système de réponse automatique aux courriels utilise une base contenant un ensemble de cas=(courriel, réponse au courriel), pour répondre aux nouveaux courriels entrants. En fonction de la similarité, une nouvelle réponse est générée. Elle utilise les informations trouvées dans

d'autres courriels soit en les généralisant (exemple : 11/07/2007 se transforme en date) pour mieux les utiliser, soit en se basant sur des règles extraites du corpus formant la base de cas. Cette nouvelle réponse est adaptée au nouveau courriel, et n'est donc pas un modèle générique dans lequel seul le destinataire a changé (comme c'est le cas de plusieurs robots qui répondent automatiquement aux courriels).

Dans le cadre du TAI utilisant le RàPC, la solution correspond souvent aux paramètres de segmentation ou de binarisation qu'il faut adopter. A problèmes similaires, il faut donc proposer des solutions similaires. Mais si dans ce cas, la solution correspond vraiment à une démarche, voire à un paramétrage à effectuer tel que dans [FPdB07], ceci n'est pas toujours le cas. Par exemple, dans [PB04], l'adaptation consiste uniquement à trouver la classe du cas le plus proche. Aucune adaptation n'est réellement effectuée.

Dans notre conception du RàPC, il nous paraît quand même utile, voire nécessaire, de s'inspirer d'une solution proposée par le cas source et de l'adapter (même si ce n'est qu'une recopie de la solution) pour le problème cible.

3.2.4 Apprentissage

Nous nous intéressons dans ce paragraphe à un des rares travaux en RàPC en TAI ayant traité l'apprentissage. Il s'agit du travail de Perner [Per98b] pour la structuration d'une base de cas pour un système d'interprétation d'images.

Les cas présents dans ce système sont des graphes : chaque graphe décrit des objets dans l'image. Une fois qu'un problème est résolu, le cas (contenant un graphe et sa solution) doit être injecté dans la base de cas. Perner propose alors de classer la base d'une manière incrémentale en utilisant une hiérarchie (c'est en fait un arbre, où chaque noeud correspond à une classe, les noeuds terminaux sont les instances des classes). Cette classification utilise les critères de variance intra-classes et inter-classes. A chaque ajout d'un cas, on cherche la partition des données qui maximise la variance inter-classes et la variance intra-classes.

Chaque classe est représentée par son graphe médian (celui qui minimise la distance à tous les autres graphes), et les ajouts d'un nouveau cas se font alors suivant 3 types d'opérations :

- le nouveau cas est simplement ajouté à une classe existante
- le nouveau cas provoque la division d'une classe en deux autres classes (node splitting)
- le nouveau cas provoque la fusion de deux classes différentes (node merging)

Cette approche n'a cependant été testée que sur un petit nombre de cas, et n'a pas été généralisée à d'autres types de données. Notre application utilise aussi des graphes, mais nous proposerons une autre type d'apprentissage incrémental utilisant les réseaux de neurones.

3.3 Conclusion

Dans ce chapitre, nous avons présenté les différents défis qui se posent lors de la définition d'un système de RàPC, à savoir : l'élaboration du problème, la recherche des cas proches, l'adaptation et enfin l'apprentissage. Nous nous sommes uniquement intéressés aux travaux en RàPC textuel et en TAI vu leurs proximités avec notre application.

L'utilité du RàPC pour notre application paraît désormais évidente. L'utilisation des connaissances accumulées au cours des expériences précédentes peut, en effet, nous permettre de mieux résoudre les nouveaux problèmes. Il suffit pour cela de représenter le problème de manière à bien mettre en valeur les informations que nous cherchons à extraire. Il faut aussi prévoir une solution de secours au cas où un problème ne peut pas être résolu. Certains systèmes de RàPC autorisent l'intervention de l'utilisateur dans une ou dans plusieurs étapes du traitement. Nous proposons

dans ce qui suit une solution alternative qui consiste à décomposer le problème en sous problèmes lorsqu'une solution globale n'est pas trouvable.

4

Le RàPC utilisé en analyse de documents

Sommaire

Nous présentons dans ce chapitre les grandes lignes de notre approche, avant de les détailler dans les chapitres suivants.

4.1 Problèmes posés

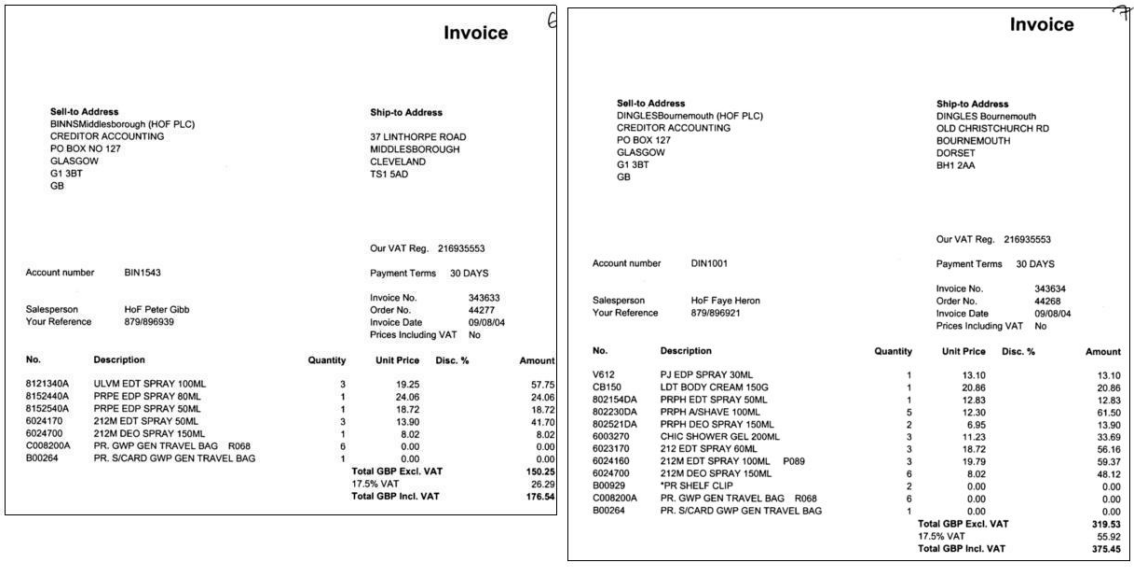
Dans une chaîne d'analyse de documents administratifs, il arrive souvent que les documents soient traités par lots. Dans cette thèse, nous appelons lot un ensemble de documents similaires, provenant d'un même fournisseur, d'une même entreprise ou administration. Les documents d'un lot ont tous les mêmes caractéristiques, ils sont représentés de la même manière et seul le contenu spécifique de chaque document est différent (par exemple, le contenu des cellules dans un formulaire). Actuellement, chez ITESOFT, un traitement par lot nécessite qu'un utilisateur intervienne pour modéliser les documents de ce lot. A partir du modèle, il devient plus facile au système d'extraire les informations recherchées dans le document.

Le premier problème est donc d'essayer de modéliser automatiquement les documents d'un lot et d'utiliser ce modèle par la suite lors des traitements par lots.

Le deuxième problème est encore plus complexe. Le système doit aussi pouvoir traiter des documents hétérogènes. Dans ce cas, deux configurations sont possibles :

- la première, et c'est la plus facile, est le cas du document pour lequel on peut trouver un modèle similaire. Ceci veut dire que des documents similaires à ce document ont déjà été traités auparavant, et cela peut donc accélérer son traitement. Les documents de la figure 4.1 ont exactement les mêmes structures. Les tableaux sont similaires, ainsi que les structures d'adresses et de paiement. Il est clair alors que si nous disposons du modèle général de ce lot, il devient alors très facile de traiter tous les documents provenant de ce lot.
- la deuxième est plus complexe. C'est le cas d'un document complètement nouveau, auquel aucun modèle existant ne peut être associé, et qui doit bien sûr être analysé. La solution la plus facile consiste à faire appel à un utilisateur, qui va extraire les informations nécessaires.

Ce n'est pas notre choix. C'est le deuxième problème de la thèse. Il consiste à essayer d'analyser les documents individuellement, sans faire appel à des modèles de documents existants. L'exemple des factures de la figure 4.2 montre deux factures provenant de deux lots différents. Il est clair que les deux factures ne doivent pas (et ne peuvent d'ailleurs pas) être traitées de la même manière. Un traitement particulier doit donc être effectué sur chacune de ces factures.



Invoice

Sell-to Address
BINNSMiddlesborough (HOF PLC)
CREDITOR ACCOUNTING
PO BOX NO 127
GLASGOW
G1 3BT
GB

Ship-to Address
37 LINTHORPE ROAD
MIDDLESBOROUGH
CLEVELAND
TS1 5AD

Our VAT Reg. 216935553

Account number BIN1543
Payment Terms 30 DAYS

Salesperson HoF Peter Gibb
Your Reference 879/896939
Invoice No. 343633
Order No. 44277
Invoice Date 09/08/04
Prices Including VAT No

No.	Description	Quantity	Unit Price	Disc. %	Amount
8121340A	ULVM EDT SPRAY 100ML	3	19.25		57.75
8152440A	PRPE EDP SPRAY 80ML	1	24.06		24.06
8152540A	PRPE EDP SPRAY 50ML	1	18.72		18.72
6024170	212M EDT SPRAY 50ML	3	13.90		41.70
6024700	212M DEO SPRAY 150ML	1	8.02		8.02
C008200A	PR. GWP GEN TRAVEL BAG R068	6	0.00		0.00
B00264	PR. S/CARD GWP GEN TRAVEL BAG	1	0.00		0.00
			Total GBP Excl. VAT		**150.25**
			17.5% VAT		26.29
			Total GBP Incl. VAT		**176.54**

Invoice

Sell-to Address
DINGLESBournemouth (HOF PLC)
CREDITOR ACCOUNTING
PO BOX 127
GLASGOW
G1 3BT
GB

Ship-to Address
DINGLES Bournemouth
OLD CHRISTCHURCH RD
BOURNEMOUTH
DORSET
BH1 2AA

Our VAT Reg. 216935553

Account number DIN1001
Payment Terms 30 DAYS

Salesperson HoF Faye Heron
Your Reference 879/896921
Invoice No. 343634
Order No. 44268
Invoice Date 09/08/04
Prices Including VAT No

No.	Description	Quantity	Unit Price	Disc. %	Amount
V612	PJ EDP SPRAY 30ML	1	13.10		13.10
CB150	LDT BODY CREAM 150G	1	20.86		20.86
802154DA	PRPH EDT SPRAY 50ML	1	12.83		12.83
802230DA	PRPH A/SHAVE 100ML	5	12.30		61.50
802521DA	PRPH DEO SPRAY 150ML	2	6.95		13.90
6003270	CHIC SHOWER GEL 200ML	3	11.23		33.69
6023170	212 EDT SPRAY 60ML	3	18.72		56.16
6024160	212M EDT SPRAY 100ML P089	3	19.79		59.37
6024700	212M DEO SPRAY 150ML	6	8.02		48.12
B00929	*PR SHELF CLIP	2	0.00		0.00
C008200A	PR. GWP GEN TRAVEL BAG R068	6	0.00		0.00
B00264	PR. S/CARD GWP GEN TRAVEL BAG	1	0.00		0.00
			Total GBP Excl. VAT		**319.53**
			17.5% VAT		55.92
			Total GBP Incl. VAT		**375.45**

FIG. 4.1 – Deux factures d'un même lot. Le modèle de ces documents est le même.

Ces deux problèmes ont soulevé d'autres problèmes, aussi importants :

- comment modéliser les documents, quelles informations doivent être représentées dans ce modèle ?
- comment profiter de l'existence de modèles de documents déjà existants ?
- quelle approche adopter pour que l'intervention de l'utilisateur soit minimale ?

La solution idéale serait alors d'essayer de profiter de l'expérience du système, au lieu de celle d'utilisateur. Au fur et à mesure du traitement, le système accumule les expériences d'analyse de documents. Au bout d'un certain temps, il est certain que le système a traité tellement de documents différents que la connaissance qu'il a acquise pourrait servir à analyser n'importe quel type de documents. Il faut donc profiter de ces connaissances. C'est l'idée principale de la solution apportée aux problèmes précédemment cités. Nous allons donc présenter un système qui non seulement traite des documents, mais qui apprend aussi au fur et à mesure.

C'est pour cette raison que nous avons choisi d'utiliser le RàPC pour le système proposé. D'abord, le mode de raisonnement existant dans le RàPC est très bien adapté aux besoins d'un système d'analyse de documents. En effet, le RàPC permet de profiter des expériences précédentes pour proposer de nouvelles solutions aux nouveaux problèmes posés. Dans le cas de notre application, un nouveau problème n'est autre qu'un nouveau document à analyser. Que ce document soit dans un lot ou soit complètement nouveau ne change rien au fait qu'il faut pouvoir l'analyser et l'interpréter. Les expériences précédentes ne sont donc autres que les documents précédemment modélisés, analysés et interprétés.

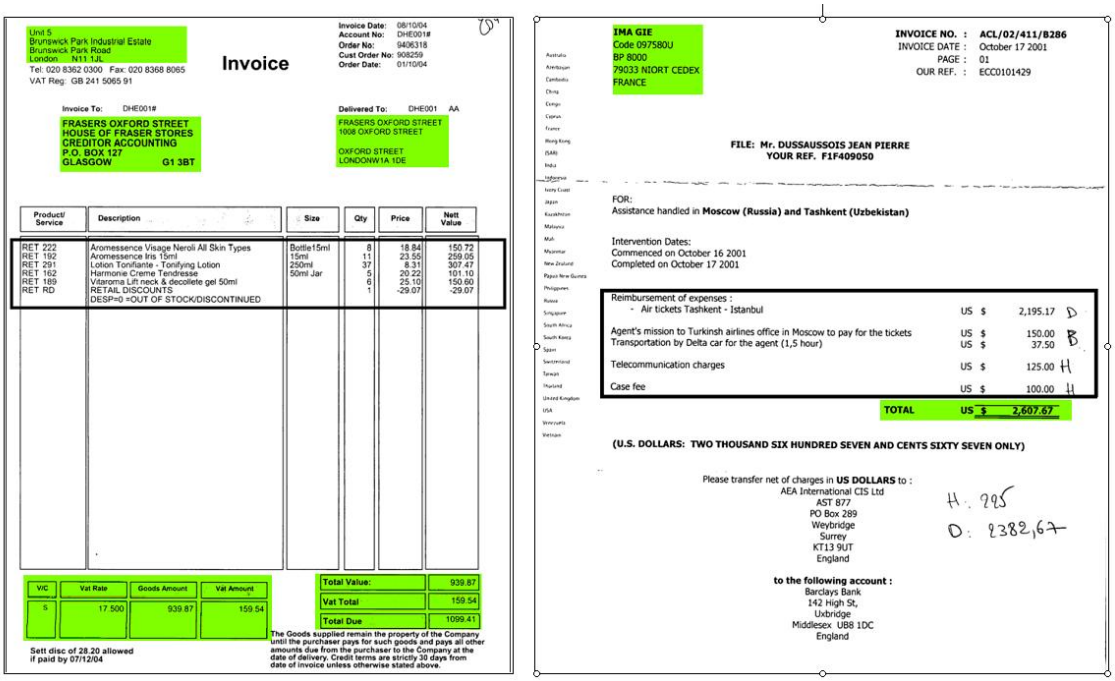


FIG. 4.2 – Deux factures provenant de lots différents.

4.2 Approche proposée

Voici donc l'approche proposée dans la figure 4.3

- Pour chaque document à traiter, on extrait le modèle du document.
- Ce modèle est comparé à une base de cas pour savoir si des documents similaires ont déjà été traités.
- Si c'est le cas, alors, il suffit d'utiliser les informations associées au modèle dans la base afin d'analyser le nouveau document.
- Sinon, l'analyse est reportée au niveau des structures du document. Nous définissons une structure du document comme étant une zone spécifique du document comme un tableau, une adresse, une zone de paiement... Ces structures sont extraites pendant la phase de modélisation du document. Le modèle du document n'est donc autre que l'agencement de ces structures. Ainsi, pour chaque structure, nous procédons de la même manière que pour le document. Nous proposons d'utiliser une base de structures, dans laquelle sont stockées toutes les structures des documents précédemment analysés. A travers nos observations de centaines de documents administratifs, nous nous sommes rendus compte que même si deux documents sont très différents, certaines similitudes peuvent exister entre leurs structures. Par exemple, les adresses françaises sont souvent écrites de la même manière, avec les noms au début, les numéros de rue et les noms de rue par la suite, pour finir le plus souvent avec le code postal et la ville. Il va de soi donc qu'analyser quelques adresses peut aider à en analyser d'autres. C'est aussi le cas des tableaux. C'est pour cela que nous proposons d'analyser le document structure par structure.

Nous traduisons dans ce qui suit notre approche en termes de RàPC.

A partir de chaque document en entrée du système, nous extrayons un problème. Ce problème correspond à toutes les informations extraites à partir du document et nécessitant une interprétation. Par exemple, extraire le mot-clé "total" n'est pas une finalité en soi si nous n'arrivons

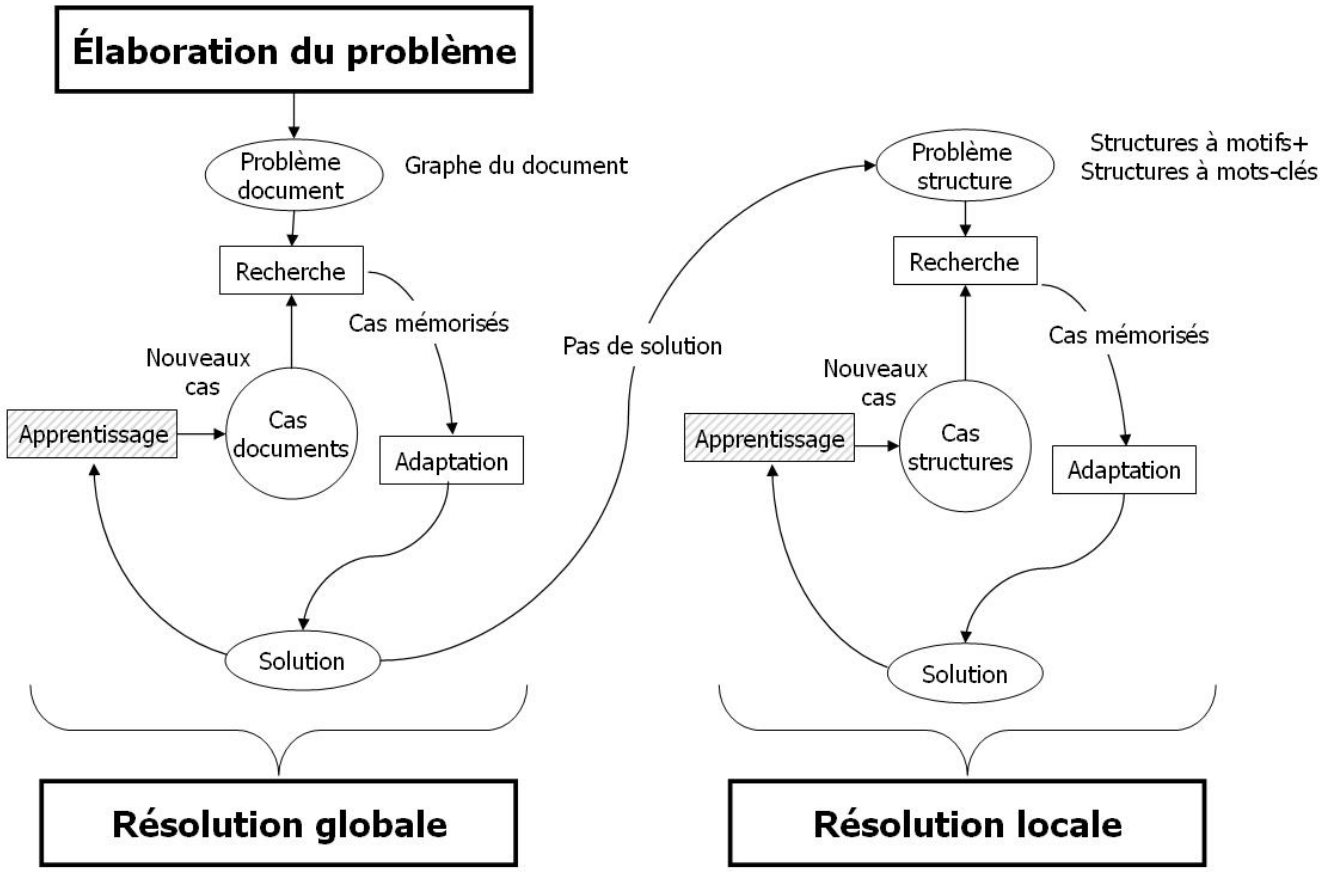


FIG. 4.3 – Approche adoptée

pas à lui associer la valeur correspondante qui est dans le document. Ainsi, le mot-clé "total" fait partie du problème du document, et la solution à cette partie du problème est le numérique qui lui associé.

Le problème du document n'est donc autre qu'un ensemble de sous problèmes. La solution la plus rapide et la plus fiable est de résoudre le problème global en entier, mais si ceci n'est pas possible, il faut résoudre le problème global en solvant ses sous problèmes un par un.

A partir de chaque document, nous extrayons plusieurs informations telles que les mots, les lignes, les zones d'adresses et les zones tabulaires. Nous regroupons toutes ces informations dans deux types de structures : les structures à motifs et les structures à mots-clés. Une structure à motif est composée d'un motif qui se répète. Un motif est la représentation d'une ligne de tableau ou de plusieurs lignes de tableau. Nous avons préféré l'appellation structures à motifs à l'appellation structures tabulaires ou tableaux. En effet, bien que les tableaux simples avec une ligne simple qui se répète plusieurs fois sont très présents dans les documents administratifs, on retrouve néanmoins d'autres types de tableaux, qui ne sont pas une répétition d'une simple ligne, mais plutôt une répétition de plusieurs blocs de lignes. La notion de motif paraît donc plus appropriée pour représenter ces zones. L'exemple de la figure 4.4 montre une structure à motif simple contenant 3 champs numériques et 2 champs alphanumériques.

V612	PJ EDP SPRAY 30ML	1	13.10	13.10
CB150	LDT BODY CREAM 150G	1	20.86	20.86
802154DA	PRPH EDT SPRAY 50ML	1	12.83	12.83
802230DA	PRPH A/SHAVE 100ML	5	12.30	61.50
802521DA	PRPH DEO SPRAY 150ML	2	6.95	13.90
6003270	CHIC SHOWER GEL 200ML	3	11.23	33.69
6023170	212 EDT SPRAY 60ML	3	18.72	56.16
6024160	212M EDT SPRAY 100ML P089	3	19.79	59.37
6024700	212M DEO SPRAY 150ML	6	8.02	48.12
B00929	*PR SHELF CLIP	2	0.00	0.00
C008200A	PR. GWP GEN TRAVEL BAG R068	6	0.00	0.00
B00264	PR. S/CARD GWP GEN TRAVEL BAG	1	0.00	0.00

FIG. 4.4 – Exemple d'une structure à motif simple. Le motif qui se répète est en vert. Il est composé de deux champs alphanumériques et de trois champs numériques.

Une structure à mots-clés est une zone du document contenant plusieurs mots-clés qui sont souvent liés à un même thème. Par exemple, dans le cas d'une structure à mots-clés représentant une adresse, les mots-clés sont : "rue", "code postal", "nom de la ville", "pays"... Ces structures à mots-clés sont très fréquentes dans les documents administratifs. Nous avons préféré les extraire et les représenter dans le problème du document plutôt que d'utiliser leurs mots-clés séparément, sans faire de liens entre ces mots.

Les sous problèmes du problème du document sont donc les problèmes de ses structures. En résolvant les problèmes des structures, nous résolvons aussi le problème du document. Nous représentons les 3 types de problèmes comme suit (voir chapitre suivant pour plus de détails) :

- le problème du document est un graphe, où les noeuds sont les structures (à mots-clés ou à motifs), et les arcs sont les positions relatives des structures les unes par rapport aux autres.
- le problème d'une structure à motif est le motif de la structure.
- le problème d'une structure à mots-clés est le graphe de ses mots-clés, où les noeuds sont les mots-clés et les arcs sont les positions relatives de ces mots-clés.

La figure 4.5 montre un exemple de graphe de document. Celui ci est composé de six structures à mots-clés, elles mêmes représentées en graphes.

Nous disposons de deux bases de cas. La première contient les cas de documents (un cas de document étant l'ensemble {problème de document, solution du document}). La deuxième contient les cas de structures.

La figure 4.3 montre notre approche. Elle comporte deux cycles de RàPC. Pour chaque nouveau document, nous en extrayons le problème. Celui-ci est comparé avec les problèmes de la base de cas de documents.

- Si un problème similaire existe dans cette base, alors la solution du problème le plus proche (problème source) est appliquée sur le problème du document cible.
- Si aucun problème similaire n'existe dans la base de cas, alors on passe au deuxième cycle de RàPC. Les problèmes des structures sont alors utilisés. Chaque problème de structure est comparé avec les problèmes de la base de cas de structures. Les solutions des problèmes les plus proches sont alors appliquées sur le problème de structure cible. Par ce processus, nous pouvons obtenir une solution complète pour le problème du document. Nous pouvons donc injecter ce cas résolu dans la base de cas de documents.

Nous détaillerons dans les chapitres suivants les différentes étapes du RàPC : élaboration, recherche de cas proches, adaptation, apprentissage et enfin, la classification de la base de cas.

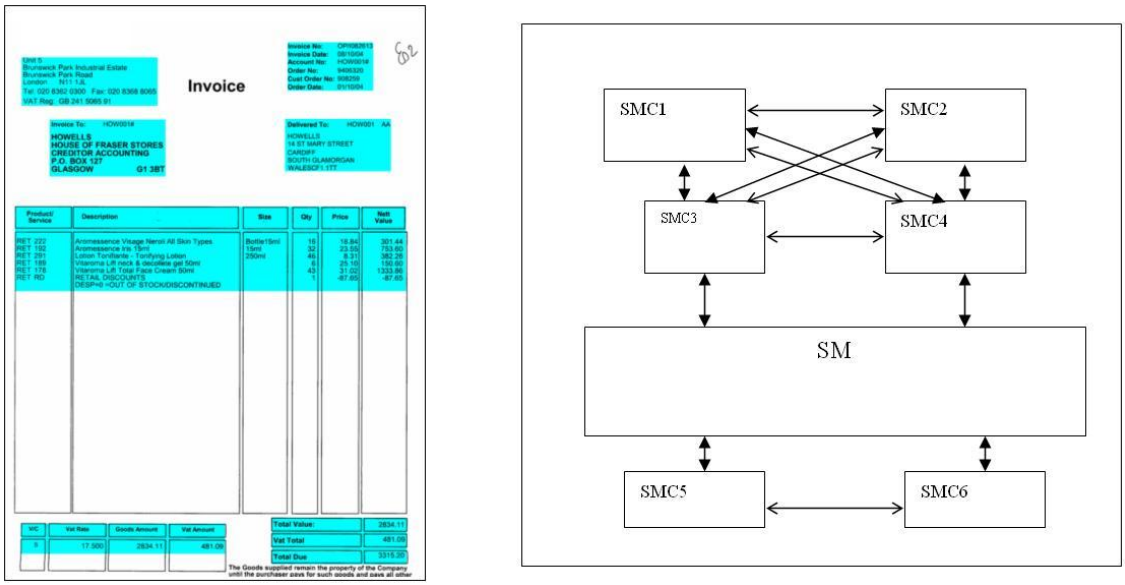


FIG. 4.5 – Un document et son problème représenté par un graphe

4.3 Conclusion

Nous avons présenté dans ce chapitre une approche d'analyse de documents utilisant le RàPC. Cette approche se résume en trois grandes parties. La première est l'élaboration du problème. Cela consiste en l'extraction automatique du modèle du document. La deuxième partie concerne la recherche de problèmes proches dans les bases de cas et leur adaptation. La troisième partie est l'apprentissage des nouveaux cas.

Nous allons dans les chapitres suivants détailler chacune de ces trois étapes et mettre l'accent sur les contributions apportées.

5

Elaboration du problème

Sommaire

Cette partie (élaboration du problème en termes de RàPC) correspond à la phase d'extraction et de représentation de la structure physique et logique en termes d'analyse de documents. Cette phase est cruciale pour la suite du traitement, elle conditionne en grande partie les résultats d'interprétation futurs. Nous allons dans la suite extraire la structure du document et la représenter sous forme de cas={problème, solution}.

5.1 Entrées du système

Dans notre application, nous disposons de documents factures et formulaires provenant de différentes origines et en plusieurs langues. Ces documents contiennent des informations très variées : adresses des fournisseurs, adresses des clients, tableaux récapitulant les transactions

effectuées, zones de paiement, détails de paiement, informations relatives aux entreprises (numéro de siret...).

Ces documents sont d'abord scannés à la chaîne puis binarisés par une méthode de binarisation adaptative. Beaucoup de méthodes de binarisation existent dans la littérature [SS04], mais celles qui donnent les meilleurs résultats pour les applications de ce type sont les méthodes de binarisation locales [SSHP97].

Les documents sont ensuite reconnus par un OCR permettant d'extraire les caractères, les mots, les lignes et les blocs :

- les caractères : sont repérés par leurs positions absolues, à savoir les coordonnées du rectangle entourant le caractère.
- les mots : sont caractérisés de la même manière que les caractères.
- les lignes : correspondent à des alignements de mots.
- les blocs : sont des ensembles de lignes alignées à gauche ou à droite.

Ces informations sont stockées dans un fichier XML comme montré dans la figure 5.1.

```
<node type="block" id="1" left="101" top="208" right="429" bottom="250">
 - <node type="line" id="2" left="101" top="208" right="429" bottom="250">
   - <node type="word" id="3" left="101" top="208" right="173" bottom="250" value="ZEP" confidence="100">
       <node type="char" id="4" left="101" top="209" right="124" bottom="250" value="Z" confidence="100"/>
       <node type="char" id="5" left="124" top="208" right="146" bottom="250" value="E" confidence="100"/>
       <node type="char" id="6" left="147" top="209" right="173" bottom="250" value="P" confidence="100"/>
     </node>
   - <node type="word" id="7" left="188" top="208" right="429" bottom="250" value="INDUSTRIES" confidence="100">
       <node type="char" id="8" left="188" top="208" right="200" bottom="250" value="I" confidence="100"/>
       <node type="char" id="9" left="201" top="209" right="230" bottom="250" value="N" confidence="100"/>
       <node type="char" id="10" left="231" top="208" right="257" bottom="250" value="D" confidence="100"/>
       <node type="char" id="11" left="259" top="209" right="285" bottom="250" value="U" confidence="100"/>
```

FIG. 5.1 – Exemple d'une partie d'un document XML lié à une facture

Après avoir pensé utiliser directement les informations obtenues par l'OCR pour les traitements futurs dans notre application (lignes et blocs), nous nous sommes vite rendus compte que ces informations ont un côté aléatoire, voire incompréhensible. Cela reste en partie inexplicable puisqu'on ne connaît pas l'algorithme original présent dans le logiciel d'OCR.

Les seules informations dont nous disposons et qui sont utilisables sont les mots, les caractères et leurs coordonnées absolues dans le document. Ces données sont inutilisables dans l'état, à moins d'effectuer une réorganisation afin de les remettre dans l'ordre dans lequel elles apparaissent dans le document.

Comme nous sommes donc limités aux informations concernant les caractères et les mots, nous partons de ces informations pour essayer de retrouver la vraie structure logique et physique du document (tableaux, structures d'adresses...). Nous allons montrer dans ce qui suit les étapes de traitement effectuées.

5.2 Extraction de la structure physique du document

Les éléments de départ étant les caractères et les mots, nous allons assembler ces éléments petit à petit jusqu'à arriver à la structure physique du document.

5.2.1 Structures à extraire

D'après nos observations des documents à traiter, nous avons constaté la présence de deux types de structures. Tout d'abord, les tableaux qui sont des parties essentielles des factures. Ces tableaux contiennent des informations très importantes comme la description des articles achetés ou vendus. Les structures telles que les adresses, zones de paiement, zones d'information décrivant l'entreprise sont aussi très présentes et nécessitent à elles seules un traitement à part. Elles sont généralement à l'intérieur de blocs verticaux, et sont facilement identifiables grâce à la présence de mots-clés comme : "total, total TTC, rue, ville, code postal, téléphone...". Nous présentons dans les paragraphes suivants les différentes étapes nécessaires à l'extraction finale de la structure physique des documents.

Dans la figure 5.2, un exemple de facture est montré avec ses différents éléments physiques encadrés : logo, mots, ligne, bloc.

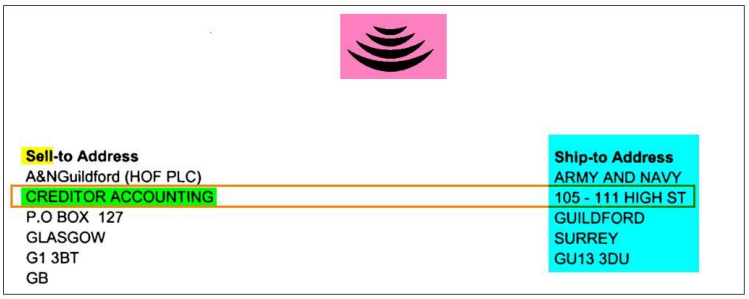


FIG. 5.2 – Un mot en jaune, un ensemble de mots proches sur une même ligne en vert, une ligne horizontale encadrée en orange, un bloc vertical en bleu, un logo en rose

A partir des mots, nous allons constituer des éléments appelés champs, qui serviront par la suite à définir les lignes du document. A partir des lignes et des champs nous allons extraire les deux éléments majeurs de la structure physique du documents : les structures tabulaires (que nous appellerons structures à motifs) et les structures à mots-clés.

5.2.2 Mots

Un mot est caractérisé par sa position absolue dans le document et une étiquette qualifiant sa nature : alphabétique pur (A), alphabétique (B), alphanumérique (C), numérique (N), entier (E).

Un alphabétique pur est un mot qui ne contient que des caractères appartenant à l'alphabet latin. Un alphabétique est un mot qui contient une majorité de lettres alphabétiques plus quelques signes de ponctuation. Un alphanumérique est un mot qui contient aussi bien des caractères numériques que des caractères alphabétiques. Un numérique est un mot qui contient des chiffres et des signes de ponctuation. Il correspond donc à un nombre réel. Un entier correspond à une suite de chiffres éventuellement précédée par le signe "-".

Nous avons fait cette distinction car nous nous sommes aperçus que les erreurs d'OCR peuvent fausser la description d'un mot. Une description très fine des natures des mots peut alors nous aider par la suite. Notons qu'il y a d'ailleurs une hiérarchie dans cet étiquetage. En effet :

$$entier \subseteq numerique \subseteq alphanumerique \supseteq alphabetique \supseteq alphabetiquepur$$

Cela peut nous aider à interpréter certaines parties du document.

En conclusion, chaque mot est caractérisé par sa position absolue dans le document et son étiquette.

5.2.3 Champs

Un champ est un ensemble de mots proches et alignés. Nous classons d'abord les mots du document selon leur ordre d'apparition dans le document (de haut en bas et de gauche à droite). Ainsi, deux mots consécutifs M1 et M2 appartiennent à un même champ si et seulement si :

- $|M1_{top} - M2_{top}| \leq T_{longueur}$ et $|M1_{bottom} - M2_{bottom}| \leq T_{longueur}$. Ceci veut dire que les coins supérieurs gauches et les coins inférieurs droits doivent être à la même ordonnée.
- $|M1_{righht} - M2_{left}| \leq T_{largueur}$. Ceci veut dire que les deux mots ne doivent pas être éloignés l'un de l'autre, l'espace les séparant ne doit pas excéder un seuil $T_{largueur}$.

$T_{longueur}$ est dans notre cas choisi égal à 15 pixels, pour tous les documents que nous avons traités. Ce seuil peut changer en fonction des documents traités. Cela permet une certaine flexibilité de la recherche, tout en évitant la prise en compte de champs indésirables. Quant à $T_{largeur}$, il est pris égal à $\delta \cdot (hauteur\ mots)$ (où δ prend des valeurs entre 1 et 1.5). Il dépend ainsi de la taille des polices. Si, par exemple, une police est très grande (police 24 en police Times pour faire le rapprochement avec le traitement de texte), l'espace entre deux mots est très supérieur à l'espace qu'il y aurait entre deux mots si la taille de la police était petite (12 par exemple). La valeur finalement retenue pour δ été choisie expérimentalement égale à 1.2, après plusieurs tests.

Si les méthodes d'extraction des caractères et d'assemblage en mots, utilisées par l'OCR, étaient disponibles (ce n'est pas le cas, l'entreprise fabricant cet OCR n'a pas divulgué ses secrets de fabrication), il nous aurait été plus facile de définir ces seuils.

Les champs sont étiquetés selon la nature des mots qu'ils contiennent :

- alphabétique pur (codé A) : combinaison de mots alphabétiques purs
- alphabétique (codé B) : combinaison de mots alphabétiques purs et alphabétiques
- alphanumérique (codé C) : combinaison de numériques (entiers ou non) et alphabétiques (purs ou non), ou toute combinaison de numérique, alphabétique et alphanumérique
- entier (codé E) : combinaison d'entiers
- numérique (codé N) : combinaison d'entiers et de numériques

Un champ est finalement caractérisé par sa position absolue et son étiquette. Un exemple de quelques champs est montré dans figure 5.3. Le champ "Code :" est alphabétique. Le champ "218 766 A" est alphanumérique...

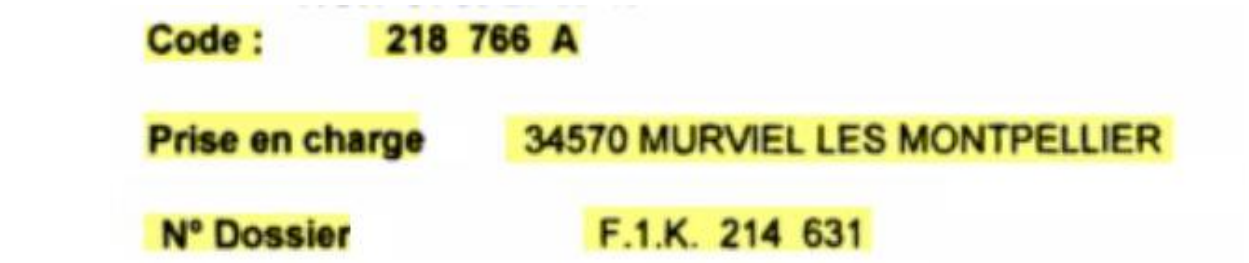


FIG. 5.3 – Exemples de champs extraits

5.2.4 Lignes horizontales

Les lignes horizontales doivent traduire les vraies lignes du document. Ces lignes nous serviront dans plusieurs tâches, comme l'extraction de tableaux ou la recherche de solutions. On définit les lignes comme suit : deux champs C1 et C2 sont sur une même ligne si et seulement si :

- $|C1_{top} - C2_{top}| \leq T_{longueur_{champs}}$
- $|C1_{bottom} - C2_{bottom}| \leq T_{longueur_{champs}}$

Ceci veut dire que les coins supérieurs et inférieurs des champs doivent être à la même ordonnée.

Dans le cas des lignes, $T_{longueur_{champs}}$ est égal à $\delta_1 \cdot (\textit{hauteur champs})$. Nous avons choisi expérimentalement δ_1 égal à 0.5.

A chaque ligne est attribuée une étiquette qui correspond à la concaténation des étiquettes des champs, de gauche à droite. Ainsi, une ligne contenant des champs ayant les étiquettes A, A, B, N, B, E aura l'étiquette AABNBE. Nous appelons cette étiquette dans la suite de cette thèse " motif ". Nous présenterons l'intérêt de ces motifs pour l'extraction des tableaux.

Chaque ligne horizontale est caractérisée par sa position absolue et son motif.

5.2.5 Blocs verticaux

L'extraction des blocs verticaux est similaire à celle des lignes horizontales. Afin de les extraire, les champs sont organisés selon l'ordre d'apparition dans le document. Deux champs C1 et C2 proches sont dans le même bloc vertical si et seulement si :

- $|C1_{bottom} - C2_{top}| \leq T_{hauteur}$ et ($|C1_{left} - C2_{left}| \leq T_{left}$ ou $|C1_{right} - C2_{right}| \leq T_{right}$)

La première condition stipule que l'espace entre les champs ne doit pas dépasser $T_{hauteur}$. Pour notre application, nous avons choisi ce seuil égal à la hauteur moyenne des champs considérés. T_{left} et T_{right} sont pris égaux à 15 pixels pour tous les documents considérés. Nous n'avons pas attribué d'étiquettes aux blocs verticaux car nous n'utiliserons pas cette information.

Chaque bloc est donc caractérisé uniquement par sa position absolue.

5.3 Structures à mots-clés (SMC)

Après avoir extrait les mots, champs, lignes et blocs, nous allons maintenant nous intéresser à l'extraction des structures à mots-clés dans le document.

5.3.1 Mots-clés : définition

Un mot-clé dans un document administratif est un mot redondant et ayant une certaine importance dans le document. Il peut avoir plusieurs formes, mais possède toujours le même sens. Il n'a vraiment d'intérêt que s'il est associé à l'information correspondante (label). Par exemple, le mot "total" est un mot-clé dans les factures. Tous ses synonymes le sont aussi. Il est donc nécessaire d'extraire ce mot-clé et ses synonymes afin d'extraire les informations qui leur sont rattachées. Dans la figure 5.4, on peut voir tous les synonymes et toutes les occurrences étrangères du mot-clé total que nous recherchons dans nos documents.

5.3.2 Extraction des mots-clés

La première solution consiste à observer un ensemble de documents de mêmes classes ou de classes différentes et à essayer d'en dégager les mots qui appartiennent au lexique de l'entreprise.

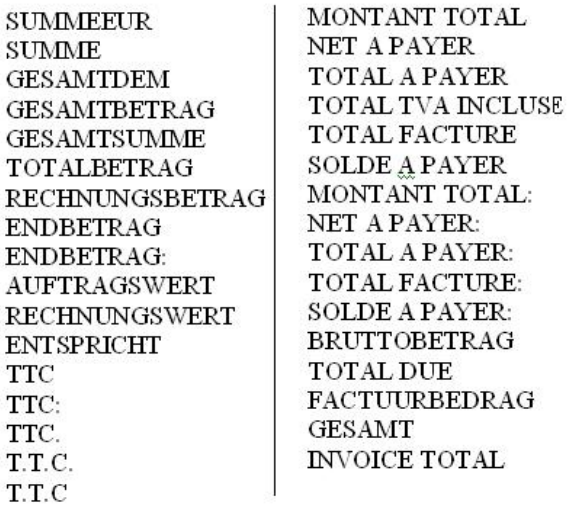


FIG. 5.4 – Les synonymes du mot-clé total, utilisés dans notre application

Cette première approche peut extraire deux types de mots : d'abord les mots qui sont utiles ("total", "rue", ...) et ceux qui se répètent dans tous les documents sans être utiles. Ces mots parasites doivent donc être éliminés par un utilisateur.

Pour notre application, nous disposons de dictionnaires de mots-clés liés au domaine des factures : ces mots clés appartiennent aux synonymes des mots suivants : "montant, devise, date, téléphone, livraison, numéro de livraison, numéro de commande, numéro de TVA, taux, type de voie, code postal, civilité, nom de ville, pays". Ces informations sont utilisées pour extraire les mots-clés dans les documents traités. Ceci est fait en comparant chaque mot de chaque document avec chaque mot des dictionnaires des mots-clés. Chaque mot d'un document obtient alors une étiquette : mot-clé ou non mot-clé. Cette information est importante car cela nous permet d'extraire les structures à mots-clés du document.

L'utilisation des mots-clés dans l'analyse de documents administratifs a souvent été utilisée dans la littérature. Cesarini [CFGS03] a utilisé les mots-clés et leurs synonymes pour interpréter des documents. De même, [SSF+03] a utilisé les mots-clés pour identifier le type de document en cours de traitement.

Pour ce travail de thèse, l'extraction des mots-clés du document est effectuée par le logiciel FullText de ITESOFT, qui prend en compte aussi bien les mots en entrées du système, que les éventuelles erreurs d'OCR qui peuvent être introduites.

5.3.3 Recherche des structures à mots-clés

Le but de cette partie est de présenter la méthode utilisée pour extraire les structures à mots-clés. Nous commençons d'abord par les définir :

- ce sont des blocs verticaux contenant des mots-clés. Cette définition spécifique suppose que tous les blocs sont extraits correctement
- ce sont des ensembles de mots-clés proches (dans un bloc, dans une ligne ou dans une zone quelconque du document). Cette définition est plus large,que la précédente

La deuxième définition inclut naturellement la première. Toutefois, et au vu des documents dont nous avons disposé pendant nos expérimentations, se restreindre à la première définition ne permet pas toujours d'extraire toutes structures à mots-clés. Donc, si les blocs verticaux

fournissent très peu de structures à mots-clés (nombre inférieur ou égal à 1), nous étendons la recherche en utilisant la deuxième définition.

Si les deux définitions ne donnent que très peu de résultats, alors on est dans l'un des deux cas suivants : soit le document contient très peu de mots-clés, soit le document contient trop d'erreurs d'OCR, ce qui rend la détection de mots-clés très difficile. Une discussion sur les erreurs d'OCR dans les paragraphes suivants montrera l'effet de l'OCR sur notre système.

La première définition est très facilement vérifiable dans la plupart des documents administratifs. Il suffit pour cela de prendre le cas des adresses par exemple. Elles contiennent souvent des champs alignés à droite ou à gauche et sont généralement séparées des autres structures par des espaces horizontaux ou verticaux. Il devient dès lors facile de les extraire en tant que blocs verticaux.

La deuxième définition s'applique par contre sur les documents très peu structurés ou sur les documents ne présentant pas d'alignements verticaux. Des mots-clés proches sont soit des mots-clés qui sont sur une même ligne (cela permet d'extraire des SMC comme celles en bas de de la figure 5.5), soit sur des champs isolés.

Au delà de l'aspect physique de ces structures à mots-clés (agencement dans un bloc vertical, proximité physique des mots-clés), nous avons choisi de regrouper les mots-clés en structures car nous avons remarqué qu'il y a aussi une proximité sémantique entre les mots-clés proches. En effet, lorsque nous extrayons un bloc d'adresse, il est impossible d'y trouver le mot-clé "total", car tous les mots-clés d'une adresse sont relatifs au domaine postal (rue, code postal, nom de ville, nom de pays). C'est le cas aussi pour les structures de montant.

Garder cette proximité sémantique nous aidera par la suite dans les phases de recherche, d'adaptation, voire de classification.

Dans la figure 5.5, quelques exemples de structures à mots-clés sont présentés. Les mots-clés sont en rose. Les trois SMC en haut de la figure sont des SMC extraites avec la première définition, alors que les deux autres SMC sont extraites avec la deuxième définition.

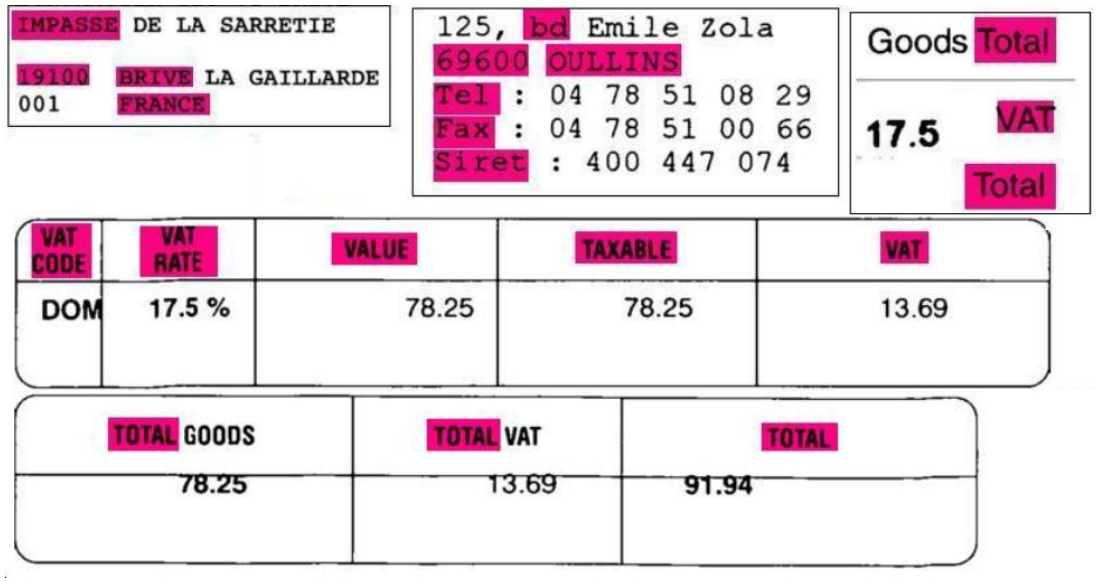


FIG. 5.5 – Quelques exemples de structures à mots-clés

En conclusion, pour extraire les structures à mots-clés, il faut soit chercher à trouver des blocs verticaux contenant des mots-clés, soit des ensembles de mots-clés proches.

5.3.4 Représentation des structures à mots-clés

Les structures à mots-clés sont représentées par des graphes dont les noeuds sont les mots-clés et les arcs les positions relatives de ces mots-clés les uns par rapport aux autres. Cette représentation permet de mémoriser la configuration locale des mots-clés sur le document. En effet, interpréter par exemple le mot-clé "total" accompagné du mot-clé "TVA" n'est pas la même tâche que d'interpréter 'total' tout seul. La représentation en graphe permet à ces structures de garder une certaine flexibilité. Les mots-clés en effet, ne sont pas toujours placés exactement aux mêmes endroits (au niveau pixel) d'une facture à l'autre. Les variations des positions absolues sont totalement absorbées par la représentation en graphes. L'exemple de la figure 5.6 montre une SMC et sa représentation à l'aide d'un graphe. On peut clairement voir que même si les positions absolues des mots-clés changent, il est très peu probable que les positions relatives de ces mots-clés changent (le mot-clé "rue" ne viendra jamais en dessous du "code postal"). Dans cette figure, on distingue deux types d'arcs, ceux qui représentent la relation (gauche, droite) sont en gras. Ceux qui représentent la relation (haut,bas) sont en traits fins. Notons que ces deux types d'arcs sont montrés afin d'expliciter les types de relations entre les noeuds. Un seul type d'arc existe en réalité, seules les étiquettes des arcs changent.

La représentation des structures à mots-clés est homogène avec celle des documents (qui sont aussi représentés en graphes). Elle permet aussi d'avoir une représentation générique de structures similaires, puisqu'elle ne prend jamais les positions absolues en compte. Dans la base de cas de structures, il suffira donc d'avoir un cas d'adresse, par exemple, pour représenter toutes les adresses d'un lot, au lieu d'avoir un cas par document.

FIG. 5.6 – Une SMC et son problème représenté en graphe

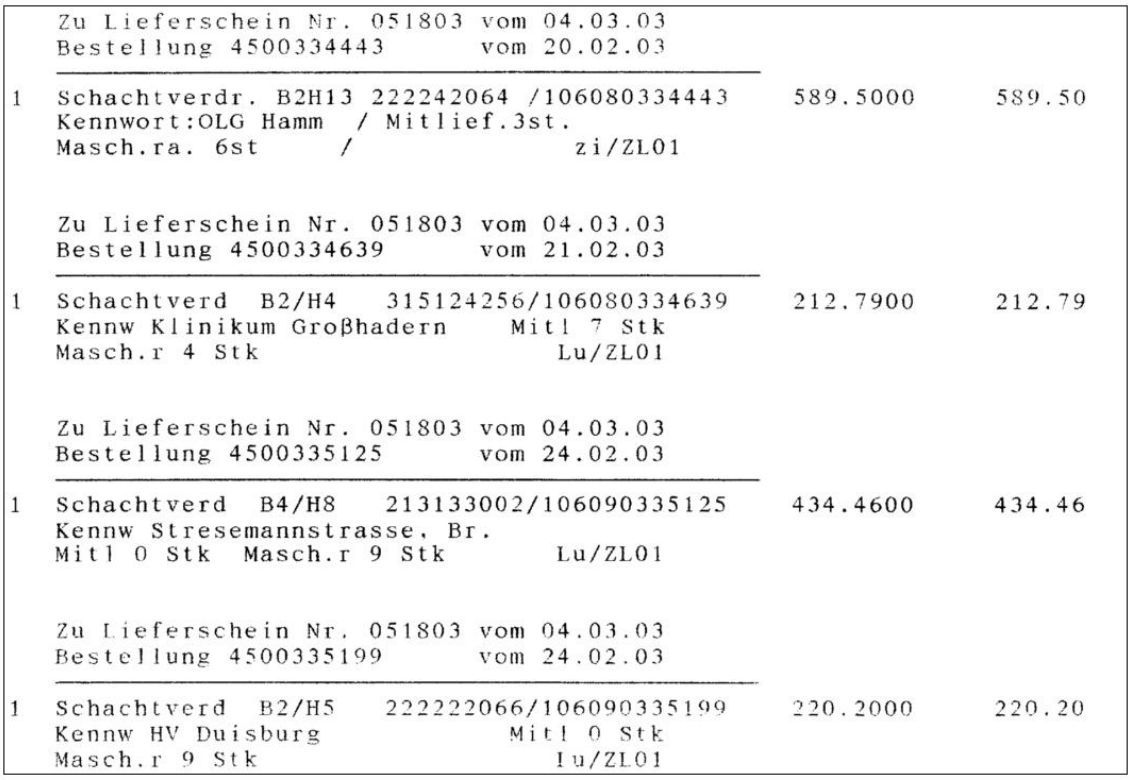
Le problème d'une SMC est donc le graphe de ses mots-clés et sa solution est l'interprétation de chaque mot-clé. Par exemple, dans une SMC contenant le mot-clé "total", la solution correspond à la nature de l'information associée ainsi que sa position. Dans le cas de "total", la solution est alors : "numérique et à droite de "total".

5.4 Structures à motifs (SM)

Nous différencions deux types de structures à motifs :

- Les structures à motif simple : ce sont des tableaux qui ne sont autre qu'une répétition de lignes similaires, avec le même type d'information à chaque ligne. Le contenu même des cellules peut changer, mais la nature des informations de deux cellules à la même abscisse dans deux lignes différentes reste la même.
- les structures à motif composite : ce sont des répétitions de plusieurs lignes plutôt que d'une seule ligne. Dans ce cas, deux lignes consécutives ne se ressemblent généralement pas.

La figure 5.7 montre un exemple de structure à motif composite.



```
   Zu Lieferschein Nr. 051803 vom 04.03.03
   Bestellung 4500334443      vom 20.02.03
   ------------------------------------------------------------------
1  Schachtverdr. B2H13 222242064 /106080334443    589.5000     589.50
   Kennwort:OLG Hamm  / Mitlief.3st.
   Masch.ra. 6st     /              zi/ZL01


   Zu Lieferschein Nr. 051803 vom 04.03.03
   Bestellung 4500334639      vom 21.02.03
   ------------------------------------------------------------------
1  Schachtverd  B2/H4   315124256/106080334639    212.7900     212.79
   Kennw Klinikum Großhadern    Mitl 7 Stk
   Masch.r 4 Stk                   Lu/ZL01


   Zu Lieferschein Nr. 051803 vom 04.03.03
   Bestellung 4500335125      vom 24.02.03
   ------------------------------------------------------------------
1  Schachtverd  B4/H8   213133002/106090335125    434.4600     434.46
   Kennw Stresemannstrasse, Br.
   Mitl 0 Stk  Masch.r 9 Stk       Lu/ZL01


   Zu Lieferschein Nr. 051803 vom 04.03.03
   Bestellung 4500335199      vom 24.02.03
   ------------------------------------------------------------------
1  Schachtverd  B2/H5   222222066/106090335199    220.2000     220.20
   Kennw HV Duisburg            Mitl 0 Stk
   Masch.r 9 Stk                   Lu/ZL01
```

FIG. 5.7 – Exemple d'une structure à motif composite. Le motif représentant un article est situé sur 5 lignes

5.4.1 Structures à motifs simples

Nous avons remarqué, à travers notre observation des documents administratifs, que les zones tabulaires sont en général des zones qui ont une certaine régularité. Nous allons exploiter cette régularité pour extraire les lignes de tableaux dans les documents.

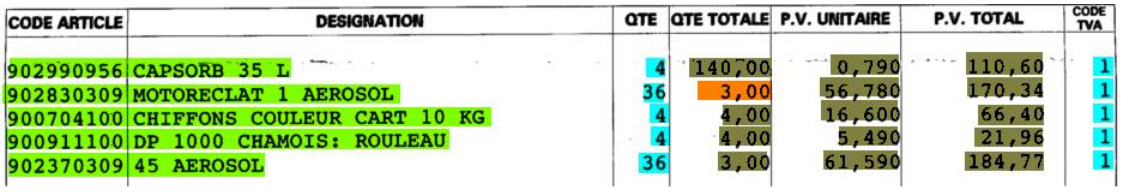
Une structure à motif simple est un tableau dont toutes les lignes sont similaires. Ses caractéristiques sont donc :

- la répétition du même motif de ligne en ligne
- l'alignement vertical des champs composants le motif.

La première caractéristique peut se traduire comme suit : tous les motifs de la zone du tableau sont similaires, ou encore, la distance entre deux motifs quelconques du tableau est nulle (distance sur les chaînes de caractères).

Cependant, et comme nous travaillons sur des données bruitées, le motif n'est pas toujours le même sur toutes les lignes d'un tableau. Les erreurs peuvent provenir de l'OCR : par exemple, un mot initialement alphabétique est étiqueté en alphanumérique, ce qui a pour cause de modifier la nature du champ contenant ce mot. Une autre source d'erreurs est une mauvaise segmentation en mots et champs à cause de la qualité du document. Ainsi, deux champs peuvent être fusionnés ensemble, ou un mot peut être décomposé en deux mots. Ceci fait qu'au lieu d'avoir par exemple un motif avec 4 champs, le motif contient 3 ou 5 champs. La figure suivante montre des exemples typiques d'erreurs de segmentation, et d'erreurs d'OCR.

Les cellules "code article" et "désignation" sont fusionnées (sous segmentation de cellules) lors de la segmentation en champs à cause de leur proximité. Au niveau de l'OCR, le mot '3.00'



CODE ARTICLE	DESIGNATION	QTE	QTE TOTALE	P.V. UNITAIRE	P.V. TOTAL	CODE TVA
902990956	CAPSORB 35 L	4	140,00	0,790	110,60	1
902830309	MOTORECLAT 1 AEROSOL	36	3,00	56,780	170,34	1
900704100	CHIFFONS COULEUR CART 10 KG	4	4,00	16,600	66,40	1
900911100	DP 1000 CHAMOIS: ROULEAU	4	4,00	5,490	21,96	1
902370309	45 AEROSOL	36	3,00	61,590	184,77	1

FIG. 5.8 – Exemples de segmentation de champs de tableaux

est mal reconnu, une mauvaise étiquette lui est attribuée. La méthode de détection de tableau doit prendre en compte ces erreurs.

La similarité entre les motifs est mesurée à l'aide d'une distance entre chaînes de caractères.

La mesure de similarité entre motifs

La mesure de similarité entre chaînes de caractères est un problème largement abordé dans la littérature. Nous nous plaçons dans le cas d'une recherche approximative ("approximate string matching") et avons choisi d'utiliser la distance d'édition entre chaînes.

Soient S et T deux motifs très proches et d la distance utilisée. Si $d(S,T) = n$, cela veut dire qu'il y a n opération d'édition à faire pour passer de S à T. Une substitution est appliquée si deux champs n'ont pas la même étiquette. Ceci peut être dû à un mauvais étiquetage suite à une erreur d'OCR. Une suppression est réalisée si un motif possède un champ de plus ou de moins que l'autre. Ceci peut être dû soit à un champ non détecté, soit à un problème de sur-segmentation ou de sous segmentation.

La méthode de détection

Nous commençons par rechercher pour chaque ligne sa liste de voisins. Il s'agit de lignes ayant un motif similaire. On considère que deux motifs M_1 et M_2 sont similaires si $d(M_1, M_2) \leq \sigma$). σ ne doit pas être très grand au risque de former des groupes de lignes voisines très large. En effet, si σ est égal à 3, et que la ligne étudiée possède le motif : ABBC, l'ensemble des lignes voisines peut contenir A, ACCB, ABBCBBB, ABBCBBC, ABBCAAA.... Nous voyons que l'ensemble des motifs voisins est très grand dans ce cas. D'après nos expérimentations, les meilleurs seuils sont $\sigma = 1$ et $\sigma = 2$. Dans les tableaux à motifs simples (constituant donc des groupes de lignes voisines), il y a souvent une à deux erreurs maximum entre deux motifs. Ces erreurs sont le plus souvent générées soit par des erreurs d'OCR, soit par des erreurs de segmentation.

Une fois les groupes de lignes voisines formés, nous étudions ces groupes afin d'en dégager le (ou les) meilleurs candidats à être des tableaux. Nous examinons donc les alignements des champs ayant les mêmes étiquettes. Prenons par exemple le cas du groupe de lignes ABBC, ABBC, ABBB, ABBC. Pour que ce groupe forme un tableau, il faudrait que les premiers 'A' soient tous alignés verticalement, que les premiers 'B' le soient aussi, et ainsi de suite. Le dernier 'B' du troisième motif doit aussi être aligné avec les derniers 'C' des autres motifs. Si tous les champs sont alignés tel que précisé précédemment, alors ce groupe de lignes est très probablement un tableau.

Nous mesurons un taux d'alignement dans chaque groupe de lignes voisines. Ce taux est :

$$T = \frac{CA}{CT}$$

où CA est le nombre de champs alignés, et CT est le nombre de champs total dans le groupe. Ce taux traduit le nombre de champs alignés dans chaque ensemble. Si ce taux approche 1, alors l'ensemble étudié est soit un tableau, soit un singleton (cas à rejeter). Dans chaque document, nous choisissons le groupe de voisins qui maximise le taux T. Si deux groupes ont un taux T identique, ou très proches, alors le document contient plusieurs tableaux.

Voici l'algorithme proposé pour détecter les tableaux.

lignes du document : ;
Pour chaque ligne **faire**
  Chercher les motifs voisins.
  Former des groupes de motifs voisins.
Fin Pour
Pour groupe de motifs **faire**
  Calculer le nombre de champs alignés verticalement
  Calculer le taux $T = \frac{CA}{CT}$
Fin Pour

Retenir le groupe qui maximise T

Cet algorithme est appliqué sur les lignes du document afin d'en extraire les SM. Grâce à cet algorithme, de très bons résultats en détection de tableaux ont pu être obtenus (voir paragraphes suivants). Il ne reste plus maintenant qu'à affiner cette détection en incluant dans la zone du tableau les lignes qui n'ont éventuellement pas été prises en compte lors de la recherche de lignes voisines, mais qui sont quand même incluses dans le tableau. Il suffit de vérifier si les ligne précédant la zone de tableau et les lignes qui le suivent possèdent le même motif, dans quel cas il faut les rajouter à la liste des lignes déjà extraites.

Tests de l'extraction des SM simple

Afin de tester notre approche, nous avons créé des documents de vérité. Dans chaque document, nous avons extrait les zones contenant les SM et nous les avons codées dans un document XML en précisant les vraies étiquettes des champs (celles que donnerait un expert). Notre base de test est constituée de 95 documents provenant de 9 classes totalement différentes et contenant 698 lignes de tableaux et 3955 champs de tableaux. Nous avons constitué cette base de test manuellement, et construit un document de vérité pour chaque document contenant un tableau. Le résultat de l'extraction est ainsi comparé avec le résultat du document de vérité. Nous avons utilisé deux mesures pour évaluer la performance de l'extraction : la première consiste à compter le taux de champs de tableaux détectés (parmi tous les champs de tableaux à extraire), et la deuxième consiste à trouver le taux de lignes de tableaux détectées sur l'ensemble des lignes de tableaux dans la base de test. La première mesure sert en fait à voir si la zone du document contenant le tableau a bien été extraite à partir du document. La deuxième permet quant à elle de voir si les lignes du tableau ont aussi bien été détectées.

La première comparaison utilise les étiquettes et les positions des champs comme moyen de comparaison entre les champs extraits et ceux du document de vérité. La deuxième mesure utilise

les motifs des lignes extraites et leurs positions absolues dans les document. Les résultats sont les suivants :

- champs détectés et étiquetés correctement : 89.63%
- lignes détectées : 94%

Ces premières expériences montrent l'efficacité de la méthode proposée dans la détection de tableaux. Le taux de lignes détectées est naturellement supérieur à celui des champs détectés parce qu'on peut détecter une ligne, mais sa répartition en champs ne correspond pas à la décomposition en champs du document de vérité. Les erreurs proviennent de :

- l'OCR qui reconnaît mal certains mots. Les étiquettes qui leur sont attribuées sont alors incorrectes. Par exemple, si un champ alphabétique est étiqueté en alphanumérique, à cause d'une erreur d'OCR sur l'un des mots du champ, il est possible que la ligne contenant ce champ ne soit pas détectée puisque son motif en sera affecté.
- notre système qui sur-segmente ou sous-segmente certains champs. Cela est dû à l'utilisation des seuils (ceux de l'extraction des champs ou des lignes).

5.4.2 Structures à motifs composites

La détection de motifs composites est basée sur celle de motifs simples. La figure 5.9 montre un exemple de motif composite. Ce premier exemple est cependant moins complexe que l'exemple de la figure 5.10.

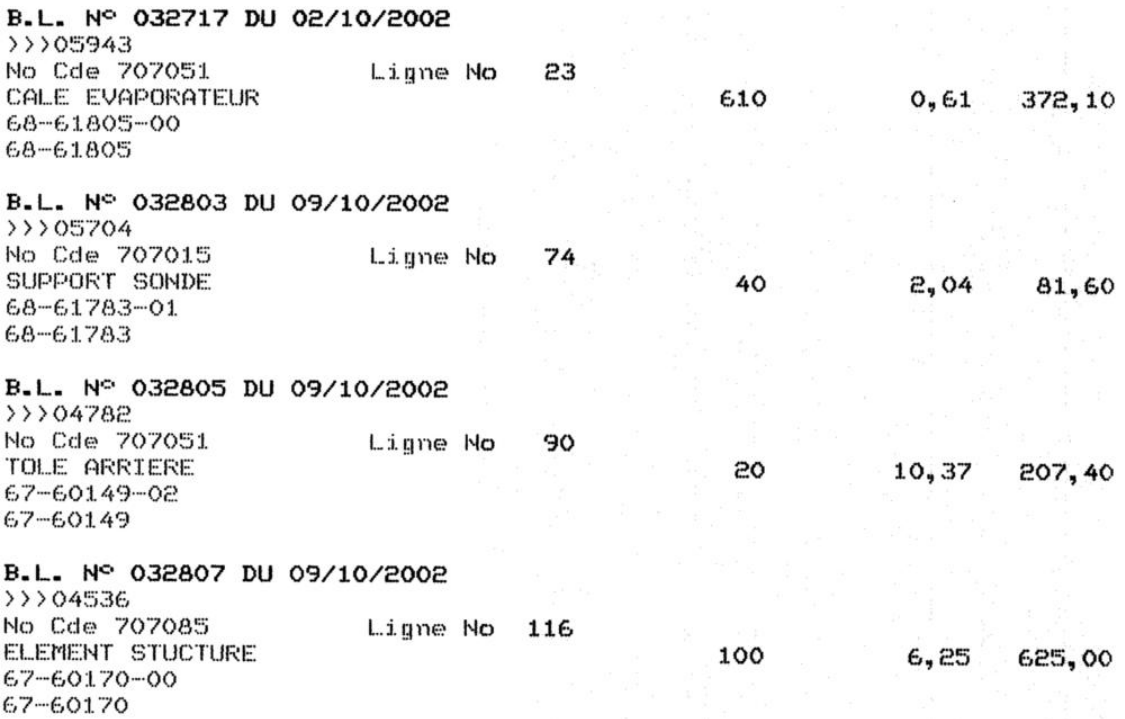


FIG. 5.9 – Exemple d'une zone à motif composite.

La méthode de détection

Un motif composite est un ensemble de motifs simples, différents et répétés. Une structure à motif composite contient un ou plusieurs motifs composites. Elle possède généralement un motif

Code Produit	Désignation du Produit	Prix de Base	Remises Unitaires	Prix Net Unitaires	Quantité	Prix Net Total
Nr Commande Client: **00427519**			Nr Commande: **285806**			
00004749	1L CLAIR NETT.MENAGER ANTIBACT. MUG	6,49/ 10 UVC		0,6600	600 UVC	396,00
	3564700023945		Ecotax	0,0110		
00004743	1L CLAIR NETT.MENAGER ANTIBACT.FLOR	6,49/ 10 UVC		0,6600	600 UVC	396,00
	3564700023952		Ecotax	0,0110		
00004748	1L CLAIR NETT.MENAGER ANTIBACT.CITR	6,49/ 10 UVC		0,6600	1.200 UVC	792,00
	3564700026779		Ecotax	0,0110		
00004747	1L CLAIR NETT.MENAGER ANTIBACT.POMM	6,49/ 10 UVC		0,6600	1.200 UVC	792,00
	3564700027806		Ecotax	0,0110		
% TVA	*Total Hors TVA*		*Total TVA*			*Total TTC*
19.60	2.376,00 EUR		465,70 EUR			2.841,70 EUR

FIG. 5.10 – Exemple d'une zone à motif composite plus complexe que celui de la figure 5.9.

simple dominant (MD) et d'autres motifs simples de moindre importance. L'extraction de la SM composite sera donc basée sur l'extraction des SM simples (dominants) qui la composent. Nous commençons par chercher le MD puis nous essayons de constituer un motif composite à l'aide des motifs voisins du MD :

- Le MD est un motif simple, il est donc extrait à l'aide de l'algorithme détaillé précédemment.
- Ensuite, on délimite une zone de recherche des motifs complexes. Elle se situe entre la première et la dernière occurrence du MD dans le document. Elle est complétée par quelques lignes horizontales situées avant la première occurrence et après la dernière.
- On filtre les lignes horizontales indésirables dans la zone de recherche en étudiant leur nombre de répétitions.
- On calcule la taille de la SM composite (approximative) en se basant sur le nombre de lignes compris entre 2 occurrences consécutives du MD .
- On délimite le début et la fin de la SM grâce à sa taille et à la disposition physique de ses occurrences dans le document.
- Après avoir identifié les différents motifs composites candidats, on choisit celui qui minimise la distance d'édition par rapport aux autres motifs.

Avec l'algorithme détaillé précédemment, on peut donc extraire les structures à motifs composites.

Tests de l'extraction des SM composites

Les tests ont été effectués sur 33 documents contenant 279 motifs complexes et 2402 champs de tableaux. Les motifs complexes sont très variés et contiennent entre 2 et 6 motifs simples. La même mesure que précédemment a été utilisée sauf que "ligne" dans le cas des SM à motifs composite indique un ensemble de lignes constituant un motif. Les résultats sont les suivants :

- champs détectés et étiquetés correctement : 95%
- lignes détectées : 85%

Dans ce cas, le taux de détection des lignes est inférieur à celui des champs. En effet, grâce aux motifs dominants, il est possible de délimiter la zone contenant tous les éléments de la SM composite. Il reste alors à trouver les lignes d'articles à l'intérieur de ces zones. Si le motif de la ligne d'article est mal choisi, cela se répercute sur le résultat final.

Malgré la difficulté de la tâche, notre approche arrive quand même à obtenir de très bons résultats de détection. Elle permet en outre de rester indépendant d'un utilisateur (qui modélise ces structures à motifs composites). D'autres améliorations peuvent être apportées dans le cas des SM composite. Nous pouvons notamment penser à l'intégration des données images comme

les projections horizontales et verticales.

Les erreurs de détection sont de la même origine que celles de la détection de SM simple. Notons cependant que si le motif dominant (qui est un motif simple) n'est pas détecté, il est alors impossible de détecter la SM composite.

5.4.3 Représentation des structures à motifs

Comme leur nom l'indique, les structures à motifs sont représentées par leurs motifs. Si le motif d'une SM est "ABBBN", alors le problème de cette SM n'est autre que le motif lui même. La solution de ce problème est l'interprétation des différents champs composant ce motif (par exemple, B="prix unitaire", N="prix total").

5.5 Problème du document

Maintenant que nous disposons de toutes les structures du document, nous pouvons extraire le modèle du document que nous avons choisi de représenter par un graphe. Cette représentation se justifie de plusieurs manières.

La représentation en graphe, avec les mots-clés ou les structures sur les noeuds, et les positions relatives des noeuds sur les arcs, permet au problème du document d'être très souple et de s'affranchir des positions absolues. Ainsi, deux documents d'un même lot auront le même graphe représentatif même si les structures n'ont pas les mêmes positions absolues. La figure 5.11 montre deux documents d'un même lot avec un même graphe représentatif. Sur des documents de classes différentes, la représentation en graphe est aussi très intéressante. Supposons que deux documents très différents contiennent exactement les mêmes mots-clés. En considérant une représentation en vecteur, il serait difficile de faire la différence entre les deux documents. Cependant, la représentation en graphe le permet puisqu'elle prend en compte les positions relatives des noeuds du graphe. Un graphe permet aussi de conserver les informations structurelles du document. Il est plus adapté à l'ajout de nouveaux types d'information qu'une représentation en vecteur par exemple. En effet, dans les représentations des documents (en TF-IDF par exemple), dès qu'il y a un nouveau mot-clé qui arrive, il faut tout de suite ajouter un élément au vecteur, on passe alors d'un traitement dans un espace de dimension n, à un espace de dimension $(n+1)$. Pour le cas des graphes, il suffit de rajouter un nouveau noeud au graphe. Les autres graphes ne contenant pas cette information n'auront pas à supporter cet ajout (alors que tous les vecteurs devront le faire).

Finalement, le système que nous proposons est un système à deux cycles de RàPC. Le premier fonctionne si un cas de document similaire existe dans la base de cas. Le deuxième fonctionne si le premier ne trouve pas de solution. Ce deuxième cycle utilise les structures du document. Une étude récente menée par Cunningham [CWP+04]a montré que l'utilisation des représentations en graphes dans le RàPC textuel, plutôt que des représentations en vecteurs permet une meilleure recherche (des cas proches) dans les systèmes de RàPC

La modélisation d'un document avec un graphe a déjà été proposée par nombre d'auteurs :

- Walischewski [Wal97] propose de modéliser des documents administratifs (lettres manuscrites et enveloppes) par des graphes, où les noeuds représentent les structures du document (adresse, corps de la lettre, signature,...) et les arcs leurs positions relatives
- Cesarini [CMS02] propose d'utiliser un arbre pour modéliser des documents de type facture. Les noeuds de cet arbre correspondent à des zones segmentées à partir de l'image par des coupes verticales et horizontales

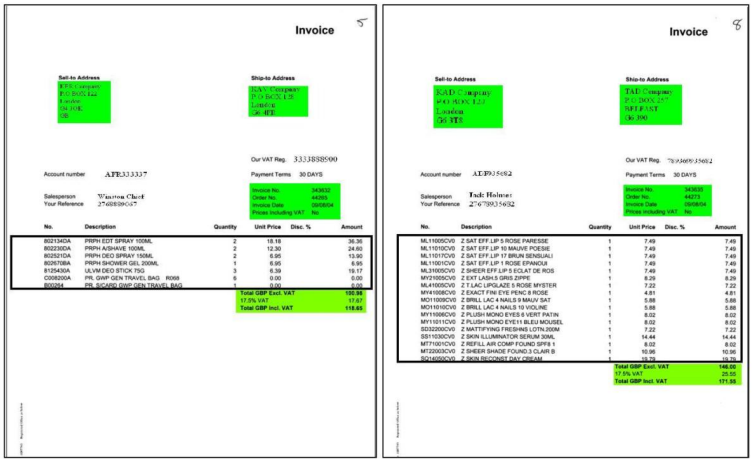


FIG. 5.11 – Deux factures d'une même classe. Ces deux factures sont représentées par le même graphe

– Liang [LD02] propose aussi d'utiliser un graphe pour modéliser un document. Appliqué à des articles scientifiques, les noeuds du graphe sont les structures logiques de l'article (titre, auteur, section) et les arcs sont les positions relatives de ces structures les unes par rapport aux autres

La nouveauté de notre approche réside non pas dans la modélisation des documents par des graphes, mais dans les traitements ultérieurs appliqués à ces graphes.

Schenker [SLBK04] a comparé la modélisation des documents par des graphes à celles par des vecteurs. Il a particulièrement étudié cela dans le domaine de la classification des documents. Il en a conclu que la représentation en graphes permet de meilleurs résultats de classification que la représentation des vecteurs pour plusieurs distances inter-graphes. Ceci conforte encore plus notre choix de représenter nos documents sous forme de graphes.

5.6 Renforcement du problème du document

Dans notre système CBRDA, il existe des documents tellement dégradés, ou tellement pauvres en information, que leurs modèles (problèmes) en sont affectés. Or, comme un traitement de documents par lot est très fréquent dans notre système, il serait dommage de ne pas prendre en compte la présence de plusieurs documents de la même classe pour modéliser ces documents de la meilleure manière possible. Nous supposons qu'un lot est toujours constitué de documents similaires, de même modèle, par exemple, toutes les factures d'un même fournisseur. Nous allons donc essayer d'utiliser les redondances des informations présentes dans les documents d'une même classe afin d'extraire le meilleur problème représentatif de ces documents.

Pour cela, nous savons que chaque document est modélisé par un graphe. Ce graphe représente les structures présentes dans le document. Supposons que deux problèmes de document soient différents par une structure S. Le modèle renforcé du document prendra en compte cette différence. Pour arriver à un problème de document renforcé, nous procédons comme suit :

– Pour chaque document du lot, nous extrayons le graphe représentatif du document. Le premier graphe est celui qui sera renforcé (on l'appelle GR)
– On compare le problème de chaque document avec GR, et on met à jour GR en fonction des informations présentes dans le graphe du document. On peut tenir compte des fréquences

de chaque structure ou élément de structure afin d'avoir la meilleure couverture possible des documents du lot.

5.6.1 Graphe pondéré représentatif d'un lot

Un concept permettant cet enrichissement existe et couvre bien notre idée. Il s'agit du concept du "Weighted Minimum commun Supergraph" (WMCS) ("le plus grand super graphe commun pondéré"). Ce concept a été proposé par Bunke [HFGV03] dans le but de classer des graphes. Il peut en effet être très judicieux de représenter une classe de graphe par le graphe WMCS puisqu'il comprend toutes les sous parties des graphes, accompagnées de leurs fréquences.

Plus formellement, ce problème revient à un problème de "graph clustering". A partir d'un ensemble de graphes $g_i, i = 1..n$, on essaie de construire un graphe qui représente au mieux ces graphes. Le graphe G doit traduire les propriétés des graphes de la classe, et contenir toutes les informations (noeuds ou arcs) présents dans g_i. Un MCS est donc un graphe tel que chaque graphe g_i réalise un isomorphisme de sous graphes avec un sous graphe SG de MCS. Ainsi, chaque graphe g_i est présent dans MCS. Si on veut prendre en compte les poids de chaque partie du graphe (dans MCS) en fonction des autres graphes, on définit alors le WMCS. Celui ci est défini formellement comme suit :

V est un ensemble fini de noeuds.

$\lambda : V \rightarrow N$ est une fonction donnant des poids positifs aux noeuds(fréquences des noeuds).

$E \subseteq V \times V$ est l'ensemble des arcs.

$\epsilon : E \rightarrow N$ est une fonction associant des poids positifs aux arcs(fréquences des arcs).

$\alpha : V \rightarrow L$ est une fonction associant des étiquettes(attributs) aux noeuds .

$\beta : E \rightarrow L$ est une fonction associant des étiquettes(attributs) aux arcs.

Le calcul du WMCS de plusieurs graphes est un problème NP complet. Dans [HFGV03], Bunke introduit une méthode d'approximation du calcul du WMCS, qui se base sur le calcul du WMCS à partir de deux graphes à chaque fois. Le but de cette approximation est de parcourir les graphes de la classe, et de calculer le WMCS du graphe étudié, et du WMCS actuel. La notion de WMCS proposée par Bunke correspond exactement à ce que nous cherchons à faire dans le cas des documents. Voici le détail de cette méthode, qui réduit considérablement la complexité de calcul (linéaire en le nombre de graphes) :

$W = g_1$: **Le premier graphe initialise le graphe pondéré. Les fréquences des arcs et des noeuds sont initialisées à 1** ;
Pour chaque graphe **faire**
  $W = WMCS(W, g_i)$
  $WMCS \leftarrow W$
Fin Pour

Algorithme 1: Calcul du WMCS

Le calcul du WMCS de deux graphes est assez simple. Il suffit en fait de comparer les arcs et les noeuds des deux graphes et d'incrémenter les fréquences des parties communes. En calculant ainsi W (qui est un WMCS intermédiaire de tous les graphes), on alors la fréquence de tous les éléments

En incluant les fréquences des arcs et des noeuds dans le graphe WMCS, on peut alors dégager plusieurs propriétés intéressantes du WMCS. On peut trouver tous les objets (sur les noeuds) qui ont une fréquence supérieure à une fréquence donnée. Pour le cas des documents par exemple, on peut ainsi retrouver le nombre de fois que le mot clé total a été utilisé dans le lot. On peut aussi savoir si un objet (ou une relation entre 2 objets) fait partie des informations importantes ou du bruit. Par exemple, si $freq(objet) << seuil$, on peut en conclure que cette information fait a priori partie du bruit.

La procédure de construction du WMCS pour le cas d'un lot de document est celle proposée par Bunke. On part d'un ensemble de documents. On extrait le graphe à partir de chaque document. On construit le nouveau WMCS à partir de l'ancien WMCS et du graphe en cours. Le graphe obtenu WMCS représente alors tous les documents traités. C'est alors le graphe du lot.

Les avantages de l'utilisation de WMCS par rapport à l'utilisation du graphe d'un seul document sont :

- le WMCS est beaucoup plus fiable que le graphe d'un seul document : les fréquences des noeuds et des arcs dénotent de cette fiabilité quand ils sont supérieurs au seuil minimal de présence
- lors de l'enrichissement de la base de cas, il vaut mieux avoir un problème qui soit le plus fiable possible. La solution à un problème formé par un WMCS sera la plus complète possible.

5.6.2 Extraction de cas par redondance

Au cours du traitement de document par lot, et indépendamment du renforcement du graphe par construction d'un graphe WMCS de tous les graphes du lot, un autre traitement pouvant faciliter les tâches ultérieures du traitement peut être envisagé. Ce traitement consiste à étudier les redondances présentes dans les documents du lot et d'essayer d'en dégager certaines règles. Nous n'allons pas nous intéresser uniquement aux redondances des mots, mais aussi aux redondances des champs, lignes, voire structures à mots-clés. Même si cette partie n'a pas été mise en oeuvre au cours de l'évaluation réelle de notre système CBRDA, elle présente néanmoins une perspective réelle.

Nous allons observer tous les documents d'un lot de documents similaires. Tous les documents de ce lot ont le même modèle. Même si le processus proposé est parfois long, il peut néanmoins apporter certains enrichissements aux bases de cas. Nous proposons de comparer tous les mots, champs, lignes, SMC de tous les documents. Ceci permettra de dégager les éléments (élément incluant mots, champs lignes et SMC) les plus redondants, et par la suite, les relations contenues dans ces éléments redondants.

Afin de dégager le plus de relations possibles, nous remplaçons tous les chiffres de chaque document par une lettre (X par exemple). Ceci permet de détecter par exemple une redondance du type : "téléphone + XX XX XX XX XX", ou encore "Montant + XX,XX". Une fois les comparaisons terminées, nous choisissons un seuil de redondance minimal, afin d'éliminer tous les bruits possibles (redondances très faibles, correspondant à une information faiblement présente sur l'ensemble des documents.

Une fois les comparaisons inter-documents effectuées, nous obtenons les redondances suivantes selon 4 formes possibles :

- un ensemble de mots redondants. Ces mots correspondent soit à des mots-clés, soit à des formes de chiffres redondants (XX.XX par exemple trouvé sur tous les documents). Ces mots redondants peuvent aussi correspondre à des informations inutiles (les notes de bas

de page, la description de l'entreprise.

- un ensemble de champs redondants. Un champ étant un groupe de mots proches, ces redondances sont plus intéressantes que les redondances de mots. Elles permettent en effet de détecter la redondance d'associations de mots qui peuvent être très intéressantes à exploiter. Supposons par exemple que le champ suivant "total + XX,XX" soit très redondant dans le lot. Ceci veut dire que l'interprétation du mot-clé " total " est située dans le même champ que le mot lui même. La nature et l'emplacement de l'interprétation sont fournies. Cette information peut alors être injectée dans la base de cas de structures. Elle constitue en fait une structure à mot-clé avec un seul mot, mais dont la solution est sûre. Cette information peut même être extrapolée pour donner une règle concernant les mots-clés synonymes de "total". La règle qu'on peut alors mettre dans la base est la suivante : "MC synonymes du mot-clé du document + réel à droite".
- un ensemble de motifs redondants. Dans le cas des documents contenant des structures à motifs, il est très probable que le motif du tableau se répète sur tous les documents. Obtenir le motif du tableau après une phase d'observations sur les documents du lot peut conforter notre méthode de détection de tableaux.
- un ensemble de SMC redondantes. Ici, nous faisons une extraction de redondance en deux dimensions. En effet, non seulement nous retrouvons les redondances au niveau des graphes des SMC, mais aussi au niveau des champs qu'ils contiennent. L'information dans ce cas est plus complète que lors de l'extraction de redondances de champs car on combine aussi bien l'information sur les problèmes (graphes) que sur les mots composants les noeuds de ces graphes. Par exemple, supposons qu'une SMC contienne 3 mots-clés "total", "TVA" et "Taxes". Les comparaisons effectuées ont permis de trouver que ces 3 mots sont bien redondants ensemble. De plus, en prenant en compte les champs contenant ces mots, nous pouvons nous apercevoir que ces champs sont aussi redondants ("total + XX,XX", "tva + XX,XX%" et "Taxes + XX,XXX"). Nous pouvons ainsi non seulement extraire des problèmes redondants, mais nous pouvons aussi extraire leurs solutions en même temps.

Les informations redondantes extraites peuvent dès lors alimenter la base de cas. Ces informations sont très fiables vu qu'elles ont été formées grâce à des observations de redondance dans un lot de documents. Leur utilisation pour la résolution de problèmes similaires peut être très avantageuse pour notre système.

5.7 Conclusion

Dans ce chapitre, nous avons présenté la phase d'élaboration du problème à partir du document. En partant des données brutes extraites de chaque document, à savoir les mots et leurs positions, nous avons formé petit à petit des assemblages d'informations (champs, lignes horizontales, blocs verticaux) jusqu'à arriver à extraire les structures à motifs et les structures à mots-clés. Notre méthode d'extraction de structures à motifs s'est avérée être très efficace, permettant d'extraire aussi bien des SM simples que des SM composites.

Une fois que les structures du document on été extraites, nous avons proposé de former un problème du document, qui n'est autre que le graphe de ces structures, traduisant la présence et les positions relatives de ces structures les unes par rapport aux autres. Ce problème est alors flexible (puisqu'il ne prend pas en compte les positions absolues des structures dans le document) et représentatif non seulement du document, mais aussi de beaucoup d'autres documents du même lot. Le fait d'utiliser un lot de documents similaires permet aussi d'extraire plusieurs types d'informations qui, une fois analysées, permettent une meilleure utilisation ultérieure de

la base de cas ainsi qu'une meilleure interprétation des documents de ce lot.

Nous pouvons envisager plusieurs pistes d'amélioration possibles concernant l'élaboration du problème :

- d'abord, l'extraction de SM peut être améliorée par l'utilisation des entêtes de tableaux. Ceci permettrait de localiser plus rapidement l'endroit exact du tableau, ou du moins, la première ligne.
- l'extraction de SMC se base sur le logiciel Full-text d'ITESOFT qui cherche dans chaque image tous les mots-clés du document, et ce, quelque soit le document à traiter. Pour une utilisation plus performante de ce logiciel, on pourrait, si le document est de classe connue, limiter l'utilisation du logiciel à certaines zones du document, ce qui permettrait un gain de temps très considérable.

Le chapitre suivant montre l'utilisation du problème pour l'interprétation du document.

6

Recherche et adaptation

Sommaire

Nous présentons dans ce chapitre les deux premières grandes étapes du RàPC : la recherche de cas proches et l'adaptation. Nous détaillons comment, et avec quelles mesures, la recherche de similarité a été effectuée, et nous définissons aussi ce qu'est un cas proche. Ensuite, nous présentons la partie adaptation. Ayant trouvé un cas proche, nous adaptons sa solution pour que le nouveau cas étudié trouve aussi une solution et soit ainsi résolu.

6.1 Recherche de cas proches

Dans notre système, nous disposons de trois types de cas (SM, SMC et document). Il faut donc définir une mesure de similarité pour chacun de ces cas. Concernant les motifs, nous avons déjà exposé dans le chapitre précédent la distance d'édition. C'est cette distance que nous utilisons pour rechercher des motifs proches dans la base de cas. Pour les SMC et les documents, nous avons opté pour des méthodes de comparaison de graphes.

6.1.1 Comparaison de graphes

Comparer deux graphes consiste à comparer leurs structures (noeuds et arcs), c'est à dire la répartition des arcs sur les sommets. Nous présentons dans cette partie plusieurs méthodes de comparaison de graphes. Ces méthodes sont généralement conçues pour des graphes non étiquetés, mais ont toutes été étendues aux graphes étiquetés (qui est un cas plus facile que

les graphes non étiquetés). Nous commencons d'abord par donner quelques définitions avant de présenter certaines distances utilisées dans la littérature.

Graphes étiquetés

Un graphe étiqueté est un graphe défini par un tuple $G = (V, E, L_V, \alpha, L_E, \beta))$ où :
- (V, E) est un graphe, c'est un ensemble d'arcs et de noeuds,
- L_V est un ensemble d'étiquettes de sommets,
- $\alpha : V \rightarrow L_V$ est une application d'étiquetage des sommets du graphe,
- L_E est un ensemble d'étiquettes d'arc,
- $\beta : E \rightarrow L_E$ est une application d'étiquetage des arcs du graphe.

Isomorphisme de graphes

Deux graphes sont isomorphes s'ils sont identiques, à un renommage de leurs noeuds près.

Définition (Isomorphisme de graphes)

Deux graphes $G = (V, E, L_V, \alpha, L_E, \beta)$ et $G' = (V', E', L_{V'}, \alpha', L_{E'}, \beta')$ sont isomorphes si et seulement si $|V| = |V'|$ et il existe un appariement bijectif $m \subseteq V \times V'$ tel que $\forall (u, v) \in V^2, (u, v) \in E \Leftrightarrow (m(u), m(v)) \in E'$ et $\forall u \in V, \alpha(u) = \alpha'(m(u))$ et $\forall (u, v) \in E, \beta((u, v)) = \beta'((m(u), m(v)))$.

Autrement dit : deux graphes G et G' sont isomorphes si et seulement si à chaque noeud du graphe G, on peut trouver un et un seul noeud correspondant dans G' et inversement, idem pour les arcs.

Définition (Isomorphisme de sous-graphes)

Un isomorphisme de sous-graphes est une fonction injective $f : V \rightarrow V'$ telle qu'il existe un sous-graphe $S \subseteq G'$ tel que f soit un isomorphisme de graphes entre G et S.

Nous présentons maintenant quelques méthodes de mesure de similarité entre graphes. Nous distinguons deux approches différentes dans la comparaison de graphes : les méthodes de comparaison exactes (celles qui cherchent à trouver un isomorphisme entre deux graphes donnés), et les comparaisons inexactes ("inexact graph matching"), qui consistent à chercher si un graphe est un sous graphe d'un autre [Ull76] (isomorphisme de sous graphes).

Parmi les méthodes de comparaison de graphes, certaines utilisent des parties communes aux graphes comparés : le plus petit super-graphe commun ("Minimun Ccommon Supergraph" MCS), et le plus grand sous-graphe commun ("maximum common subgraph" mcs). Ces notions sont utilisées pour calculer la distance entre deux graphes.

Soient G et G' deux graphes étiquetés. MCS et mcs sont définis comme suit :
- mcs est un graphe qui est inclus dans G et G'. C'est le plus grand sous-graphe commun entre G et G'. Il réalise un isomorphisme de sous-graphes entre lui même et les sous-graphes correspondants dans G et G'.
- MCS est un graphe tel que $G \subseteq MCS$ et $G' \subseteq MCS$. Ceci veut dire que G et G' réalisent des isomorphismes de sous -graphes entre eux et MCS. MCS est le plus petit super graphe contenant G et G'.

Voici des exemples de distances utilisant mcs et/ou MCS :

– La première distance utilise le mcs de G et G' :

$$dMCS(G,G') = 1 - \frac{|mcs(G,G')|}{max(|G|,|G'|)}$$

où |...| correspond à la taille du graphe (nombre d'arcs et de noeuds).

– La deuxième distance est aussi inférieure à 1 ; elle prend en compte la taille du mcs par rapport à l'union des deux graphes :

$$dWGU(G,G') = 1 - \frac{|mcs(G,G')|}{|G| + |G'| - |mcs(G,G')|)}$$

Notons ici que le dénominateur représente l'union de G et G'. C'est la somme des tailles des graphes à laquelle on enlève l'intersection (la partie commune) des deux graphes.

– La troisième distance est différente des précédentes dans la mesure où elle utilise le MCS en plus du mcs :

$$dMMCSN(G,G') = 1 - \frac{|mcs(G,G')|}{|MCS(G,G')|}$$

– Notons ici que les trois distances présentées sont toutes normalisées. Il existe d'autres distances non normalisées :

$$dUGU(G,G') = |G| + |G'| - 2 \cdot |mcs(G,G')|)$$

$$dMMCS(G,G') = |mcs(G,G')| - |MCS(G,G')|$$

– Bunke [Bun97] a démontré un lien direct entre la distance d'édition et le mcs. En effet, si on choisit la fonction de coût suivante : (seules les insertions ou les suppressions de noeuds ont un coût de 1, les autres opérations sur les noeuds ou les arcs ont un coût nul), le coût des opérations d'édition pour passer de G à G' est alors :

$$Cout(edit) = |G| + |G'| - 2 \cdot |V|$$

où V est la taille d'un sous-graphe de G ayant un isomorphisme avec un sous-graphe de G' (donc c'est un sous graphe commun). Comme la distance d'édition doit correspondre au coût minimal des opérations d'édition, ceci revient donc dans la formule précédente à maximiser V. Il faut donc que V soit égal au mcs. D'où le lien entre le mcs et la distance d'édition entre graphes.

6.1.2 Distance d'édition entre graphes

Une autre distance, qui peut aussi être liée à MCS et mcs a été introduite par Bunke [Bun97]. Très inspirée des travaux portant sur la distance d'édition entre les chaînes de caractères, cette mesure de similarité prend en compte les opérations d'édition (suppression, substitution, insertion de noeuds et d'arcs) afin de passer d'un graphe G à un graphe G'. La distance d'édition entre G et G' est alors prise comme l'ensemble le moins coûteux d'opérations nécessaires pour passer de G à G'.

Terminologie

Prenons les deux graphes G et G'. Soit m la fonction d'appariement entre G et G'. Pour éditer G en G', on peut réaliser les opérations suivantes sur les noeuds :

- substituer un noeud u de G par un noeud v de G' : $m(u) = v$. Si $\alpha(u) = \alpha'(m(u))$, alors c'est une substitution identique (opération qui ne comptera finalement pas), sinon, c'est une substitution non identique.
- supprimer un noeud, ceci arrive si pour un noeud u de G, $m(u) = ensemble\ vide$
- insérer un noeud, ceci arrive si pour un noeud v de G', $m^{-1}(v) = ensemble\ vide$

Ces 3 opérations peuvent être appliquées sur les arcs aussi.

Selon l'application, des coûts différents peuvent être attribués aux différentes opérations d'édition. Ces coûts doivent se baser sur des heuristiques relatives à l'application traitée [MB98].

La distance entre G et G' est alors

$$d(G, G') = \sum(cout(editions))$$

où $cout(editions)$ est le coût des opérations d'édition.

Le calcul de la distance d'édition peut être accélérée par utilisation d'une approche proposée par Messmer [MH98] dans le cadre d'une comparaison d'un graphe avec une base de graphes. Il propose de décomposer les graphes de la base en un ensemble de sous graphes, chaque sous graphe étant représenté une seule fois dans la base. Cette décomposition de la base entraîne un gain de temps considérable et permet d'envisager l'utilisation de la distance d'édition dans des tâches de comparaison d'un graphe avec plusieurs autres.

Application aux graphes de structures

Nous avons choisi la distance d'édition pour comparer les graphes des structures à mots-clés. Nous avons décidé d'utiliser cette distance avec des coûts similaires pour toutes les opérations sur les noeuds et les arcs parce que nous avons estimé que tous les éléments d'un graphe de structure ont la même importance (arc et noeud). La position relative de deux mots-clés est aussi importante que ces mots-clés eux mêmes. Ceci permet de donner de l'importance aussi bien aux données qu'à leur disposition spatiale dans le document.

6.1.3 Sondage de graphes

Le sondage de graphe est une méthode de mesure de similarité entre graphes, proposée dans [LW03] et qui a la particularité d'approcher la distance d'édition, tout en étant beaucoup moins complexe et beaucoup plus rapide. Cette méthode se base sur une comparaison des noeuds et de la structure d'arcs correspondant à chaque noeud entre deux graphes donnés. Elle s'applique aussi bien à des graphes non étiquetés et non orientés qu'à des graphes orientés et étiquetés (notre cas). Comme montré dans l'article [LW03], cette méthode a été appliquée avec succès sur les graphes de documents.

Le sondage de graphe se base sur les calculs suivants :

Cas des graphes non orientés et non étiquetés

Etant donné deux graphes G_1 et G_2, on calcule le degré de chaque noeud, qui correspond au nombre de noeuds n auxquels il est lié. On fait varier n de 0 au maximum que l'on trouve dans les deux graphes. On construit un vecteur représentatif de cette information pour chaque graphe. Ainsi, dans la figure 6.1(a), le vecteur représentatif du graphe G_1 est (0,2,1), ce qui est équivalent à dire que ce graphe possède un noeud lié à deux autres noeuds et deux noeuds liés à un noeud seulement. Une fois les deux vecteurs constitués, on calcule la norme $L1$ de la différence entre ces deux vecteurs, obtenant ainsi la mesure de similarité entre G_1 et G_2. Dans le cas de la figure 6.1 (a), cette mesure vaut 4 et est notée "probe", "dist" étant la distance d'édition.

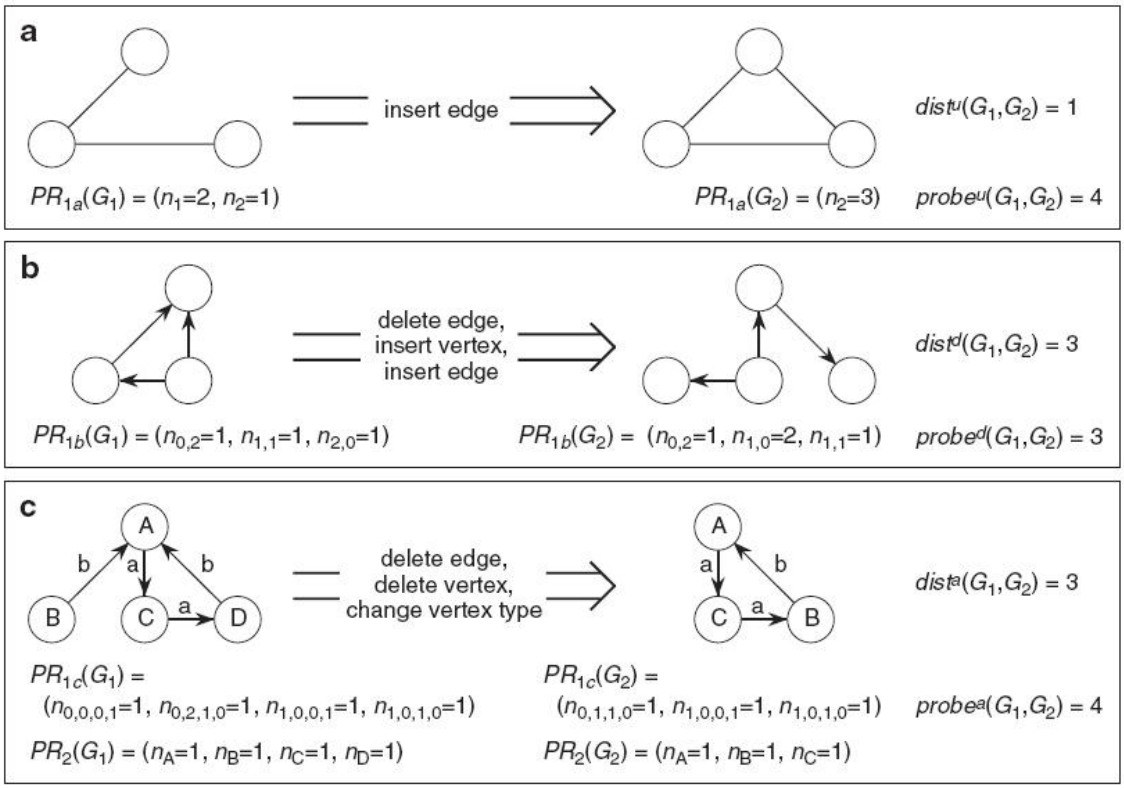


FIG. 6.1 – Sondage de graphes d'après [LW03] : (a) cas de graphes non orientés et non étiquetés, (b) cas des graphes orientés et non étiquetés, (c) cas des graphes orientés et étiquetés

Cas des graphes orientés et non étiquetés

Ce cas est plus complexe que le précédent mais il se base sur les mêmes principes. En effet, les noeuds ne sont pas étiquetés, mais une différence existe entre un noeud duquel sortent deux arcs et un autre qui est pointé par deux arcs. Le calcul de la similarité se base donc sur la mesure du nombre d'arcs sortants et entrants dans chaque noeud. Etant donné deux graphes G_1 et G_2, on calcule pour chaque graphe le nombre de noeuds ayant n noeuds entrants et m noeuds sortants, n et m variant selon les noeuds présents dans les deux graphes. Un vecteur est ainsi formé pour chaque graphe pour traduire la structure d'arcs entrants et sortants des noeuds. Il est évident que si G_1 contient un noeud ayant $n1$ noeuds entrants et $m1$ noeuds sortants et que G_2 n'en contient pas, alors l'entrée correspondante dans le vecteur décrivant G_2 est nulle. De même que précédemment, la mesure de similarité est la norme de la différence entre les deux vecteurs (figure 6.1 (b)).

Cas des graphes orientés et étiquetés

Les noeuds du graphe étant étiquetés, une mesure sur les labels de ces noeuds est introduite en plus des mesures précédentes. En effet, étant donné les deux graphes G_1 et G_2, on mesure le nombre de noeuds ayant un label X dans chaque graphe. Ainsi, pour chaque label présent dans l'ensemble des labels de G_1 et G_2, une mesure de fréquence est effectuée dans les deux graphes. Un vecteur est aussi formé pour représenter ces fréquences, et la norme de la différence est utilisée pour calculer la mesure de similarité sur les noeuds.

De plus, les arcs étant orientés, et chaque noeud ayant sa propre structure d'arcs, la fréquence de chaque structure d'arc est mesurée dans les deux graphes. Par exemple, si un noeud possède deux arcs sortants et deux arcs entrants, sa structure d'arcs est (2,2). Il faut donc chercher dans les deux graphes le nombre de noeuds ayant cette structure d'arcs, former un vecteur représentatif de cette mesure pour chaque graphe, et calculer la mesure de similarité (norme de la différence entre les deux vecteurs).

La mesure de similarité finale est la somme des deux mesures de similarités sur les arcs et les noeuds.

Sondage de graphes appliqué aux graphes de documents

Nous avons choisi d'utiliser le sondage de graphe comme mesure de similarité entre les graphes de documents. Ce choix se justifie d'abord par le fait que le sondage de graphes est moins complexe que la distance d'édition, et plus rapide en exécution aussi. En effet, étant donné deux graphes G_1 et G_2, le calcul de la structure des arcs pour chaque graphe coûte $O(|N| + \alpha\,|A|)$, où N désigne les noeuds et A les arcs, et α le nombre maximum des étiquettes des arcs. De plus, le calcul des fréquences des noeuds dans chaque graphe coûte $O(|N|)$, ce qui fait une complexité totale de $O(2.\,|N| + \alpha\,|A|)$ (complexité linéaire en le nombre d'arcs et de noeuds), alors que la distance d'édition est un problème NP difficile.

De plus, contrairement à la distance d'édition, le sondage de graphes permet de représenter les graphes de la base sous forme de vecteurs et de conserver cette représentation. Ceci permet, lors de la comparaison avec un graphe en entrée du système, un gain de temps très important puisqu'il suffit alors d'extraire un vecteur à partir du graphe en entrée et de comparer les deux vecteurs.

Les graphes de documents que nous extrayons sont constitués de deux types de noeuds : les noeuds représentant les structures (indiquant qu'il s'agit d'une SM ou d'une SMC), et ceux représentant les mots-clés d'une SMC. Les arcs quant à eux, représentent les positions relatives des structures ou des mots-clés les uns par rapport aux autres, ou indiquent qu'une SMC contient des mots-clés. Ainsi, la structure d'arcs de chaque élément du graphe est constituée de 5 éléments (C,H,B,G,D) indiquant respectivement : le nombre de mots-clés que l'élément contient (C) (donc 0 si l'élément considéré est un mot-clé), le nombre d'éléments liés par un arc étiqueté "en haut" (H), le nombre d'éléments liés par un arc étiqueté "en bas" (B), le nombre d'éléments liés par un arc étiqueté "à droite" (D) et le nombre d'éléments liés par un arc étiqueté "à gauche" (G).

Exemple : dans le cas de la figure 6.2, on a les mesures suivantes :

- les vecteurs représentant les noeuds des graphes sont les suivants :
  1. $((S,1),(rue,1),(CP,1),(ville,1))$
  2. $((S,1),(rue,1),(CP,1),(ville,0))$
- la différence entre ces deux vecteurs vaut 1.
- les vecteurs représentant les structures d'arcs sont les suivants :
  1. $(((3,0,0,0,0),1),((0,2,0,1,1),1),((0,0,1,1,0),1),((0,0,1,0,1),1))$. Chaque vecteur contient 6 composantes qui représentent $((contient, haut, bas, gauche, droite), frequence)$. Par exemple, la structure d'arc du noeud de structure S est $((3,0,0,0,0),1)$. Elle implique que ce noeud n'est lié à aucune autre structure, mais qu'il contient 3 mots-clés.
  2. $(((2,0,0,0,0),1),((0,1,0,1,0),1),((0,0,1,0,0),1))$
  3. la différence entre les deux représentations vaut alors 7, puisqu'il n'y a aucune composante en commun.

– la mesure de similarité totale donne finalement 8 (7+1), alors que la distance d'édition vaut 3 (insetion d'un noeud et de deux arcs). La relation introduite par Lopresti ($probing \leq 4.d_{edit}$) est vérifiée dans cet exemple.

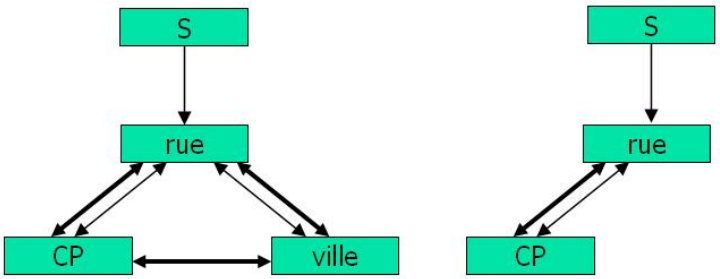


FIG. 6.2 – Deux graphes proches

Cette mesure de similarité appelée "sondage de graphes" respecte 2 des 3 conditions nécessaires à une distance : l'inégalité triangulaire et la symétrie, mais elle ne respecte pas le fait que si $d(G_1, G_2) = 0$, alors G_1 est isomorphe à G_2. Ceci fait que le sondage de graphes n'est pas considéré comme une distance. Cependant, le cas où $d(G_1, G_2) = 0$ et $G_1 \neq G_2$ se pose très peu en pratique. Cela supposerait en fait que les deux documents représentés par G_1 et G_2 contiennent exactement les mêmes structures mais que ces structures soient réparties différemment dans l'espace.

Recherche de cas proches

Grâce à la définition de la mesure de similarité entre graphes de documents, nous pouvons maintenant chercher les cas proches dans la base de cas. Dans notre application et au niveau de la résolution globale, nous nous sommes toujours contentés de chercher le cas le plus proches (un seul, et non plusieurs). Cela a toujours été suffisant pour résoudre un problème de document connu en entrée. Cependant, il n'est pas exclu dans l'avenir de considérer un ensemble de cas proches au lieu d'un seul cas proche.

6.2 Adaptation

En RàPC, adapter une solution signifie prendre la solution d'un cas source proche (généralement le plus proche) et la modifier pour qu'elle résolve le problème cible étudié. L'adaptation est un des points forts du RàPC dans la mesure où c'est cette étape qui va déterminer la solution du problème posé. Ayant 3 types de cas différents (document, SM et SMC), nous détaillons dans la suite les 3 adaptations effectuées dans notre système CBRDA.

6.2.1 Adaptation dans le cas du document

Maintenant qu'une mesure de similarité entre les graphes des documents est définie, nous pouvons adapter les solutions des cas proches sur les cas posés. Une fois le problème du document extrait, il s'agit de chercher un cas proche dans la base de cas. Le cas le plus proche est celui qui sera utilisé pour l'adaptation. En effet, un cas proche est celui qui a le plus d'éléments en commun (arcs, noeuds, structures) avec le cas étudié, ou autrement dit, c'est celui qui est le moins différent. Nous allons nous contenter dans ce chapitre de dire qu'un cas proche est un cas

dont la distance de son problème au problème cible est inférieure à un seuil T. Nous montrerons dans le chapitre suivant quel seuil T nous pouvons adopter pour estimer qu'un cas est proche ou pas. L'adaptation consiste donc à prendre la solution proposée dans le cas le plus proche et à essayer de la transposer vers le problème étudié. Dans notre approche,une fois qu'un cas proche est trouvé, il faut chercher pour chaque structure du document "source" la structure du document "cible" correspondante. Pour cela, la distance d'édition entre les différentes structures est calculée pour faire correspondre chaque structure du document étudié avec une structure du document solution, et les solutions sont alors adaptées structure par structure. Si le cas source et le cas cible sont vraiment très proches (appartiennent à la même classe), nous pouvons être certains que l'adaptation résoudra une très grande partie du problème cible (document en cours de traitement).

Si le système n'arrive pas à trouver un problème proche dans la base de cas (il y a donc une condition de rejet que nous détaillerons dans le chapitre prochain), le deuxième cycle de RàPC commence. Il faut donc résoudre le problème structure par structure. L'ensemble des solutions des structures donnera la solution finale du document. $Solution\ document = \{solutions\ structures\}$.

6.2.2 Adaptation dans le cas des SMC

L'adaptation d'une solution pour une SMC consiste à appliquer les solutions des mots-clés de la SMC cible (cas de la base) avec ceux de la SMC source. Si deux mots-clés sont identiques dans les deux SMC, alors la solution du mot-clé appartenant à la SMC cible est celle du mot-clé correspondant dans la SMC source $solution_{MC_{cible}} = solution_{MC_{source}}$. Par exemple, si la solution proposée par la SMC source est : "alphanumérique+ droite", alors la solution recherchée pour la SMC cible doit être alphanumérique et se trouver à droite du mot-clé. Une recherche d'un mot de ce type doit donc se faire à côté du mot-clé pour trouver la solution.

Afin d'optimiser la recherche de la solution, nous avons aussi introduit certaines règles à respecter lors de la recherche de solutions. Par exemple, pour le mot-clé "total", la solution numérique proposée ne doit pas être étiquetée en tant que numéro de téléphone. De même, pour un mot-clé tel que "rue", il est très peu probable que le nom de la rue soit associé à un numérique. Ces conditions permettent de mieux gérer l'extraction de la solution tout en laissant le système chercher la meilleure solution en utilisant la base de cas.

Solution à droite ou à gauche d'un mot clé

Dans ce cas, la solution est cherchée (en respectant sa nature) sur la ligne horizontale contenant le mot-clé. On commence par chercher la solution dans le même champ que le mot-clé puis dans les autres champs de la ligne horizontale. Plus on s'éloigne du mot-clé, plus la probabilité que la solution proposée soit la bonne est faible. La figure 6.3 montre un exemple de solution éloignée de son mot-clé.

FIG. 6.3 – Solution à droite du mot-clé. La solution ne se trouve pas dans le même champ que le mot-clé, mais elle se trouve sur la même ligne horizontale

Solution en bas ou en haut d'un mot clé

Dans ce cas, la solution se trouve obligatoirement dans un champ différent de celui du mot-clé. Le système cherche la solution dans les lignes horizontales au dessus ou en dessous du mot-clé. De plus, on limite les recherches aux mots qui, dans le cas d'une projection verticale, ont une intersection non nulle avec le champ contenant le mot-clé. La figure 6.4 montre la solution associée au mot-clé 'Vat Amount'. Dans cette figure, toutes les solutions sont en dessous des mots-clés.

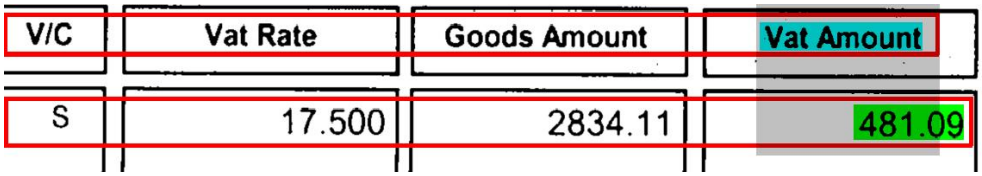


FIG. 6.4 – Solution en bas du mot-clé.

Une fois la phase d'adaptation effectuée, la solution de la SMC peut être soit complète, soit incomplète. Elle est complète si tous les mots-clés ont une solution. Elle peut être incomplète dans les cas suivants :

- Le nombre de mots-clés communs entre la SMC source et la SMC cible est inférieur au nombre de mots-clés de la SMC cible.
- Le nombre de mots-clés communs est égal au nombre de mots-clés de la SMC cible, mais le système n'arrive pas à trouver de solutions sur le document. Ceci se produit lorsque la segmentation du document en champs et lignes horizontales est de mauvaise qualité, ou à cause des erreurs d'OCR qui perturbent l'étiquetage des mots. Par exemple, si le mot recherché possède l'étiquette numérique pur, et que l'OCR donne des mots alphanumérique, la solution recherchée ne sera pas trouvée.

Recherche de solution par utilisation de connaissances génériques

Dans le cas des SMC, il arrive que certains mots-clés ne soient pas résolus ou que certaines structures ne trouvent de solution à aucun des mots-clés extraits. Cela peut provenir de l'absence totale d'information relative à ce type de mot-clé. Dans ce cas, nous constituons un ensemble de règles correspondant à chaque type de mot-clé utilisé. Par exemple, pour le mot-clé "total", on sait que la solution sera toujours un numérique (entier ou réel), peu importe sa position. Pour tous les autres mots-clés, il est facile d'avoir ce type d'informations très générales. Ces informations peuvent être utilisées en cas d'échec de la recherche de solution avec la base de cas des SMC.

Il est à noter cependant que cette base de règles générales n'est pas suffisante à elle seule pour résoudre les problèmes des SMC. En effet, et bien que Cesarini dans [CFGS03] ait présenté un travail dans lequel chaque mot-clé est interprété indépendamment des autres, ceci peut être problématique dans les cas d'applications réelles. La complexité des documents que nous traitons fait que le contexte de chaque mot-clé est très important pour son interprétation. Le fait de savoir qu'en dessus du mot-clé "total", il y a un autre mot clé "TVA", évite au système d'aller chercher des solutions sur la ligne du mot-clé "TVA", et vice versa. Le fait d'avoir des exclusions mutuelles comme celles-la évite au système de faire des recherches de solutions inutiles.

La figure 6.5 montre quelques exemples de règles génériques utilisées, stockées dans un fichier XML. En plus du mot-clé, nous fournissons la position de la solution ainsi que sa nature.

```
<node type="MC" id="2" nature="NETAMOUNTLabel" nat_solution="c" pos_solution="3"/>
<node type="MC" id="3" nature="NETAMOUNTLabel" nat_solution="c" pos_solution="2"/>
<node type="MC" id="4" nature="TABTotalAmountLb" nat_solution="c" pos_solution="2"/>
<node type="MC" id="5" nature="VATAMOUNTLabel" nat_solution="c" pos_solution="3"/>
<node type="MC" id="6" nature="VATAMOUNTLabel" nat_solution="c" pos_solution="2"/>
<node type="MC" id="7" nature="TOTAMOUNTLabel" nat_solution="c" pos_solution="3"/>
<node type="MC" id="8" nature="TOTAMOUNTLabel" nat_solution="c" pos_solution="2"/>
<node type="MC" id="9" nature="DELIVNUM_LABEL" nat_solution="c" pos_solution="2"/>
<node type="MC" id="10" nature="DELIVNUM_LABEL" nat_solution="c" pos_solution="3"/>
```

FIG. 6.5 – Exemple de quelques règles génériques

Perspective pour l'adaptation pour les SMC

Une perspective intéressante pour l'adaptation des solutions pour les SMC concerne l'utilisation de la base de cas avant d'utiliser les connaissances génériques. Soit $S1$ une SMC que l'on doit interpréter grâce à la base de cas. Supposons que cette SMC contienne N mots-clés, et que la base permette d'en interpréter uniquement M($<$N). Une solution consiste à considérer le graphe constitué uniquement des mots-clés non résolus (M-N mots-clés) et les arcs qui lient ces mots-clés pour obtenir un nouveau graphe (une nouvelle SMC) qui sera à son tour interprétée grâce à la base de cas de structures. Une telle stratégie permet de garder les liens entre les mots-clés restants. Cette opération peut être effectuée jusqu' à ce qu'il ne reste qu'un seul mot-clé, dans quel cas, il faut utiliser les connaissances génériques.

6.2.3 Adaptation dans le cas des SM

Dans le cas des SM, la solution consiste en l'interprétation de chaque colonne quand cela est possible, plus l'extraction de règles entre les différents champs du motif.

- Si un motif similaire est trouvé dans la base de cas, nous essayons d'appliquer la règle associée aux champs du motif étudié. Si la règle s'applique parfaitement, c'est que les deux motifs sont identiques. Pour vérifier l'application de la règle, il suffit de l'appliquer sur une ligne du motif (les autres lignes étant identiques dans les cas d'une SM à motif simple).
- Si la règle ne s'applique pas, alors un autre motif similaire est cherché dans la base, tel que $d(motif, motif_{base}) \leq 1$ (d étant la distance d'édition). De même que précédemment, nous essayons d'appliquer les règles du $motif_{base}$ sur le motif étudié. Ceci concerne particulièrement les champs numériques, alphanumériques et numériques purs. Lors de l'application des règles du $motif_{base}$, nous nous intéressons particulièrement à ces champs parce qu'une règle existe entre ces champs. Les champs alphabétiques dans les lignes des tableaux servent souvent à décrire un objet, ils ne contiennent pas d'information relative aux autres champs de la ligne.

L'adaptation dans le cas des SM trouve son intérêt surtout pour les SM composites. En effet, pour les motifs simples, il n'est pas souvent nécessaire de chercher les règles dans la base de cas, puisqu'il suffit d'effectuer certaines combinaisons sur les champs numériques pour retrouver la règle entre ces champs. Par contre, si le motif est composite et qu'il est sur plusieurs lignes, il est beaucoup plus facile de retrouver les règles liant les différents champs en retrouvant un motif similaire dans la base de cas.

Dans la perspective de notre travail, nous allons intégrer dans la base de cas des structures à motifs l'information des étiquettes de chaque colonne. C'est en utilisant cette information que l'utilisation de la base de cas devient avantageuse par rapport à une simple recherche sur les

motifs extraits.

6.3 Expérimentations

Après avoir défini formellement notre approche, il est nécessaire de la tester afin de la valider.

6.3.1 Base de tests

Nous avons utilisé une base de documents très hétérogènes. Ces documents proviennent de différents fournisseurs et entreprises. A chaque document est associé un fichier XML contenant tous les mots du document, reconnus par l'OCR utilisé par l'entreprise.

Une fois que le graphe du document est formé, deux traitements sont possibles selon que le document est de classe connue ou non.

Dans nos expérimentations, nous avons choisi de limiter les tests pour le cas le plus simple, celui du premier cycle de RàPC (cas où le document est de classe connue). Nous avons donc construit une base de cas contenant 10 représentants de classes (10 cas) et nous avons testé le premier cycle sur 150 documents. Ces documents ont tous un représentant parmi ces 10 représentants dans la base de cas. Il suffit donc de trouver pour chaque document le cas le plus proche dans la base, et d'appliquer sa solution sur le document en cours.

Concernant le deuxième cycle de RàPC (au niveau des structures), nous avons choisi de le tester sur un nombre beaucoup plus grand de documents, puisqu'il s'agit d'une résolution par partie. Résoudre un problème sans le connaître en entier est naturellement plus difficile. Pour cela, nous avons utilisé une base de 850 documents différents. La base de cas de structures est composée de 300 cas tous résolus, et dont 20% seulement ont été extrait à partir des documents de la base de test.

Les cas de la base de cas de documents sont modélisés par des documents XML (le graphe du document est traduit dans un document XML), de même que la base de cas de structures qui est représentée en entier dans un seul document XML dans lequel on peut ajouter au fur et à mesure les cas de structures résolus.

Résultats

Afin de mesurer la qualité des résultats, nous avons associé à tous les documents de tests des documents résultats fournis par ITESOFT. Ces derniers contiennent les résultat recherchés, que nous comparerons avec les résultats de notre système.

Nous avons évalué notre système à l'aide de la mesure 6.1 sur X où X correspond à une information de vérité sur les SM, les SMC ou les documents :

$$R_X = \frac{|solutions\ correctes\ |}{|solutions\ dans\ X|}. \tag{6.1}$$

Nous avons aussi essayé qualifié les erreurs, c'est à dire les solutions qui n'ont pas trouvé de correspondance dans les documents de vérité. Les erreurs peuvent être soit causées par le système CBRDA lui même (exemples dans la suite), soit provenir d'erreurs d'OCR.

Les résultats obtenus sont présentés dans le tableau 6.1 :

Avec 85.29% de bons résultats pour les documents lorsque ceux ci sont de classe connue, nous pouvons donc estimer que notre idée consistant à considérer que deux documents d'une même classe peuvent être analysés et interprétés d'une manière similaire s'avère fondée. Par contre, la résolution locale (structure par structure) donne quant à elle un score inférieur de près de 9%.

	R_{doc}	R_{SMC}	R_{SM}
Résolution globale	85.29%	82.22%	88.75%
Résolution locale	76.33%	76.38%	76.28%

TAB. 6.1 – Résultats de l'approche CBRDA

Cela était néanmoins prévisible puisque nous analysons des documents de classe inconnue, mais cela reste quand même un très bon score. En effet, le but de notre système n'est pas d'avoir des résultats proches du 100%, score impossible à réaliser au vu de plusieurs contraintes liées à la qualité des documents, à l'OCR et à la complexité des informations présentes dans les documents, mais d'essayer d'extraire le maximum d'informations afin d'analyser au mieux chaque document. En résumé, il s'agit de minimiser l'effort de l'utilisateur final de notre système. Si le système permet de réduire de 85% le travail de l'utilisateur, c'est alors déjà un très bon résultat.

Nous avons classé les erreurs obtenues en deux catégories différentes : les erreurs du système CBRDA, et les erreurs d'OCR. Les erreurs d'OCR concernent toutes les solutions qui ont été mal lues par l'OCR. Par exemple, si la solution du document de vérité est "77,13" et que l'OCR la lit "77,I3", cette solution est incorrecte et est considérée comme une erreur de substitution de l'OCR. Les erreurs du système sont quant à elles de trois types : une solution qui n'est pas trouvée, une solution qui est donnée mais qui n'est pas juste (extraction de la solution d'une autre mot, c'est typiquement une confusion), et finalement une solution complètement fausse (extraction d'un mot qui n'est pas du tout lié au problème posé).

Voici le détail de ces erreurs :

Pour les documents de classe connue (ayant un document proche dans la base de cas), les 14.71% manquants correspondent à 7.76% d'erreurs du système (mauvaise solution, pas de solution, confusion avec d'autres solutions) et 6.95% d'erreurs d'OCR.

Pour les documents de type inconnu, les erreurs pour les SMC sont dues à :

- 15.57% d'erreurs du système (mauvaise solution, pas de solution, confusion avec d'autres solutions) ;
- 8.08% d'erreurs d'OCR.

Pour les documents de type inconnu, les erreurs pour les SM sont dues à :

- 16.66% d'erreurs du système (pas de détection de tableau, lignes manquantes dans le tableau, mauvaise interprétation) ;
- 7.14% d'erreurs d'OCR (par exemple, le mot "23.7" est lu comme "23.T", deux champs sont fusionnés, etc...).

L'OCR utilisé par ITESOFT est un OCR du marché professionnel, les erreurs produites ne sont pas dues à l'OCR lui même mais plutôt à

- la qualité des documents. Dans notre base de test, nous avons au moins 15% de documents de mauvaise qualité (c'est donc soit le document qui est très mauvais, soit le sanner qui n'a pas bien fonctionné) ;
- des informations manquantes dans le texte tels que des caractères ou des mots manquants. Comme précisé dans le chapitre 1, l'OCR donne de très bons résultats (de l'ordre de 99%) sur des documents de bonne qualité et 85% sur des documents de qualité moyenne. Or, si on reporte ce résultat au niveau des mots (on veut donc connaître le nombre de mots qui ont été lus correctement), ce résultat peut très facilement descendre à 85% pour le premier cas et 70% pour le second cas. Ceci n'est pas dû à la qualité de l'OCR, mais au fait que le nombre de mots est très inférieur au nombre de caractères dans le document. Ainsi, si un caractère est mal reconnu, il influe très peu sur le taux de reconnaissance des

caractères, mais le mot contenant ce caractère sera considéré comme mal reconnu, ce qui influe beaucoup plus sur le taux de reconnaissance des mots.

D'après ces résultats, il est clair que certains points restent à améliorer avant d'atteindre des résultats de meilleur niveau. Cela fait partie des perspectives de ce travail.

Voici quelques exemples d'erreurs. Dans la figure 6.6, le mot-clé TVA est bien extrait, la solution est bien localisée, mais l'OCR n'arrive pas à bien lire le vrai montant (à cause de la ligne qui traverse le mot). Cette solution compte donc pour une erreur d'OCR.

Dans la figure 6.7, la solution de deux mots-clés est trouvée par la base de cas. Le mot-clé "Total H.T" demeure non résolu. En utilisant la base de règles, la solution proposée est : "total + numérique" en bas ou "total + numérique" à droite. C'est la deuxième règle qui sera donc utilisée pour interpréter le mot-clé. En enrichissant la base de cas au fur et à mesure, ce cas de figure peut être résolu par la suite uniquement par la base de cas de structures.

La figure 6.8 montre un exemple d'erreur de détection de tableau. La première ligne du tableau n'est pas détectée car elle est mal étiquetée à cause du bruit qui l'environne (les mots sont mal reconnus, ils sont ainsi mal étiquetés, et la ligne n'a ainsi pas le même motif que les deux autres lignes).

FIG. 6.6 – Exemple d'une solution bien extraite, mais mal reconnue par l'OCR.

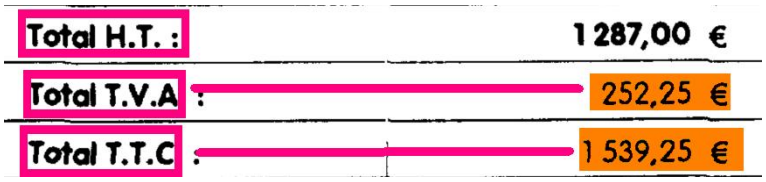


FIG. 6.7 – Exemple d'une SMC contenant 3 mots. Deux mots seulement sont résolus.

CODE ARTICLE	DESIGNATION	QTE	QTE TOTALE	P.V. UNITAIRE	P.V. TOTAL	CODE TVA
902830309	MOTORECLAT 1 AEROSOL	48	4,00	56,780	227,12	1
900911100	DP 1000 CHAMOIS: ROULEAU	6	6,00	5,490	32,94	1
900703100	R.P.H	18	3,00	1,470	4,41	1

FIG. 6.8 – Exemple d'une SM contenant 3 lignes. Deux lignes seulement sont détectées.

6.3.2 A propos du stockage des problèmes et solutions

Dans notre approche, nous manipulons des graphes de documents. Après les avoir extraits, il devient nécessaire de les stocker afin d'éviter une nouvelle extraction. Ainsi, à chaque document, nous avons associé un fichier XML, contenant le graphe du document (avec les structures, leurs positions par rapport aux autres structures, ainsi que les informations associées aux mots-clés). Dans le même fichier, il est aussi possible de spécifier la solution de chaque mot-clé et de chaque motif.

Nous avons aussi créé des fichiers correspondant aux documents de vérité. Ces documents, stockés aussi en XML, permettent de comparer rapidement la solution extraite avec la solution réelle.

Notons que dans les deux cas, nous avons utilisé des noms de balises correspondant à notre application. Ceci facilite la compréhension même par un utilisateur externe au système. Par exemple, dans la figure 6.9, une structure à mot-clé est représentée (avec ses mots-clés et ses arcs, ainsi que les solution de ses mots-clés). Il suffit d'utiliser la matrice d'adjacence du graphe pour avoir les différentes relations spatiales entre les mots.

```
- <ocr lastid="72" left="0" top="0" right="2946" bottom="3506">
  - <node type="SMC" id="1" name="montant" rank="0">
      <node type="MC" id="2" nature="NETAMOUNTLabel" nat_solution="c" pos_solution="3"/>
    - <node type="MC" id="3" nature="NETAMOUNTLabel" nat_solution="c" pos_solution="2">
        <node type="Edge" id="4" source="NETAMOUNTLabel" dest="NETAMOUNTLabel" etiquette="d"/>
        <node type="Edge" id="5" source="NETAMOUNTLabel" dest="VATAMOUNTLabel" etiquette="d"/>
        <node type="Edge" id="6" source="NETAMOUNTLabel" dest="NETAMOUNTLabel" etiquette="d"/>
        <node type="Edge" id="7" source="NETAMOUNTLabel" dest="TOTAMOUNTLabel" etiquette="d"/>
      </node>
      <node type="MC" id="8" nature="NETAMOUNTLabel" nat_solution="c" pos_solution="3"/>
    - <node type="MC" id="9" nature="NETAMOUNTLabel" nat_solution="c" pos_solution="2">
        <node type="Edge" id="10" source="NETAMOUNTLabel" dest="VATAMOUNTLabel" etiquette="d"/>
        <node type="Edge" id="11" source="NETAMOUNTLabel" dest="NETAMOUNTLabel" etiquette="d"/>
        <node type="Edge" id="12" source="NETAMOUNTLabel" dest="TOTAMOUNTLabel" etiquette="d"/>
      </node>
    - <node type="MC" id="13" nature="VATAMOUNTLabel" nat_solution="c" pos_solution="3">
        <node type="Edge" id="14" source="VATAMOUNTLabel" dest="NETAMOUNTLabel" etiquette="d"/>
        <node type="Edge" id="15" source="VATAMOUNTLabel" dest="TOTAMOUNTLabel" etiquette="d"/>
      </node>
    - <node type="MC" id="16" nature="NETAMOUNTLabel" nat_solution="c" pos_solution="3">
        <node type="Edge" id="17" source="NETAMOUNTLabel" dest="TOTAMOUNTLabel" etiquette="d"/>
      </node>
      <node type="MC" id="18" nature="TOTAMOUNTLabel" nat_solution="c" pos_solution="3"/>
    </node>
  - <node type="SMC" id="19" name="montant" rank="1">
      <node type="MC" id="20" nature="TABTotalAmountLb" nat_solution="c" pos_solution="2"/>
    </node>
```

FIG. 6.9 – Exemple d'une partie d'un document XML créé pour représenter un problème de document et sa solution

Les différentes balises SMC, MC, Edge représentent respectivement structure à mot-clé, mot-clé et arc. Les attributs "nature", "nat_solution" et "pos_solution" représentent respectivement le contenu du mot-clé, la nature de la solution associée et sa position. Finalement, les attributs "source", "dest" et "étiquette" représentent le noeud origine d'un arc, son noeud destination et l'étiquette position qu'il possède.

6.4 Conclusion

Nous avons présenté dans ce chapitre les différentes étapes du RàPC dans le système CBRDA. Pour un problème élaboré, nous avons essayé de lui trouver une solution globale en cherchant un cas proche dans la base de cas de documents. Pour cela, nous avons utilisé le sondage de graphes, qui nous a paru être le plus approprié pour cette application. Etant dans un cadre industriel où le temps d'exécution et de traitement est un facteur important, nous avons vite écarté l'idée d'utiliser la distance d'édition comme distance entre graphes au vue de la taille des graphes utilisés et de la taille de la base qui peut croître rapidement. Le sondage de graphes peut encore être amélioré dans notre application en introduisant des notions de coûts par exemple lors des comparaisons (en donnant plus d'importance par exemple aux structures d'arcs).

Si un cas proche est trouvé dans la base de cas de documents, nous avons montré comment l'adaptation par recopie s'effectue afin d'extraire la solution du document en cours. Nous avons aussi montré que lors de l'absence de ce cas proche (nous allons expliciter ce seuil de ressemblance dans le chapitre suivant), le document est traité structure par structure. Cette idée s'est avérée être une bonne alternative puisqu'elle permet de résoudre jusqu' à 75% des cas posés, ce qui est un très bon taux pour des documents qui n'ont jamais été traités auparavant par le système.

Dans cette partie, nous avons utilisé la distance d'édition aussi bien sur les graphes que sur les chaînes représentant les motifs des zones tabulaires. D'autres distances auraient pu être utilisées, mais cela peut être aussi considéré comme perspective de ce travail. Nous avons plus essayé de mettre l'accent sur le fait que les deux cycles de RàPC permettent de se compléter l'un l'autre, et que surtout, le deuxième cycle de RàPC réussit à donner un apport très évident au système.

Toutes les étapes citées dans ce chapitre ont été implémentées en C++ et notre système a été testé sur près de 1000 factures. Les résultats obtenus ont été confrontés à des documents de vérité fournis par ITESOFT et nous avons comparé nos résultats avec les leurs. Nous pensons que le système CBRDA donne de très bons résultats d'un point de vue industriel. Il devrait être exploité prochainement par cette entreprise dans un cadre réel.

7

Apprentissage et classification de la base de cas

Sommaire

Nous nous intéressons dans ce chapitre à la partie apprentissage dans le RàPC (aussi bien dans le premier cycle que dans le deuxième cycle de RàPC dans notre système). L'apprentissage est indispensable dans tout système de RàPC dans la mesure où il permet de prendre en compte les nouveaux cas résolus, de les intégrer et de les réutiliser par la suite pour la résolution d'autres cas. Le problème de l'apprentissage est lié à celui de la classification de la base de cas. Cette classification a pour entrée : la base de cas à un instant T donné et un nouveau cas, et doit donner en sortie : la base de cas après intégration du nouveau cas.

Nous nous intéressons dans ce chapitre à la classification de la base de cas. Nous disposons de deux bases de cas : une base de cas de structures et une base de cas de documents. Les deux bases, même si elles sont initialement très petites, peuvent rapidement augmenter de taille, et il serait dommage de ne pas essayer de les classer afin d'optimiser l'accès à ces bases. Un accès séquentiel à ces bases de cas peut très rapidement devenir très coûteux, non seulement à cause du nombre de cas, mais aussi à cause de leur représentation en graphes. La classification de la base peut résoudre ce problème. En effet, en mettant sous la même étiquette des cas similaires, on regroupe beaucoup de cas en un seul, et au lieu de faire M comparaisons avec les M cas de la base, nous n'avons qu'à faire $M1 << M$ comparaisons, M1 étant le nombre de classes (en comparant à chaque fois avec le représentant de la classe).

Plusieurs approches ont été proposées dans la littérature pour organiser des bases de cas lorsque les cas sont des graphes [Per98b], pour faire de la classification incrémentale [Fri94b],

ou pour accélérer le calcul de la distance entre un graphe en entrée et des graphes dans une base [MB98], nous allons les étudier avant de proposer une approche originale aussi bien dans le domaine du document que celui du RàPC, se basant sur les réseaux de neurones.

L'organisation de la base de cas suppose que nous devons être capables de gérer les nouveaux cas qui n'ont jamais été vus auparavant, en même temps que ceux qui existent déjà dans la base. Ceci nous amène donc à considérer deux hypothèses de traitement :

- La première hypothèse consiste à classer les M premiers cas de la base en M1 classes. Les nouveaux cas sont alors rajoutés aux classes déjà définies (en fonction de leur proximité). Cette solution nécessite de refaire la même opération de classification de la base périodiquement. Cette hypothèse peut se poser sérieusement et n'est donc envisageable que si la base n'est pas alimentée en continu par de nouveaux cas.
- La deuxième hypothèse consiste à effectuer une classification incrémentale, qui augmente son nombre de classes (ou qui le diminue) en fonction des entrées qui se présentent. C'est l'hypothèse qui nous a semblé être la plus intéressante (et c'est celle que nous avons choisi) puisqu'elle permet de s'adapter aux nouvelles entrées sans que cela ne mette en cause les anciennes classifications

Deux problèmes se posent donc :

- quel type de classification incrémentale choisir ?
- comment adapter l'algorithme choisi sur des graphes ?

Nous proposons des réponses à ces questions dans ce chapitre.

7.1 Classification incrémentale : état de l'art

Dans [GC00], une classification incrémentale est caractérisée par le fait que :

- les exemples d'apprentissage soient disponibles les uns après les autres (un à la fois). Ils ne sont pas tous disponibles au début, mais le deviennent au fur et à mesure.
- l'apprentissage peut continuer indéfiniment.

Cette caractérisation de la classification incrémentale la rend très différente de la classification "statique". En effet, les classifieurs statiques tels que les réseaux de neurones, les réseaux bayésiens, les chaînes de Markov... utilisent en général une base d'apprentissage statique sur laquelle aucun ajout ni suppression ne sont effectués. Les exemples d'apprentissage sont disponibles tous en même temps et des informations générales relatives aux bases d'apprentissage peuvent être extraites (moyenne, écart type, variance...).

La classification incrémentale commence avec peu de classes. Puis, en fonction des données qui arrivent, les classes existantes sont modifiées, voire supprimées. De nouvelles classes peuvent aussi être créées pour modéliser les nouvelles données.

La classification incrémentale est utilisée dans divers domaines. Dans les domaines de l'analyse de l'écriture et/ou du document, nous avons recensé plusieurs travaux (il n'y a pas eu d'état de l'art exhaustif des travaux utilisant la classification incrémentale). Nous proposons un bref aperçu de méthodes incrémentales dans ce qui suit.

La classification incrémentale se fait aussi bien par des classifieurs supervisés que non supervisés. Dans le cadre de la classification supervisée, elle a été appliquée par Polikar [PUUH01] à la classification de données synthétiques et par Hébert [HPG99]à la détection d'écriture manuscrite. Nous ne détaillerons pas ces deux méthodes parce que nous ne nous sommes pas placés dans un contexte de classification supervisée.

Les travaux détaillés dans la suite sont relatifs à des classifieurs non supervisés. Khy [KIK06] propose d'utiliser un Kmeans incrémental en prenant en compte la similarité intra-classe (voir

paragraphes suivants pour les détails). Il applique cet algorithme à la classification de documents web. Dans le cadre de la même application, Hammouda [HK03] étudie les variations des histogrammes de similarités de classes et classe ainsi incrémentalement les données Web disponibles. Ces deux applications sont détaillées dans la suite de ce chapitre.

D'autres approches construisent incrémentalement une hiérarchie de classes. En renforçant des classes, en en supprimant d'autres ou en en fusionnant plusieurs, la hiérarchie construite traduit la distribution des données dans l'espace. C'est ce que fait Seong [SKP92] pour classer des objets modélisés en graphes, et Perner [Per98b] pour classer une base de cas d'images représentées en graphes. La différence entre ces deux approches est que la première utilise l'entropie des classes alors que la deuxième utilise les variances intra et inter classes. D'autres approches peuvent être dérivées à partir de celles là, il suffit donc de s'appuyer sur des caractéristiques intrinsèques des classes.

La dernière méthode étudiée, et c'est celle que nous adoptons pour cette thèse, est la méthode neuronale, non supervisée. Les premières approches incrémentales neuronales et non supervisées ont été introduites par Fritzke [Fri94b] [Fri94a]. La première méthode proposée par Fritzke est le "Growing Neural Gas" (GNG) : c'est un réseau de neurones non supervisé qui commence avec deux neurones puis crée des neurones périodiquement afin d'épouser au mieux l'espace des données qui arrivent au fur et à mesure. Il a été appliqué à la classification de données synthétiques. La deuxième méthode est les "Growing Cell Structures" (GCS) : c'est un réseau de neurones non supervisé qui commence avec trois neurones et qui crée de nouveaux neurones en fonction de l'arrivée des données (même principe général que le GNG), et il s'arrête au bout d'un nombre maximal de neurones et un nombre d'itérations prédéfini. Il a de même été appliqué à des données synthétiques.

Diherty [DAD] a proposé une extension du GNG, le TreeGNG (arbre appris sur le GNG) : les neurones du GNG sont classés et regroupés dans les noeuds d'un arbre. C'est une classification hiérarchique qui prend les neurones du GNG comme entrée. De même, Hodge [HA01]a proposé le "TreeGCS" (arbre appris sur le GCS). La différence majeure avec le "TreeGNG" est le fait d'utiliser le GCS au lieu du GNG.

Finalement, Prudent [PE05] a proposé un "GNG incrémental" (IGNG). C'est en fait une version améliorée du GNG dans la mesure où les noeuds ne sont plus ajoutés périodiquement, mais en fonction de la distance séparant les entrées de leurs neurones les plus proches. Afin de vérifier cette amélioration, Prudent et al. l'ont testé sur des données synthétiques, ainsi que sur des données de la base manuscrite MNIST.

Kmeans incrémental

Dans [KIK06], un Kmeans incrémental est utilisé pour classer des documents Web. Inspiré du très populaire algorithme Kmeans, le nouvel algorithme possède en plus la capacité de rajouter des classes à un ensemble de classes déjà apprises. Il se base sur la notion de similarité intra-classes. Cette similarité peut s'écrire comme suit : $G = \sum_{p=1}^{k} |C_p| \, avgsim(c_p)$, où $|C_p|$ est le nombre d'éléments dans la classe C_p, et $avgsim(c_p)$ est la similarité inter-classe dans la classe C_p qui est calculée comme suit : $avgsim(C_p) = \frac{1}{|C_p|(|C_p|-1)} \sum_{d_i \in C_p} \sum_{d_j \in C_p, d_i \neq d_i} sim(d_i, d_j)$, où d_i est un document.

Avec la notion de similarité intra-classe, l'algorithme Kmeans incrémental est celui donné dans 2 :

Soit K classes initiales et un seuil δ :
Calculer les similarités intra-classes
Calculer G
Pour chaque nouveau document d **faire**
  pour chaque classe, calculer la similarité intra-classe si le document d est ajouté à cette classe
  Assigner d à la classe dont la similarité augmente le plus.
  Si (aucune similarité intra-classe n'est augmentée) **Alors**
    ajouter d à une liste de documents à part (outlier list)
  Fin Si
Fin Pour
Recalculer les centres des classes ainsi que G
Si ($\frac{(G_{nouveau}-G_{ancien})}{G_{ancien}} < \delta$) **Alors**
  arrêter l'apprentissage
Sinon
  Reclasser les documents de la outlier list
Fin Si

Algorithme 2: Algorithme Kmeans incrémental

Avec la dernière condition, nous constatons que cet algorithme permet aussi de faire de l'apprentissage sur une base de documents fixe (non évolutive). L'extension à une base de documents qui s'enrichit au fur et à mesure du temps est facile à faire. En effet, il suffit de ne plus considérer la dernière condition et de faire l'apprentissage en continu. De plus, en éliminant cette condition, le calcul de G devient inutile. Les documents mis dans la liste de documents à part peut évoluer au fur et à mesure pour constituer par la suite des classes indépendantes, d'où l'intérêt du Kmeans incrémental.

Méthode de Hammouda et al.

Dans [HK03], une classification incrémentale utilisant les histogrammes de similarité des classes est présentée. Un histogramme de similarité de classes est un histogramme traduisant la similarité inter-documents dans une classe. Un histogramme de similarités est défini comme suit : il est bâti à partir des similarités inter-documents (deux à deux). Chaque barre dans cet histogramme représente le nombre de documents similaires dans un intervalle donné. A partir de cet histogramme, une mesure appelée Historgam-Ratio (HR) correspondant à la cohésion d'une classe est définie pour évaluer la qualité d'une classe. Si l'ajout d'une donnée à une classe diminue cette valeur HR, on sait alors que la donnée ne doit pas appartenir à la classe. L'algorithme final présenté dans ce papier est détaillé dans 3. Cet algorithme utilise donc des caractéristiques propres à chaque classe pour ajouter ou non un nouveau document au sein de la classe dont le HR augmente le plus, ou dont la variation du HR est inférieure à un seuil ϵ prédéfini . Cette technique est indépendante de la connaissance de paramètres liés à l'ensemble des données déjà présentées. Appliquée à la classification incrémentale de documents, cet algorithme a donné de meilleurs résultats que les algorithmes : Kppv et HAC (Hierarchical agglomerative clustering), prouvant ainsi la supériorité de la classification incrémentale à d'autres approches de classification

statique.

$L \leftarrow Listeclusters$, 2 seuils : HR_{min} et ϵ :
Pour chaque document **faire**
  Pour chaque classe C dans L **faire**
    $HR_{ancien} = HR_C$.
    On simule le rajout de d à C.
    $HR_{nouveau} = HR_C$
    Si $((HR_{nouveau} > HR_{ancien})$ ou $((HR_{nouveau} > HR_{min})$ et $(HR_{ancien} - HR_{min}) < \epsilon))$ **Alors**
      Ajouter d à C
    Fin Si
  Fin Pour
  Si (D n'est ajouté à aucune classe) **Alors**
    Créer une nouvelle classe C
    Ajouter d à C
    Ajouter C à L
  Fin Si
Fin Pour

Algorithme 3: Algorithme de Hammouda et Kamel 3

Les deux algorithmes présentés sont similaires en plusieurs points :
- d'abord ils peuvent tous les deux commencer avec une ou plusieurs classes
- ils utilisent des caractéristiques relatives aux classes les plus proches. La proximité dans le Kmeans incrémental correspond à la variation maximale de similarité inter-classes. Dans l'algorithme 3, la proximité correspond aussi à une variation de l'indice HR. Ces deux notions de classe la plus proche, ou la plus adéquate est différente de la notion de calcul de distance entre une entrée et un représentant de classe. D'ailleurs, dans l'algorithme 3, la notion de centre ou de représentant de classe n'existe même pas.

Méthodes neuronales

Les réseaux de neurones permettent de faire de la classification incrémentale. Nous ne détaillerons dans ce paragraphe que les algorithmes les plus importants (GNG et IGNG), afin de les améliorer pour notre application.

Le Growing Neural Gas (GNG), proposé par Fritzke [Fri94b], est un réseau de neurones non supervisé, qui adapte sa topologie en fonction de la topologie de l'espace d'entrée. Un neurone dans un GNG est de la même nature que l'entrée traitée. Donc si l'entrée est un vecteur de dimension 10, les neurones le seront aussi. Le GNG de part ses suppressions et rajout de neurones permet d'épouser un espace d'entrée d'une manière beaucoup plus optimale que les réseaux de neurones non supervisés classiques tels que : les cartes de Kohonen, le Neural Gas, la Growing Grid, et ce, grâce à la liberté de ses neurones et à l'absence de liens obligatoires entre les différents neurones du réseau. Par exemple, dans la carte de Kohonen, un neurone est obligatoirement lié à au moins deux autres neurones (neurones dans les coins ou les bords de la carte) et peut être lié

à quatre neurones dans une carte à topologie rectangulaire. Ceci peut limiter considérablement les mouvements dans l'espace de ces neurones et explique que la couverture d'un espace d'entrée par une carte de Kohonen est souvent moins bien effectuée que par un réseau de type GNG. La liberté des neurones dans l'espace, ainsi que la liberté d'ajouter des neurones dans les zones sous représentées, ou en enlever dans les zones sur-représentées permet donc au GNG d'épouser au mieux l'espace d'entrée. Contrairement à cela, dans les réseaux de type carte de Kohonen, le choix du nombre de neurones initial est une contrainte parfois insurmontable en classification. Il faut alors refaire parfois la classification avant d'avoir une carte qui traduit bien la distribution des données dans l'espace.

Bien que les Growing Cell Structures [Fri94a] aient été proposés par Fritzke presque en même temps que le GNG, nous nous concentrons uniquement sur le GNG car il a été prouvé que celui-ci est meilleur que le GCS (bien qu'ils aient un principe globalement similaire). En effet, dans [DH98], une comparaison de quatre réseaux de neurones (GNG, GCS, MLP et les ARTMAP flous (FAM)) sur quatre bases différentes a donné le classement final suivant : GNG, GCS, MLP et FAM, d'après les critères ci-après : taux d'erreur sur les bases de test, rapidité de la convergence, nombres de neurones ajoutés pendant l'apprentissage et l'indépendance des paramètres du réseau. Nous avons nous même constaté que le GNG est très indépendant des paramètres en entrée, il en est de même aussi pour le IGNG ([PE05]).

Les figures 7.1 et 7.2 [3] montrent l'efficacité du GNG à épouser l'espace d'entrée et sa supériorité sur les cartes de Kohonen. Le fait de commencer avec 2 neurones pour arriver à couvrir par la suite l'espace d'entrée est un avantage très important pour le GNG.

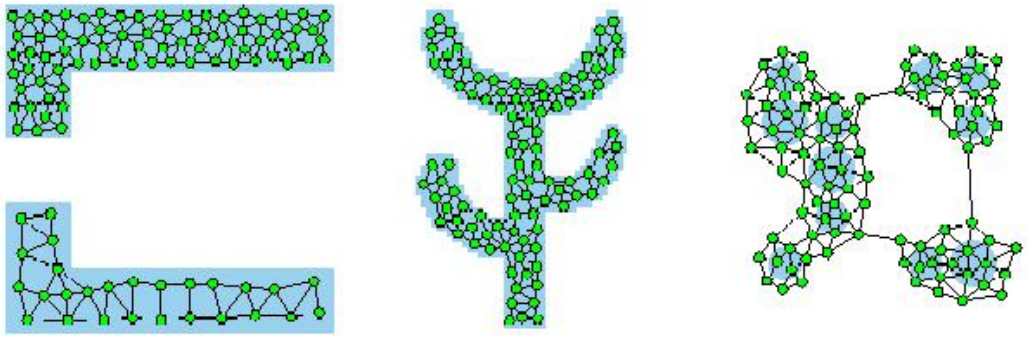


FIG. 7.1 – Incrémentalité du GNG et son adaptation à l'espace d'entrée

La première étape du GNG est de trouver les neurones les plus proches pour une entrée donnée E. Ceci permet de savoir à quelle classe la donnée va éventuellement appartenir. L'augmentation de l'âge des arcs liés au neurone vainqueur sert à mettre en évidence que ce neurone a été sollicité. Si un des arcs liés est sollicité plus de a_{max}, il est alors supprimé. Si cette suppression conduit à des neurones isolés, ils sont aussi supprimés.

La création d'un nouveau neurone est conditionnée par le pas (la granularité, la finesse) que l'on veut avoir dans le résultat final. Si on veut obtenir des neurones espacés, un pas assez grand doit être choisi. Cependant, dépendre de ce pas pour créér ou non un nouveau neurone peut être discutable. En effet, il se peut qu'un GNG à un moment donné n'ait pas besoin d'une insertion d'un nouveau neurone. De même, il se peut qu'un réseau ait vraiment besoin de l'insertion de

[3] www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/GNG.html

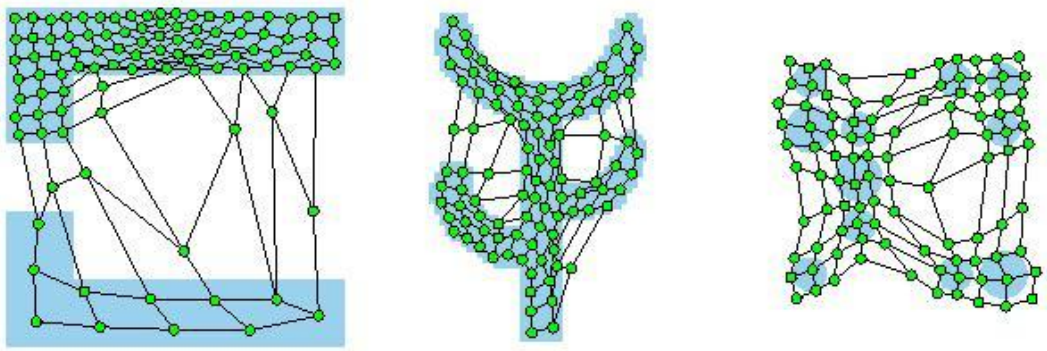


FIG. 7.2 – Adaptation d'une carte de Kohonen à un espace d'entrée

plusieurs nouveaux neurones et non d'un seul, mais cela ne se fait qu'une fois tous les λ entrées. Nous allons montrer dans la suite comment le IGNG remédie à ce problème.

7.2 Classification de la base de cas dans CBRDA

La classification de la base de cas s'impose car :
- la taille des bases peut très rapidement devenir très importante (des milliers de cas de documents, des milliers de cas de structures). La recherche séquentielle des cas les plus proches demandera beaucoup de temps.
- le regroupement des cas similaires de la base peut aider à mieux résoudre un problème en entrée du système.

Suite à l'état de l'art précédent, nous avons choisi de nous diriger vers des systèmes de type neuronal. Ce choix est motivé par le fait que les réseaux de neurones non supervisés ont déjà donné de très bons résultats dans la littérature sur la classification de très grandes masses de documents (utilisation de cartes de Kohonen pour classer des millions de documents par exemple [KHLK98]), d'où l'espoir de voir des réseaux plus performants et moins contraints donner de meilleurs résultats (GNG, IGNG). Les réseaux de neurones non supervisés ont aussi été utilisés pour classer des documents représentés en graphes. Marinai, en effet, a utilisé dans [SEG06] une carte de Kohonen qu'il a adaptée sur des arbres représentant des documents. Il a construit ainsi un système de recherche documentaire qui répond à une requête (document en entrée) par le renvoi d'un ensemble de documents similaires.

Une autre raison justifiant le choix des réseaux de neurones pour notre application est que ces réseaux incrémentaux n'ont à notre connaissance pas encore été utilisés pour les données structurées (graphes, arbres). Le travail présenté dans ce chapitre présente donc un intérêt théorique aussi bien que pratique.

7.2.1 Incremental Growing Neural Gas

L'inconvénient du GNG, résidant dans le choix du paramètre λ, peut vraiment être un handicap si on ne connaît pas du tout l'ensemble des données que le système va traiter. Par exemple, si un ensemble de nouveaux cas très différents parvient au système et que tous ces cas doivent appartenir à des classes (ou neurones) différents, le GNG ne fera aucun rejet, et il classera chaque entrée dans le neurone le plus proche (même si celui ci est vraiment trop éloigné de l'entrée en réalité). De nouveaux neurones ne sont donc pas créés pour représenter ces données trop éloignées

Tant que (un critère d'arrêt n'est pas rencontré) **faire**
prendre une entrée E.
trouver le premier et deuxième neurone les plus proches, n_1 et n_2.
incrémenter l'âge des tous les arcs liés à n_1,
Si (un arc entre n_1 et n_2 existe) **Alors**
$age_{arc} \leftarrow 0$
Sinon
cet arc est créé
Fin Si
$n_1 = \epsilon_b.(n_1 - E)$ et pour chaque neurone voisin $n_m = \epsilon_n.(n_m - E)$
le carré de la distance entre E et n_i est ajouté à une variable locale K. $K+ = ||n_1 - entry||$
tous les arcs dont l'âge est supérieur à a_{max} sont supprimés.
Si (la suppression des arcs produit des neurones isolés) **Alors**
supprimer ces neurons
Fin Si
Pour chaque λ entrées **faire**
trouver le neurone n_{H1} avec la plus grande erreur relative K
trouver le neurone n_{H2} avec la plus grand erreur relative parmi les voisins de n_{H1}.
créer un nouveau neurone entre n_{H1} et n_{H2}. $Neurone_{nouveau} = 0.5.(n_{H1}+n_{H2})$.
diminuer K pour n_1 et n_2.
Fin Pour
diminuer K pour tous les neurones.
Fait

Algorithme 4: GNG [Fri94b]

des neurones du réseau. Ceci peut entraîner des erreurs surtout lors de la recherche du cas le plus proche.

Nous avons finalement choisi d'utiliser le IGNG [PE05], qui est une améliroation du GNG sur plusieurs points. Le IGNG s'inspire très largement du GNG (d'où son nom d'ailleurs) sur les points suivants :

- initialisation avec un ou plusieurs neurones,
- recherche du neurone le plus proche,
- augmentation de l'âge des arcs liés au neurone vainqueur,
- mêmes formules de mise à jour des neurones vainqueurs,
- suppression des neurones isolés.

Il se différencie du GNG sur un point crucial : la création de neurones. Comme présenté précédemment, la création d'un nouveau neurone dans le GNG se fait une fois tous les λ entrées. Dans le IGNG, cette création dépend de l'entrée. En effet, pour chaque nouvelle entrée, on calcule la distance entre cette entrée et tous les neurones. On trouve le neurone le plus proche, et on compare cette distance par rapport à un seuil S. Si la distance entre l'entrée et son neurone le plus proche est supérieure à S, alors l'entrée ne fait pas partie de la classe représentée par

le neurone. Elle est donc mise à côté et un neurone appelé embryon est créé au même endroit que l'entrée. Ceci fait que dans un réseau du type IGNG, à un instant T, si une entrée est trop éloignée de tous les neurones, elle ne sera guère associée à un neurone existant, mais elle créera sa propre classe. Ceci est très intéressant dans la mesure où on ne mélangera pas des classes qui sont très hétérogènes. Chaque ensemble de données homogènes aura son propre représentant.

Une fois qu'un neurone embryon est créé, il peut être sollicité par d'autres entrées. Si c'est le cas, son âge augmente progressivement jusqu'à dépasser un seuil $a_{neurone_{max}}$, paramètre choisi par l'utilisateur et qui indique que le neurone devient mature. Un neurone mature est donc un neurone qui a été créé au début pour représenter une donnée éloignée des autres neurones, mais qui peut par la suite être renforcé pour former un neurone. L'algorithme du IGNG est détaillé en 5.

a_{max}, S : **paramètres d'entrée du réseau** ;
Tant que (une condition d'arrêt n'est pas remplie) **faire**
  choisir une entrée E.
  trouver son neurone le plus proche c_1.
  Si (le réseau est vide ou $d(E, c_1) > S$) **Alors**
    insérer un nouveau neurone embryon : $Neurone_{nouveau} = E$
  Sinon
    Chercher le deuxième neurone le plus proche
    Si (il n'y a qu'un seul neurone dans le réseau ou $d(\epsilon, c_2) > S$) **Alors**
      insérer un nouveau neurone embryon tel que $Neurone_{nouveau} = E$
      créer une nouvel arc entre c_1 et $Neurone_{nouveau}$.
    Sinon
      incrémenter l'âge de tous les arcs provenant de c_1.
      $\omega_{c_1} += \epsilon_b \cdot (Neurone_{nouveau} - c_1)$
      $\omega_{c_n} += \epsilon_n \cdot (Neurone_{nouveau} - c_1)$, (n étant les voisins de c_1)
      Si (c_1 et c_2 sont reliés par un arc) **Alors**
        $age_{c_1 \to c_2} = 0$
      Sinon
        créer un arc entre c_1 et c_2
      Fin Si
      incrémenter l'âge de tous les neurones voisins de c_1
      Pour chaque neurone embryon **faire**
        Si ($age(neurone) > a_{max}$) **Alors**
          le neurone devient un neurone mature
        Fin Si
      Fin Pour
    Fin Si
  Fin Si
Fait

Algorithme 5: IGNG [PE05]

Le IGNG possède un effet mémoire qui est très intéressant. En effet, lors d'une classification incrémentale, non seulement on désire classer toutes les données en cours, mais on veut aussi

qu'un nouveau type de données n'influe pas sur la topologie du réseau déjà obtenue. Si cela arrivait, on ne serait alors plus capable de représenter les données telles qu'elles ont été fournies au début. Cette capacité à s'adapter vient du fait que les neurones ne sont pas créés périodiquement, mais à chaque fois que cela est nécessaire. Par contre, pour le GNG, si une nouvelle topologie apparaît dans l'espace d'entrée, il n'est souvent pas possible que le GNG puisse l'épouser complètement puisqu'il a un manque de neurones (puisque créés périodiquement) et il se peut ainsi que le GNG initial soit dégradé. La figure 7.3 montre la dégradation du GNG lors de l'apparition d'une nouvelle topologie, alors que le IGNG n'est pas dégradé.

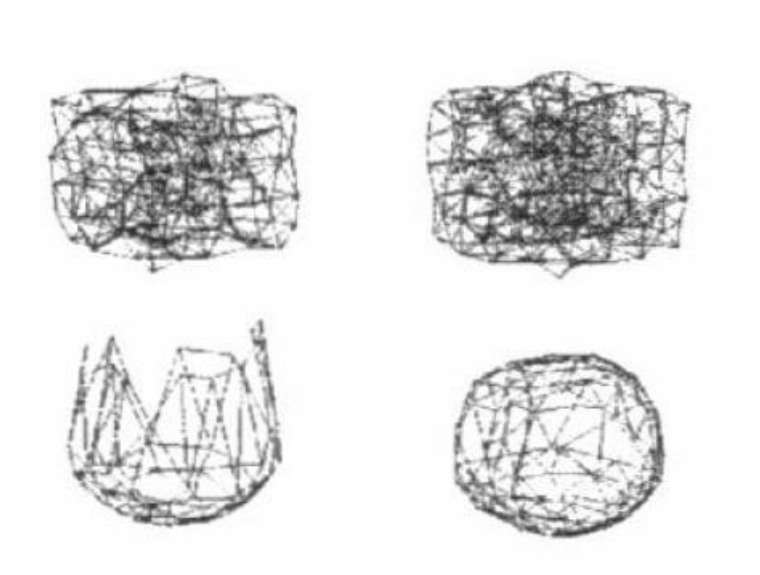


FIG. 7.3 – GNG versus IGNG en classification incrémentale. Source [PE05]

7.2.2 Notre contribution : Improved Incremental Growing Neural Gas I2GNG

Le IGNG, malgré tous ses avantages, a certains inconvénients que nous nous sommes proposés d'améliorer. Tout d'abord, le seuil S au delà duquel un nouveau neurone doit être créé doit être réétudié. En effet, dans l'article présentant le IGNG, les auteurs ont proposé comme seuil S l'écart type de la base. Ceci peut effectivement être appliqué quand on connaît toutes les informations relatives aux données que le IGNG va traiter. Cela n'est cependant pas une bonne solution quand une classification incrémentale est effectuée.

En effet, une classification incrémentale est par définition une classification où les données arrivent une par une. On ne peut donc rien connaître sur les données futures dès le début de la classification. Le système incrémental doit donc réagir, ajuster ses paramètres et prendre en compte les nouvelles données tout en essayant de garder les anciennes en mémoire le plus possible. De plus, comme c'est un seuil global, il se peut qu'il ne soit vraiment plus utilisable au cours de l'apprentissage, il faut donc trouver une manière d'avoir un seuil adaptatif, qui prenne en compte la ou les classes les plus proches de chaque entrée. Nous allons donc essayer de nous inspirer des premières approches non neuronales exposées au début de ce chapitre ainsi que des approches neuronales pour trouver un seuil adéquat.

La figure 7.4 montre deux exemples typiques de problèmes pouvant se produire avec un seuil global :

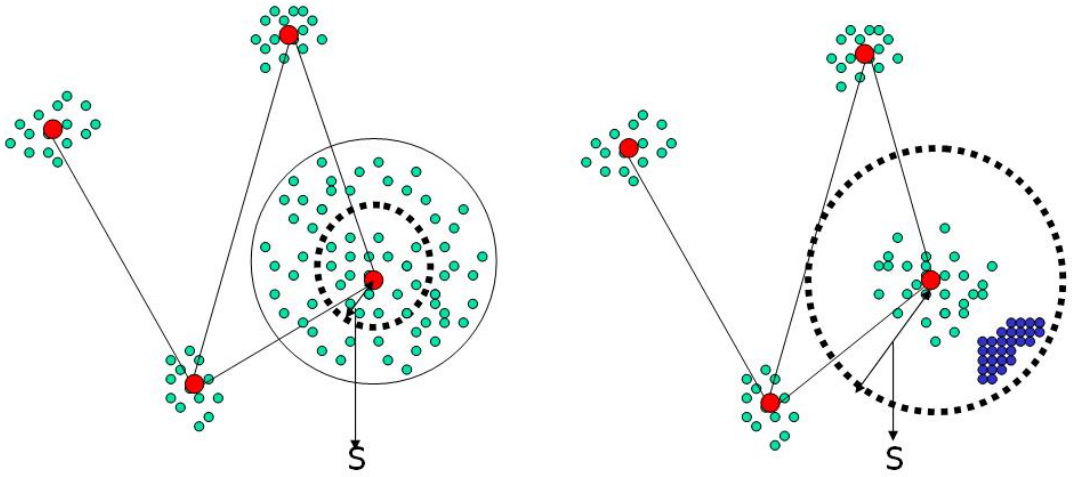


FIG. 7.4 – Problèmes avec le IGNG. A gauche, un seuil trop petit pour représenter toutes les données d'une classe. A droite, un seuil trop grand,

- le premier (figure 7.4 à gauche) se produit quand le seuil S est très petit et qu'une classe existante est homogène et dont la taille dépasse le seuil S. Dans ce cas, plusieurs neurones peuvent être créés pour représenter une même classe qui ne présente pas de variations.
- le deuxième se produit quand le seuil S est très grand. Dans ce cas, si une classe apparaît à l'intérieur de la zone d'influence d'un neurone existant, elle ne sera pas représentée par un neurone qui lui est dédié. C'est le cas de la figure 7.4 à droite, avec les points bleus qui représentent une autre classe de données qui ne peut pas être représentée par un neurone, car ces données se trouvent déjà dans le périmètre d'un autre neurone.

Etude du seuil S

Le seuil S que nous proposons ne peut pas être un seuil global relatif à tous les neurones du IGNG. En utilisant des seuils locaux (donc relatifs à chaque neurone), nous assurons que le IGNG s'adapte vraiment aux données dont il dispose. Un I2GNG qui fonctionne bien est un réseau qui permettrait par exemple pour certaines classes d'ajouter une donnée distante de d_1, alors qu'il ne permettrait pas le rajout d'une autre donnée distante elle aussi de d_1 à une autre classe. Le fait de s'adapter aussi bien aux données présentées qu'aux neurones du IGNG constitue un de nos objectifs.

Nous nous inspirons des travaux de [HK03] et [KIK06] qui proposent d'utiliser des caractéristiques locales à chaque classe pour en créér de nouvelles. Nous proposons donc pour le seuil S d'utiliser aussi des caractéristiques locales à travers la formule suivante :

$$S = m_N + \alpha.\sigma_N$$

, où m_N est la moyenne des distances des données par rapport au neurone, σ_N est l'écart type de ces distances, et α un paramètre à fixer soit par l'utilisateur, soit automatiquement (voir résultats expérimentaux pour vérifier l'effet de α). Ce seuil est évidemment un seuil qui ne dépend que de la classe la plus proche. En prenant en compte la moyenne des distances des données d'une classe par rapport au neurone représentant la classe, nous permettons à toutes les données dont

la distance est inférieure à la moyenne des distances d'appartenir à la classe. L'ajout de $\alpha.\sigma_N$ permet une tolérance vis à vis de la distance par rapport au neurone.

Chaque neurone est donc maintenant virtuellement entouré d'un "cercle d'influence" au delà duquel les données ne sont pas acceptées 7.5. Remarquons ici que ce cercle d'influence bouge au fur et à mesure que les données sont associées à la classe. En effet, comme le neurone est mis à jour à chaque fois qu'une donnée vient se greffer sur une classe, il va de soi que ses coordonnées dans l'espace changent au fur et à mesure. Ceci ne pose aucun problème puisque les autres neurones bougent aussi.

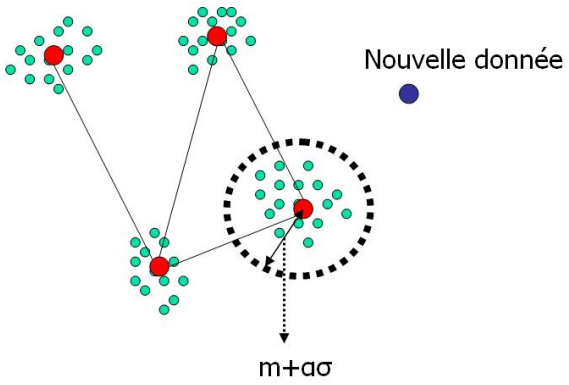


FIG. 7.5 – Zone d'influence d'un neurone

L'ordre d'arrivée des données, en considérant le seuil proposé ici, ne doit pas avoir d'influence sur le résultat final. Nous avons d'ailleurs testé cela en essayant de construire un I2GNG sur deux distributions différentes : l'une circulaire et l'autre triangulaire. Nous avons donné des points de chacune des distributions aléatoirement (une fois dans le cercle, une fois dans le triangle) lors de la première expérience (7.6 à droite) puis donné tous les points du cercle suivis de tous ceux du triangle dans une deuxième expérience (7.6 à gauche). Tout d'abord, nous remarquons que le I2GNG s'est très bien déployé sur les deux distributions que ce soit dans le cas de l'arrivée aléatoire des données ou dans l'autre cas. Ceci conforte le fait que le I2GNG peut vraiment faire un apprentissage incrémental sans que cela nuise aux informations qu'il a déjà apprises.

Nous remarquons par contre que l'espace entre les neurones dans le cas de l'arrivée aléatoire des données est plus grand que dans l'autre cas. Ceci s'explique par le fait que lorsque des données arrivent sur le cercle ou sur le triangle, elles ont deux choix possibles :

- soit elles trouvent un neurone déjà créé, dans quel cas elles s'attachent à ce neurone.
- soit elles sont loin des neurones existants, dans quel cas elles forment de nouveaux neurones embryons. Comme les données arrivent aléatoirement, il y a beaucoup de neurones embryons qui se créent, sans arriver au stade mature (parce que les données sont très dispersées dans l'espace et ne renforcent jamais un seul neurone). Lors de la classification finale, seuls les neurones matures participent à cette classification, les neurones embryons ne sont pas pris en considération, ce qui explique la distance entre les neurones dans la figure 7.6 à droite.

Afin d'optimiser le fonctionnement du I2GNG, il peut être utile, dans le cas où l'utilisateur a déjà une connaissance des données à classer, d'initialiser les premiers neurones manuellement en les choisissant de manière à ce qu'ils soient les plus différents possibles. Ceci éviterait au système de rassembler tous les premiers neurones au même endroit, ce qui peut empêcher le réseau de se déployer correctement par la suite.

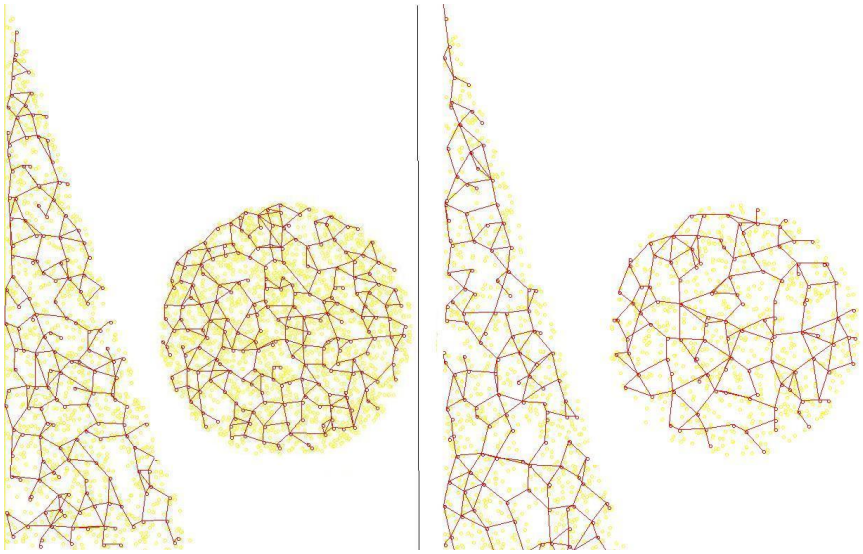


FIG. 7.6 – Formation d'un I2GNG sur deux distributions. A gauche, arrivée des données sur le cercle puis sur le triangle. A droite, arrivée aléatoire des données.

Etude de la condition de suppression d'un neurone

Dans le GNG et le IGNG, un neurone est supprimé s'il n'est plus lié à aucun autre neurone. Ceci veut dire que tous les arcs connectés à ce neurone ont aussi été supprimés parce que leur âge a dépassé le seuil $a_{arc_{max}}$. Or, quand l'âge d'un arc augmente, ceci veut dire qu'un des neurones auxquels il est connecté a été sollicité (une donnée lui a été associée). Nous allons montrer un cas où un neurone doit être supprimé (selon les algorithmes originaux du GNG et IGNG) alors que, selon l'application, il pourrait être conservé.

Dans la figure 7.7, nous avons 3 neurones X, Y et Z. X est très éloigné des neurones Y et Z. Supposons que les neurones Y et Z soient très sollicités à un moment donné (cela veut dire qu'il y a beaucoup de données qui vont être attribuées à Y et/ou Z). Les âges des arcs liant X à Z et X à Y vont augmenter progressivement jusqu'à atteindre le seuil $a_{arc_{max}}$. Une fois cet âge atteint, les deux arcs précédents vont être supprimés, et le neurone X aussi.

Or, le neurone X représente des données qui sont certes très éloignées du reste des données, mais ces données existent bel et bien et il n'est pas toujours judicieux de les représenter par un neurone très éloigné (c'est comme si on noie cette information dans la masse). Dans certaines applications, le fait de garder un représentant pour les données éloignées ou rares peut avoir son importance. C'est le cas de notre système : si une classe de factures est très éloignée des classes actuelles, il est intéressant de la garder.

Nous proposons donc d'ajouter une condition sur la suppression des neurones. Pour chaque neurone susceptible d'être supprimé, nous examinons sa distance avec les neurones les plus proches. Si le neurone est proche (distance inférieure à un seuil S_1), alors il peut être supprimé, sinon, il faut le garder même s'il n'est plus lié à aucun autre neurone. Si on se réferre à la figure 7.7, le seuil S_1 doit être supérieur à $S_2 = m_X + \alpha.\sigma_X + m_Y + \alpha.\sigma_Y$ en considérant que Y est le neurone le plus proche de X. En effet, si la distance entre ces neurones est inférieure à S_2, les deux classes ont probablement une intersection, elles ne sont donc pas très loin l'une de l'autre. Par contre, si la distance est égale à $3 \times S_2$ par exemple, il va sans dire que le neurone à supprimer est très éloigné de son neurone le plus proche. Il faut donc alors le conserver.

Nous proposons donc le seuil $S_1 = \beta.(m_X + \alpha.\sigma_X + m_Y + \alpha.\sigma_Y)$, où $\beta > 1$ peut être choisi

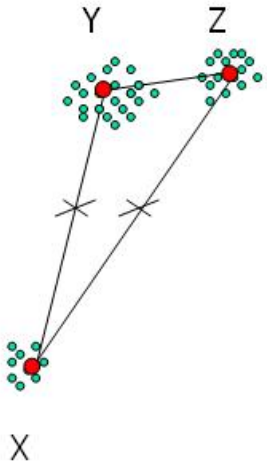


FIG. 7.7 – Cas d'un neurone supprimé dans le GNG et le IGNG, alors qu'il aurait dû être conservé

par l'utilisateur. Ce seuil garantit que les deux classes ne se touchent pas (leurs frontières sont disjointes) et qu'elles sont donc suffisamment distantes pour que les données de l'une ne soient pas affectées à l'autre.

7.2.3 Représentation des graphes dans les réseaux de neurones

Les réseaux présentés précédemment sont utilisés généralement avec des vecteurs. Notre problématique, qui est la classification et l'enrichissement de la base de cas, utilise quant à elle des graphes . Il s'agit alors de savoir comment modéliser les graphes pour pouvoir utiliser le GNG et IGNG. A notre connaissance, ceci n'a pas encore été fait dans la littérature, et même si cela n'est pas une tâche très difficile, il n'empêche qu'il est intéressant de tester des algorithmes de classification incrémentale sur des graphes.

Les réseaux de neurones ont déjà été utilisés avec des entrées en arbres ou en graphes. Dans [Spe01], Sperduti a introduit l'application des réseaux de neurones (perceptron multicouches et cartes de Kohonen) aux données de type structure (graphes, arbres). La première étape à faire consiste tout d'abord à trouver un parcours du graphe. Ce sens de parcours va permettre de modéliser le graphe sous forme de vecteur. Chaque élément dans ce vecteur coïncidera avec un noeud du graphe. De plus, si on choisit la même méthode de parcours pour tous les graphes étudiés, les relations induites par les arcs se trouvent aussi conservées. Il faut aussi pouvoir coder les noeuds des graphes de telle manière à ce qu'ils puissent correspondre à des valeurs numériques dans les vecteurs.

La méthode proposée par Sperduti est en fait une méthode récursive pour le calcul de la SOM (on ne présentera que l'adaptation faite à la SOM). Elle est d'ailleurs appelée R-SOM. Voici les étapes de l'algorithme R-SOM (simplifié) :

1. On choisit un parcours du graphe. Chaque graphe est représenté par un vecteur V.
2. Les neurones sont aussi initialisés aléatoirement.
3. Pour chaque graphe, on commence par le dernier noeud (dans le parcours). Chaque Noeud du graphe est représenté par un vecteur qui prend en compte les fils de ce noeud (les noeuds suivants) dans le graphe. Si un noeud donné N n'a pas assez de fils pour remplir tout le vecteur $V1$, on met alors des valeurs aléatoires, qui ne sont pas présentes dans la SOM.

- on cherche le neurone le plus proche dans la SOM
- on met à jour les coordonnées du neurone en le rapprochant de $V1$
- on garde les coordonnées du neurone vainqueur

4. Pour chaque noeud, le vecteur présenté à la SOM est codé sous la forme suivante :

$$V1 = [NoeudN, W(N1), W(N_2)...., W(N_N)]$$

$W(N_2)$ correspond aux coordonnées (dans la carte) du neurone vainqueur quand le noeud N_2 a été présenté à la carte. Ceci explique le caractère récursif du R-SOM.

5. Chaque neurone est mis à jour en fonction de sa proximité au noeud présenté (comme dans l'algorithme classique de la SOM).

6. Le neurone correspondant à un graphe G donné est celui qui est finalement excité par le noeud origine du graphe (celui qui engendre tous les autres noeuds).

7. A la fin de l'apprentissage, on s'aperçoit que certains neurones dans la carte sont plus sensibles aux noeuds situés au milieu des graphe, alors que d'autres neurones sont plus sensibles aux noeuds origines des graphes.

Méthode de Gunter et Bunke pour l'utilisation de la SOM sur des graphes

Gunter et Bunke [GB02] utilisent les graphes en entrée (sans les encoder en vecteurs) de la SOM. Chaque noeud de la SOM est alors un graphe (avec des attributs sur les noeuds, sur les arcs...) qui peut être mis à jour comme dans l'algorithme d'apprentissage de la SOM. L'originalité de la méthode de Bunke est le fait d'utiliser les opérations d'édition afin de modifier un noeud (le mettre à jour).

Si X est un vecteur, il est facile de modifier un vecteur X d'une valeur α. Ceci n'est pas aussi facile dans le domaine des graphes. En effet, supposons qu'un graphe doit être modifié de α : cela implique que les modifications doivent être apportées aux noeuds et aux arcs du graphe. Or multiplier un arc ou le diviser n'a pas vraiment de sens, de même que pour un noeud. Les seules opérations possibles sont donc l'addition ou la suppression d'information (arcs ou noeuds) à partir ou sur ces graphes.

[GB02] a présenté une idée originale qui consiste à utiliser les principes de la distance d'édition pour modifier un graphe (suppression, insertion, substitution). En effet, un graphe qui doit être modifié de α peut l'être si on arrive à effectuer sur ce graphe un ensemble d'opérations d'édition c_i telles que la somme de leurs coûts vaut α : $\sum(cout(operations)) = \alpha$. Le principe de l'approche consiste donc à appliquer des opérations d'édition au graphe jusqu'à atteindre un coût de transformation égal à α.

Cette approche a été utilisée avec succès pour les cartes de Kohonen, elle a même été appliquée sur la classification de chiffres manuscrits (modélisés en graphes). L'extension de cette méthode au GNG ou IGNG est simple. Les seules opérations où la modification du graphe intervient est lors de la mise à jour des neurones les plus proches. Sur les vecteurs, ces opérations sont $\omega_{c_1} += \epsilon \cdot (\xi - c_1)$. Transposé dans le cas des graphes, la formule devient : $c_1 += \epsilon \cdot d(graphe_{neurone}, c_1)$, où c_1 est ici le graphe en entrée du réseau. Pour modifier c_1 , on utilise donc le principe énoncé précédemment.

Nous avons fait le choix d'utiliser la méthode proposée par Bunke pour la classification de nos graphes de documents car elle est non seulement plus intuitive, plus compréhensible en théorie, mais elle est aussi plus facilement programmable.

7.2.4 Initialisation des neurones

Le problème d'initialisation des neurones dans un réseau de neurones non supervisé est un problème important puisque le résultat de la classification peut en dépendre. En effet, nous avons constaté pendant nos travaux (publiés dans [HBS05]) que deux cartes de Kohonen ayant exactement les mêmes caractéristiques peuvent donner deux résultats différents à cause d'initialisations différentes (une aléatoire et l'autre plus ciblée). Plusieurs techniques d'initialisation des neurones pour le cas des cartes de Kohonen (et qui peuvent être généralisées facilement aux cas des GNG, IGNG et I2GNG) ont été proposées dans la littérature.

- On trouve d'abord la méthode la plus triviale qui consiste à initialiser quelques neurones aléatoirement (à partir des données dont on dispose) et de commencer la classification avec ces neurones. Cette méthode possède le risque d'initialiser les neurones avec des données qui sont très similaires (ce qui peut facilement fausser la classification par la suite). Ceci est particulièrement vrai dans les réseaux de type cartes de Kohonen puisque les neurones sont liés les uns aux autres. Nous pouvons penser que ce risque est moindre dans le cas d'un GNG ou d'un IGNG. En effet, supposons que tous les neurones soient choisis dans la même classe dans l'espace d'entrée. En fonction des données qui vont arriver, ces neurones seront soit mis à jour par des données de la classe qui a servi à l'initialisation, soit non utilisés par ce que d'autres neurones viennent de se créer. Dans le GNG, la création de nouveaux neurones est assurée toutes les λ données et elle est effectuée à l'endroit qui contient le plus d'erreur relative. Donc, même si les neurones initiaux accaparent toute l'information au début, ils finiront par se libérer et bouger progressivement sur d'autres classes. Pour le IGNG, ce problème d'initialisation aléatoire peut même ne pas se poser du tout. En effet, ce réseau peut commencer avec un seul neurone (vu que la création d'un neurone se fait en fonction de la distance de l'entrée par rapport aux neurones les plus proches du réseau). Cependant, et afin d'accélérer la convergence de ces réseaux, il reste préférable de ne pas les initialiser aléatoirement.
- Une autre méthode consiste à initialiser les premiers neurones avec des données extrêmement différentes. Supposons que nous ayons une certaine connaissance des données avec lesquelles on peut commencer la classification. Si on initialise les premiers neurones avec des données très différentes et très éloignées les unes des autres, il devient alors plus probable de couvrir l'espace des données d'une manière plus optimale.
- Bunke propose dans [GB02] de ne pas choisir aléatoirement les neurones du début (pour la carte de Kohonen), mais de les sélectionner et de les perturber. Ceci fait que la classification n'est pas par la suite biaisée par des choix de l'utilisateur.

Pour notre classification incrémentale, le but est de classer des graphes de documents. Nos neurones sont donc représentés en graphe. Nous avons initialisé le IGNG et I2GNG avec des documents choisis aléatoirement dans des classes différentes.

7.2.5 Application dans CBRDA

Une fois que le système d'apprentissage est fonctionnel, nous disposons désormais d'un moyen pour juger qu'un document en entrée du système possède un cas proche dans la base ou pas. En effet, comme la base est représentée par le I2GNG, chaque document est comparé avec les neurones du I2GNG. Si la distance entre le document et le neurone le plus proche est inférieure à $S = m_N + \alpha.\sigma_N$, alors ce document possède un cas similaire dans la base de cas et c'est le premier cycle de RàPC qui sera utilisé. Sinon, on passe directement au deuxième cycle de RàPC.

Cette condition de rejet est très intéressante puisque elle permet d'éviter une intervention de

l'utilisateur à ce niveau du système. De plus, grâce au système incrémental, un document qui est rejeté à un instant donné ne l'est pas plus tard, si un document similaire a été appris entre temps.

7.3 Expérimentations sur la MNIST

Les expérimentations sur la MNIST nous ont permis de vérifier si le I2GNG peut être adapté à la classification de données images. Cette base de chiffres manuscrits a été utilisée par de multiples auteurs et avec de multiples classifieurs. Prudent [PE05], a appliqué le GNG et le IGNG sur la reconnaissance de ces chiffres. Les images n'ont pas été utilisées dans leur état brut, mais une décomposition pyramidale à quatre niveaux a été effectuée sur ces images avant d'obtenir une représentation en vecteurs. Les résultats obtenus sont de l'ordre de 91.44% pour le GNG et 91.71% pour le IGNG. Les bases d'apprentissage et de test ont été utilisées partiellement pour ces tests (2626 exemples pour l'aprentissage et 2619 exemples pour le test).

Nous proposons d'utiliser la base d'apprentissage en entier. Nous avons utilisé les 60000 exemples d'apprentissage progressivement. A chaque expérimentation, nous avons utilisé $n \cdot 10000$ échantillons pour l'apprentissage du I2GNG, n variant de 1 à 6 , avec un seul passage de ces échantillons. Nous avons effectué les tests sur les 10000 exemples de test après chaque apprentissage. Nous avons aussi essayé d'utiliser la distance la plus simple possible entre les images, à savoir la distance euclidienne sur les pixels. Même si cette distance n'est pas très appropriée pour le calcul de distances entre images, nous l'avons utilisé uniquement pour évaluer le I2GNG. Les paramètres de ce réseau sont les suivants : $\alpha = 2, \epsilon_b = 0.01, \epsilon_n = 0.002, a_{arc} = 5, a_{noeud} = 5$. Les résultats sont résumés dans le tableau 7.1. Nous avons refait les mêmes tests mais en passant l'échantillon d'apprentissage deux fois au lieu d'une seule. Nous avons à chaque passage gagné deux points (2) en taux de reconnaissance.

échantillons	reconnaissance
10000	88.45%
20000	91.02%
30000	92.58%
40000	93.66%
50000	94.06%
60000	94.29%

TAB. 7.1 – Résultats obtenus avec la MNIST après un seul passage des exemples d'apprentissage. Nous constatons que plus il y a d'exemples, plus le taux de reconnaissance augmente.

Si on compare ces résultats à ceux de la littérature [4], nous constatons que ce résultat s'approche de celui donné par le classifieur K-plus-proches voisins (Kppv) avec la même distance euclidienne (qui donne 5% d'erreur) mais qu'il est meilleur que celui doné par des classifieurs linéaires (qui donnent au moins 7 % d'erreur). Notons ici que les résultats du classifieur Kppv augmentent nettement pour atteindre 99.48% de succés du moment qu'un ensemble d'opérations de pré-traitement sont effectuées sur les images de la base. Nous pouvons donc nous attendre à une amélioration des résultats du I2GNG si ces mêmes opérations sont effectuées dans notre système.

[4] http ://yann.lecun.com/exdb/mnist/index.html

Les résultats obtenus sont incomparables avec ceux obtenus avec un SVM ou un réseau de neurones à couches de convolution, mais il est à rappeler que les résultats affichés dans 7.1 sont obtenus avec un seul passage des données. Ceux obtenus avec les classifieurs précédemment cités le sont après plusieurs passages et un temps d'apprentissage beaucoup plus long.

L'avantage, cependant, de ces classifieurs est que même s'ils sont très lents en apprentissage, leur vitesse de reconnaissance est beaucoup plus grande que celle du I2GNG (de même que le IGNG et le GNG). En effet, dans le cas d'un perceptron multi-couches classique par exemple, la reconnaissance consiste uniquement à propager les informations de l'image d'entrée dans le réseau. Contrairement à cela, la reconnaissance dans le I2GNG consiste à parcourir tous les neurones afin de trouver le plus proche. Même si le nombre de comparaisons est beaucoup plus réduit qu'un Kppv, une solution doit être trouvée pour accélérer encore plus le processus de reconnaissance.

7.4 Expérimentations sur des cas de documents

7.4.1 Evaluation du I2GNG

Nous avons testé le I2GNG sur des documents administratifs représentés par des graphes. A chaque facture est associé un graphe de mots-clés où les noeuds représentent les mots-clés et les arcs leurs positions relatives les uns avec les autres. Nous n'avons pas essayé le I2GNG sur les graphes de documents tels que nous les avons définis lors de la phase d'élaboration du problème, mais cela ne nécessite qu'une simple adaptation pour passer d'un graphe à l'autre. Nous avons aussi préféré étudier le I2GNG en utilisant la distance d'édition, mais nous aurions aussi pu choisir le sondage de graphes.

Les tests du I2GNG sur les documents ont nécessité une base d'apprentissage et une base de test. Ceci est un processus d'évaluation classique pour les réseaux de neurones supervisés (par exemple), mais cela est aussi possible dans le cas de réseaux de neurones non supervisés. Nous avons utilisé 324 documents en apprentissage répartis en 8 classes de documents. Les documents de test sont au nombre de 169 et sont aussi répartis en 8 classes (les mêmes que celles de l'apprentissage). Nous avons non seulement évalué les résultats du I2GNG, mais aussi testé l'influence des différents paramètres sur la qualité des résultats obtenus. L'incrémentalité du I2GNG est prouvée par le fait que ce réseau commence avec 2 neurones, pour par la suite se retrouver avec un nombre de neurones supérieur à 2. Il est donc inutile de tester le I2GNG à chaque fois qu'il y a création d'un nouveau neurone, puisque les neurones crées sont testés par la suite lors de la phase de reconnaissance.

Le taux de reconnaissance est calculé comme suit. Pour chaque document D de test :

- le neurone le plus proche est recherché parmi tous les neurones du I2GNG. Si ce neurone représente un ensemble de documents de la même classe que le document D, alors ceci est considéré comme un succès. Si par contre, le neurone représente un ensemble de documents d'une autre classe différente de la classe de D, alors c'est un échec.
- si le neurone le plus proche représente un ensemble de documents provenant de plusieurs classes dont la classe de D, alors le document le plus proche est recherché à l'intérieur de ces documents. Si ce document est de la même classe que D, alors c'est un succès, sinon, c'est un échec.

Comme nous pouvons le constater, les différents paramètres ont une influence notable sur les résultats obtenus. Notons d'abord le fait que le nombre de neurones affiché ne correspond pas au nombre de neurones créés réellement. Le nombre de neurones créés est supérieur forcément au

a_{arc}	neurones	rec	α	neurones	rec
10	14	99.40%	0.5	10	98.22%
20	18	97.63%	1	15	98.22%
30	18	97.63%	1.5	12	98.81%
40	16	98.22%	2	14	98.81%
50	16	98.22%	2.5	12	99.40%
60	16	98.22%	3	18	97.63%

TAB. 7.2 – influence de α et de a_{arc}

nombre de neurones final (parfois même très supérieur). Ceci est dû au fait que seuls les neurones matures participent à la classification.

Le nombre de neurones final est aussi supérieur au nombre de classes d'apprentissage. Ceci n'est pas gênant puisque le but de cet apprentissage incrémental n'est pas d'avoir un nombre de neurones égal au nombre de classes, mais d'avoir des neurones qui représentent des données homogènes.

Voici quelques commentaires sur les paramètres testés :

- tout d'abord, nous pouvons penser que le choix d'un α ni très grand ni très petit peut conduire à une bonne classification. Dans le tableau 7.2, le meilleur score de reconnaissance est obtenu avec un alpha de 2.5. Ceci a une explication. En effet, avec un α très petit, la formule $m + \alpha \cdot \sigma$ tend vers m. Ceci peut alors contribuer à la création de beaucoup plus de neurones que nécessaire au cours de l'apprentissage et ces neurones n'auront pas l'occasion de mûrir. Avec un α très grand, nous créons de moins en moins de classes au cours de l'apprentissage, mais ces classes mûrissent très vite car elles n'ont pas de classes concurrentes. En mûrissant vite, le nombre de classes final est assez grand.
- dans le cas d'une distribution gaussienne, 95% de la population est dans l'intervalle $[m - 2 \cdot \sigma\,, m + 2 \cdot \sigma]$, et 99% de la population est dans l'intervalle [m-3$\cdot\sigma$, $m + 3 \cdot \sigma$]. Il n'est donc pas étrange que les meilleurs résultats soient obtenus avec un α qui vaut 2.5. Pour la suite des tests, nous préconisons d'utiliser cette valeur.
- le meilleur taux de reconnaissance a été obtenu avec $a_{arc} = 10$. Le choix d'un âge d'arc élevé (>10) n'est pas un choix très judicieux. En effet, cela implique qu'un neurone n'est supprimé que si lui même ou des neurones voisins ont été sollicités au moins a_{arc} fois. Un tel choix peut handicaper le processus de suppression de neurones et on peut facilement se retrouver avec un nombre de neurones très élevé.

7.4.2 Comparaison du IGNG avec le I2GNG

Afin de vérifier la supériorité du I2GNG sur le IGNG, nous avons testé les deux réseaux sur les mêmes documents, et nous avons comparé le nombre de neurones obtenus à la fin de l'apprentissage. Les résultats montrent que lorsque le seuil S du IGNG (choisi manuellement) correspond aux seuils automatiques des neurones dans le I2GNG (entre 100 et 250), le nombre de classes final est sensiblement le même.

Si ce seuil est très grand, le nombre de classes du IGNG chute et on se retrouve uniquement avec deux neurones matures. Ceci est prévisible puisque avec un seuil S très grand, les neurones initialement créés occupent tout l'espace des données. Aucun autre neurone ne peut être créé. Même s'il y a création d'un neurone, celui-là n'a pas le temps de mûrir.

Le seuil S optimal est obtenu pour une valeur de 250. Il est comparable au seuil S optimal

S	neurones	reconnaissance
100	14	97.63%
150	18	97.63%
200	17	98.81%
250	11	99.40%
350	4	85.20%
1000	2	85.20%

TAB. 7.3 – Nombre de neurones obtenus par le IGNG en fonction du seuil S.

obtenu pour le I2GNG avec un $\alpha = 2.5$.

Les résultats obtenus montrent qu'un seuil local dépendant uniquement du neurone le plus proche est meilleur qu'un seuil global, choisi par l'utilisateur, qui peut ne pas connaître la distribution des données dans l'espace.

En termes de temps de calcul dans le système CBRDA, nous cherchons le cas le plus proche de chaque cas en entrée. Deux étapes sont donc nécessaires : la première consiste à trouver la classe la plus proche (donc le neurone le plus proche) et la deuxième consiste à trouver le cas le plus proche dans cette classe.

Le nombre d'opérations à effectuer avant de trouver le cas le plus proche est donc en $O(N+m)$ où N est le nombre de neurones et m le nombre moyen de cas par neurones.

Si le nombre de classes est très petit (N=2 par exemple) ou par contre, le nombre de classes est très élevé et approche le nombre de données réelles, le gain de temps espéré est minime. Il faut donc un compromis entre ces deux cas extrêmes afin d'avoir une classification qui permette d'avoir un gain de temps considérable. Grâce à un seuil adaptatif, nous pouvons désormais avoir un nombre de classes qui répond à cette contrainte.

7.5 Conclusion

Dans ce chapitre, nous avons exposé la méthode incrémentale que nous avons adoptée pour classer la base de cas. Nous nous plaçons dans le cadre d'une classification incrémentale de graphes de documents. Peu de travaux ayant été effectués dans ce domaine (surtout en RàPC) ; nous avons choisi de travailler sur un algorithme déjà existant et ayant fait ses preuves en classification incrémentale de vecteurs, pour l'adapter et l'améliorer en vue de son application sur des graphes.

Les GNG et IGNG sont des réseaux de neurones incrémentaux qui ont fait leurs preuves dans le domaine de la classification. Ils ont l'avantage de ne pas dépendre d'un nombre final de neurones à respecter et peuvent continuer leur apprentissage au fur et à mesure de l'arrivée des données. C'est pour cette raison que nous avons choisi de les utiliser. L'arrivée de nouveaux cas dans la base de cas est continue et ne doit pas remettre en cause toute la classification, mais juste une partie de la classification (modifications locales et non globales).

Le I2GNG permet donc aujourd'hui à la base de cas de s'enrichir, tout en conservant une classification permettant un accès rapide à la base. Les neurones représentant des cas de documents, ils sont modifiés, déplacés, supprimés en fonction des nouvelles données entrantes dans la base, tout en maintenant en mémoire les cas précédemment appris. Les améliorations apportées au IGNG permettent donc non seulement de garder en mémoire les cas les moins fréquents, mais aussi ceux qui sont très différents des cas les plus fréquents.

La classification que nous avons proposée a été testée sur différents corpus (données synthétiques, base MNIST, documents) et a donné de très bons résultats. Même si les bases d'apprentissage et de test ne sont pas très grandes, nous pensons qu'elles permettent néanmoins d'avoir une idée sur la qualité et la robustesse de l'approche.

8

Conclusion

Dans cette thèse, nous avons abordé le problème de l'analyse de documents administratifs. Nous avons essayé d'identifier les problèmes qui existent dans ce domaine et nous avons essayé de les résoudre en proposant un système intelligent. Ce système commence par modéliser automatiquement les documents à traiter, avant de les analyser et les interpréter en utilisant les connaissances accumulées au cours des traitements précédents.

L'état de l'art nous a permis de faire plusieurs constats :

- peu de travaux traitent aussi bien les tableaux que les autres informations contenues dans un document administratif
- peu de travaux traitent aussi bien les documents de classe connue que les documents complètement nouveaux
- peu de travaux utilisent les connaissances accumulées au cours des traitements de documents précédents
- peu de systèmes qui parviennent à apprendre au fur et à mesure sans remettre en cause ce qui a été appris auparavant

Nous avons proposé un système qui essaie de tenir compte de tous ces éléments.

Lors de la phase de modélisation du document, nous avons proposé une manière originale de représenter les documents administratifs. Nous nous sommes basés sur l'extraction des structures du document, structures que nous avons classées en deux types : structures à motifs et structures à mots-clés. Les structures à motifs correspondent aux zones tabulaires dans les documents. Nous avons proposé pour cela une approche originale qui consiste à étudier la répétition d'un motif dans un document. Nous avons représenté le document par un graphe où les positions absolues des structures sont absentes au profit des positions relatives des structures les unes par rapport aux autres, ceci permettant d'absorber les différentes variations possibles.

Ce modèle de document est comparé à une base de cas de documents, qui sont aussi représentés en graphes. Nous avons proposé pour cela d'utiliser le sondage de graphes pour la recherche du cas le plus proche. Ce choix a été motivé par la vitesse d'exécution de cet algorithme de comparaison de graphes.

Afin de pouvoir proposer une analyse complète de chaque document en entrée de notre système, nous avons proposé de traiter le problème du document structure par structure si le premier cycle n'arrive pas à produire de solution. Cette deuxième étape de raisonnement permet d'éviter l'intervention de l'utilisateur à ce niveau puisqu'elle utilise les structures des documents déjà analysées afin de traiter les structures du document en cours.

Cette approche nous a permis d'avoir de très bons résultats aussi bien pour les documents connus que pour les documents inconnus. Cela peut encore être amélioré, et nous pouvons espérer

qu'avec des bases de cas beaucoup plus riches, nous pouvons atteindre de meilleurs résultats.

Une fois qu'une solution est proposée par le système, il faut alors l'apprendre pour en profiter par la suite. Nous avons proposé pour cela d'utiliser une amélioration d'un réseau de neurones incrémental existant, que nous avons appelé I2GNG. Ce réseau permet de suivre l'évolution de l'arrivée des données en créant et/ou supprimant des neurones existants. Le processus de création/suppression de neurones a été étudié dans cette thèse et nous avons pu l'améliorer par rapport aux réseaux incrémentaux étudiés (GNG, IGNG). Nous continuons aujourd'hui à tester le I2GNG dans un cadre industriel sur plusieurs corpus différents (images, textes...).

Ces différentes contributions ont toutes été implémentées et testées sur une grande variété de documents. Nous avons donc non seulement essayé d'avoir des résultats que nous pouvons qualifier visuellement, mais également qualitativement en les comparant à des documents de vérité fournis par la société. Nous avons comparé nos résultats avec ceux des documents de vérité et pu ainsi mesurer les erreurs du système et celles de l'OCR.

La première perspective de ce travail consiste à le ré-implémenter dans un cadre industriel et de le tester sur un volume de documents beaucoup plus important. En plus des tests, des améliorations du système sont en cours d'étude.

Dans la partie élaboration du problème, nous utiliserons les entêtes de tableaux afin de mieux représenter le problème des structures à motifs et ainsi, le problème du document. En utilisant un problème du document plus fiable, il devient alors beaucoup plus facile d'avoir une interprétation correcte du document.

Nous nous concentrons particulièrement sur la partie apprentissage incrémental afin de rendre l'algorithme I2GNG plus indépendant de ses paramètres initiaux. Nous étudions aussi la possibilité de créer un I2GNG hiérarchique, à l'image du TreeGNG [DAD] et du TreeGCS [HA01]. Cette perspective peut être d'une grande utilité si le nombre de neurones du I2GNG devient très grand. En regroupant les neurones proches dans un même neurone d'un étage de classification supérieur, nous pouvons réduire le temps de recherche du cas le plus proche ou de la classe de l'entrée.

Ces perspectives sont en cours d'étude aujourd'hui. Elles seront implémentées et testées chez ITESOFT.

Bibliographie

[AO97] Hiroyuki Arai and Kazumi Odaka. Form processing based on background region analysis. In *ICDAR*, pages 164–169, 1997.

[AP94] A. Aamodt and E. Plaza. Case-based reasoning : Foundational issues, methodological variations, and system approaches. In *IOS press*, 1994.

[BA05] Stefanie Brüninghaus and Kevin D. Ashley. Reasoning with textual cases. In *ICCBR*, pages 137–151, 2005.

[BB98] Yolande Belaïd and Abdel Belaïd. Form analysis by neural classification of cells. In *Document Analysis Systems*, pages 58–71, 1998.

[BB04] Yolande Belaïd and Abdel Belaïd. Morphological tagging approach in document analysis of invoices. In *ICPR*, pages 469–472, 2004.

[BB07] Hans-Dieter Burkhard and Ralf Berger. Cases in robotic soccer. In *ICCBR*, pages 1–15, 2007.

[BHK95] R. Burke, K. Hammond, and J. Kozlovsky. Knowledge-based information retrieval from semistructured text, 1995.

[BL00] Yungcheol Byun and Yillbyung Lee. Form classification using dp matching. In *SAC '00 : Proceedings of the 2000 ACM symposium on Applied computing*, pages 1–4, New York, NY, USA, 2000. ACM.

[BSY97] Ulrich Bohnacker, Johannes Schacht, and Tulug Yucel. Matching form lines based on a heuristic search. In *ICDAR*, pages 86–, 1997.

[Bun97] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8) :689–694, 1997.

[CB07] Nawei Chen and Dorothea Blostein. A survey of document image classification : problem statement, classifier architecture and performance evaluation. *IJDAR*, 10(1) :1–16, 2007.

[CFGS03] Francesca Cesarini, Enrico Francesconi, Marco Gori, and Giovanni Soda. Analysis and understanding of multi-class invoices. *IJDAR*, 6(2) :102–114, 2003.

[CGMS95] Francesca Cesarini, Marco Gori, Simone Marinai, and Giovanni Soda. A system for data extraction from forms of known class. In *ICDAR*, pages 1136–, 1995.

[CGMS98] Francesca Cesarini, Marco Gori, Simone Marinai, and Giovanni Soda. Informys : A flexible invoice-like form-reader system. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(7) :730–745, 1998.

[CMS02] Francesca Cesarini, Simone Marinai, and Giovanni Soda. Retrieval by layout similarity of documents represented with mxy trees. In *Document Analysis Systems*, pages 353–364, 2002.

[CMSS02] Francesca Cesarini, Simone Marinai, L. Sarti, and Giovanni Soda. Trainable table location in document images. In *ICPR (3)*, pages 236–240, 2002.

[Coü06] Bertrand Coüasnon. Dmos, a generic document recognition method : application to table structure analysis in a general and in a specific way. *IJDAR*, 8(2-3) :111–122, 2006.

[CS05] William Cheetham and Joe Shultz. Using ensembles of binary case-based reasoners. In *ICCBR*, pages 152–162, 2005.

[CWP+04] Colleen Cunningham, Rosina Weber, Jason M. Proctor, Caleb Fowler, and Michael Murphy. Investigating graphs in textual case-based reasoning. In *ECCBR*, pages 573–586, 2004.

[DA02] Pinar Duygulu and Volkan Atalay. A hierarchical representation of form documents for identification and retrieval. *IJDAR*, 5(1) :17–27, 2002.

[DAD] Kevin Doherty, Rod Adams, and Neil Davey. Treegng - hierarchical topological clustering.

[DB07] Sarah Jane Delany and Derek G. Bridge. Catching the drift : Using feature-free case-based reasoning for spam filtering. In Rosina Weber and Michael M. Richter, editors, *ICCBR*, volume 4626 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2007.

[DH98] Fred Hamker Dietmar Heinke. Comparing neural networks : A benchmark on growing neural gas, growing cell structures, and fuzzy artmap. volume 9, pages 1279–1291, 1998.

[DMR01] Michael Dittenbach, Dieter Merkl, and Andreas Rauber. Hierarchical clustering of document archives with the growing hierarchical self-organizing map. In *ICANN*, pages 500–508, 2001.

[DTJ+97] Sébastien Diana, Éric Trupin, F. Jouzel, Yves Lecourtier, and Jacques Labiche. From acquisition to modelisation of a form base to retrieve information. In *ICDAR*, pages 762–765, 1997.

[eSJT06] Ana Costa e Silva, Alípio Mário Jorge, and Luís Torgo. Design of an end-to-end method to extract information from tables. *IJDAR*, 8(2-3) :144–171, 2006.

[FPdB07] Maria Frucci, Petra Perner, and Gabriella Sanniti di Baja. Watershed segmentation via case-based reasoning. In *ICCBR*, pages 419–432, 2007.

[Fri] Bernd Fritzke.

[Fri94a] Bernd Fritzke. Growing cell structures–a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9) :1441–1460, 1994.

[Fri94b] Bernd Fritzke. A growing neural gas network learns topologies. In *NIPS*, pages 625–632, 1994.

[Fri99] Bernd Fritzke. Be busy and unique ... or be history - the utility criterion for removing units in self-organizing networks. In *KI*, pages 207–218, 1999.

[GB02] Simon Günter and Horst Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23(4) :405–417, 2002.

[GC00] Christophe G. Giraud-Carrier. A note on the utility of incremental learning. *AI Commun.*, 13(4) :215–224, 2000.

[GDPP05] Basilios Gatos, Dimitrios Danatsas, Ioannis Pratikakis, and Stavros J. Perantonis. Automatic table detection in document images. In *ICAPR (1)*, pages 609–618, 2005.

[GGC07] Paulo Gomes, Pedro Gandola, and Joel Cordeiro. Helping software engineers reusing uml class diagrams. In *ICCBR*, pages 449–462, 2007.

[HA01] Victoria J. Hodge and Jim Austin. Hierarchical growing cell structures : Treegcs. *Knowledge and Data Engineering*, 13(2) :207–218, 2001.

[HBB07] Hatem Hamza, Yolande Belaïd, and Abdel Belaïd. Case-based reasoning for invoice analysis and recognition. In *ICCBR*, pages 404–418, 2007.

[HBS05] Hatem Hamza, Abdel Belaïd, and Eddie Smigiel. Neural based binarization techniques. In *ICDAR*, pages 317–321, 2005.

[HFGV03] H.Bunke, P. Foggia, C. Guidobaldi, and M. Vento. Graph clustering using the weighted minimum common supergraph. In *GbRPR*, pages 235–246, 2003.

[HK03] Khaled M. Hammouda and Mohamed S. Kamel. Incremental document clustering using cluster similarity histograms. In *Web Intelligence*, pages 597–601, 2003.

[HPG99] Jean-François Hébert, Marc Parizeau, and Nadia Ghazzali. Cursive character detection using incremental learning. In *ICDAR*, pages 808–811, 1999.

[Ish01] Yasuto Ishitani. Model based information extraction and its application to document images. In *DLIA*, 2001.

[Ito93] K. Itonori. Table structure recognition based on textblock arrangement and ruled line position. In *ICDAR*, Japan, 1993.

[Jar98] Jacek Jarmulak. Case-based classification of ultrasonic b-scans : Case-base organisation and case retrieval. In *EWCBR*, pages 100–111, 1998.

[KGKD01] Bertin Klein, Serdar Gökkus, Thomas Kieninger, and Andreas Dengel. Three approaches to "industrial" table spotting. In *ICDAR*, pages 513–517, 2001.

[KHLK98] Samuel Kaski, Timo Honkela, Krista Lagus, and Teuvo Kohonen. Websom - self-organizing maps of document collections. *Neurocomputing*, 21(1-3) :101–117, 1998.

[KHOY01] Taizo Kameshiro, Takashi Hirano, Yasuhiro Okada, and Fumio Yoda. A document retrieval method from handwritten characters based on ocr and character shape information. In *ICDAR*, pages 597–601, 2001.

[KIK06] Sophoin Khy, Yoshiharu Ishikawa, and Hiroyuki Kitagawa. Novelty-based incremental document clustering for on-line documents. In *ICDE Workshops*, page 40, 2006.

[Kol83] J. Kolodner. Maintaining organization in a dynamic long-term memory. In *Cognitive Science*, 1983.

[LD02] Jian Liang and David S. Doermann. Logical labeling of document images using layout graph matching with adaptive learning. In *Document Analysis Systems*, pages 224–235, 2002.

[LdB+03] Jean Lieber, Mathieu d'Aquin, Pierre Bey, Amedeo Napoli, Maria Rios, and Catherine Sauvagnac. Acquisition of adaptation knowledge for breast cancer treatment decision support. In *AIME*, pages 304–313, 2003.

[LL04] Luc Lamontagne and Guy Lapalme. Textual reuse for email response. In *ECCBR*, pages 242–256, 2004.

[LR95] Philippe Lefèvre and François Reynaud. Odil : an sgml description language of the layout structure of documents. In *ICDAR*, pages 480–, 1995.

[LW99] David B. Leake and David C. Wilson. Combining cbr with interactive knowledge acquisition, manipulation and reuse. In *ICCBR*, pages 203–217, 1999.

[LW03] Daniel P. Lopresti and Gordon T. Wilfong. A fast technique for comparing graph representations with applications to performance evaluation. *IJDAR*, 6(4) :219–229, 2003.

[MAM96] Jianchang Mao, Marlon Abayan, and K. Mohiuddin. A model-based form processing sub-system. In *ICPR*, 1996.

[MB98] Bruno T. Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5) :493–504, 1998.

[MH98] B. Messmer and H.Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. In *PAMI*, 1998.

[MLM97] Jianchang Mao, Raymond A. Lorie, and K. M. Mohiuddin. A system for automatically reading iata flight coupons. In *ICDAR*, pages 153–157, 1997.

[MM95] Robert T. Macura and Katarzyna J. Macura. Macrad : Radiology image resource with a case-based retrieval system. In *ICCBR*, pages 43–54, 1995.

[PB04] Petra Perner and Angela Bühring. Case-based object recognition. In *ECCBR*, pages 375–388, 2004.

[PE05] Yann Prudent and Abdel Ennaji. A new learning algorithm for incremental self-organizing maps. In *ESANN*, pages 7–12, 2005.

[Per98a] P. Perner. Using cbr learning for the low-level and high-level unit of a image interpretation system, 1998.

[Per98b] Petra Perner. Different learning strategies in a case-based reasoning system for image interpretation. In *EWCBR*, pages 251–261, 1998.

[Per99] Petra Perner. An architecture for a cbr image segmentation system. In *ICCBR*, pages 525–534, 1999.

[Per00] Petra Perner. Cbr-based ultra sonic image interpretation. In *EWCBR*, pages 479–490, 2000.

[Per01] Petra Perner. Why case-based reasoning is attractive for image interpretation. In *ICCBR*, pages 27–43, 2001.

[PHR06] Petra Perner, Alec Hlot, and Michael Richter. Image processing in case-based reasoning. *The Knowledge Engineering Review*, 20(3) :311–314, 2006.

[PSAE98] P.Héroux, S.Diana, A.Ribert, and E.Trupin. Etude de méthhodes de classification pour l'identification automatique de classes de formulaires. In *CIFED*, 1998.

[PUUH01] Robi Polikar, L. Upda, S. S. Upda, and Vasant Honavar. Learn++ : an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 31(4) :497–508, 2001.

[RSE+98] TRAN VAN THOM R, DIANA S, TRUPIN E, LECOURTIER Y, and LABICHE J. Textual reuse for email response. In *1er Colloque International Francophone sur l'Ecrit et le Document*, 1998.

[SA96] S. Shimotsuji and M. Asano. Form identification based on cell structure. In *ICPR '96 : Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume III-Volume 7276*, page 793, Washington, DC, USA, 1996. IEEE Computer Society.

[SEG06] S.Marinai, E.Marino, and G.Soda. Tree clustering for layout-based document image retrieval. In *DIAL*, France, 2006.

[SFFW02] Hiroshi Sako, Naohiro Furukawa, Masakazu Fujio, and Shigeru Watanabe. Document-form identification using constellation matching of keywords abstracted by character recognition. In *Document Analysis Systems*, pages 261–271, 2002.

[SKP92] Dong Su Seong, Ho Sung Kim, and Kyu Ho Park. Formal definition and entropy calculation of hierarchical attributed random graph. *Pattern Recognition Letters*, 13(8) :545–555, 1992.

[SLBK03] Adam Schenker, Mark Last, Horst Bunke, and Abraham Kandel. Comparison of distance measures for graph-based clustering of documents. In *GbRPR*, pages 202–213, 2003.

[SLBK04] Adam Schenker, Mark Last, Horst Bunke, and Abraham Kandel. Classification of web documents using graph matching. *IJPRAI*, 18(3) :475–496, 2004.

[SMF+03] H. Sako, M.Seki, N. Furukawa, H.Ikeda, and A.Imaizumi. Form reading based on form-type identification and form-data recognition. In *ICDAR*, Scotland, 2003.

[Spe01] Alessandro Sperduti. Neural networks for adaptive processing of structured data. In *ICANN*, pages 5–12, 2001.

[SS04] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation, 2004.

[SSF+03] Hiroshi Sako, Minenobu Seki, Naohiro Furukawa, Hisashi Ikeda, and Atsuhiro Imaizumi. Form reading based on form-type identification and form-data recognition. In *ICDAR*, pages 926–, 2003.

[SSHP97] Jaakko J. Sauvola, Tapio Seppänen, Sami Haapakoski, and Matti Pietikäinen. Adaptive document binarization. In *ICDAR*, pages 147–152, 1997.

[SSRL95] Sargur N. Srihari, Yong-Chul Shin, Vemulapati Ramanaprasad, and Dar-Shyang Lee. Name and address block reader system for tax form processing. In *ICDAR*, pages 5–10, 1995.

[TBB95a] E. Turolla, Yolande Belaïd, and Abdel Belaïd. Form item extraction based on line searching. In *GREC*, pages 69–79, 1995.

[TBB95b] E. Turolla, Yolande Belaïd, and Abdel Belaïd. Line and cell searching in tables or forms. In *ICIAP*, pages 509–514, 1995.

[TL96] A. Ting and M. K. H. Leung. Business form classification using strings. In *ICPR '96 : Proceedings of the 13th International Conference on Pattern Recognition*, page 690, Washington, DC, USA, 1996. IEEE Computer Society.

[TL97] Yuan Yan Tang and Jiming Liu. Information acquisition and storage of forms in document processing. In *ICDAR*, pages 170–174, 1997.

[Ull76] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1) :31–42, 1976.

[VJ99] Aditya Vailaya and Anil K. Jain. Incremental learning for bayesian classification of images. In *ICIP (2)*, pages 585–589, 1999.

[WAB06] Rosina Weber, Kevin Ashley, and Stefanie Bruninghaus. Textual case-based reasoning. *The Knowledge Engineering Review*, 20(3) :255–260, 2006.

[Wal97] Hanno Walischewski. Automatic knowledge acquisition for spatial document interpretation. In *ICDAR*, pages 243–248, 1997.

[Web99] Rosina Weber. Intelligent jurisprudence research : a new concept. In *ICAIL*, pages 164–172, 1999.

[WH97] Toyohide Watanabe and Xiaoou Huang. Automatic acquisition of layout knowledge for understanding business cards. In *ICDAR*, pages 216–220, 1997.

[WM94] Ian Watson and Fahri Marir. Case-based reasoning : A review. volume 9, pages 355–381, 1994.

[WN97] Matthias Wolf and Heinrich Niemann. Form–based localization of the destination address block on complex envelopes. In *ICDAR*, pages 908–, 1997.

[won]

[WT98] Claudia Wenzel and Wolfgang Tersteegen. Precise table recognition by making use of reference tables. In *Document Analysis Systems*, pages 283–294, 1998.

[YLXS91] J. Yuan, L, Xu, and C.Y. Suen. Form items extraction by model matching. In *ICDAR*, France, 1991.

[YTS95] Jianxing Yuan, Yuan Yan Tang, and Ching Y. Suen. Four directional adjacency graphs (fdag) and their application in locating fields in forms. In *ICDAR*, pages 752–755, 1995.

[Zuy97] Konstantin Zuyev. Table image segmentation. In *ICDAR*, pages 705–708, 1997.

Résumé

Cette thèse traite de l'analyse et de la reconnaissance de documents administratifs. L'arrivée continue des documents nous a conduit à choisir une méthodologie prenant en compte les expériences précédentes. Aussi, nous avons opté pour le raisonnement à partir de cas. A partir d'une structuration de base du document représentant ses élements comme les adresses, les zones de montants et les tableaux, un modèle du document est construit sous forme d'un graphe. Il correspond au problème à résoudre.

Ce problème est pensuite comparé à une base de cas de documents en utilisant le sondage de graphes. Si un cas de document similaire existe, alors il est adapté pour analyser et interpréter le cas courant. Sinon, une analyse structure par structure est effectuée en utilisant une base de cas de structures élémentaires de documents. L'arrivée continue des données impose un mode d'apprentissage incrémental, qui peut être fait au fur et à mesure du traitement. Nous avons donc proposé une amélioration d'un réseau de neurone incrémental existant appelé Incremental Growing Neural Gas. L'amélioration proposée consiste à prendre en compte uniquement le voisinage local du neurone le plus proche lors de la phase de création d'un nouveau neurone. Le réseau proposé a été testé avec succès aussi bien sur des documents (factures, formulaires) que sur des données synthétiques.

Cette thèse étant effectuée en collaboration avec l'entreprise ITESOFT, nous avons testé toutes les étapes de notre approche sur des cas réels.

Mots-clés: Raisonnement à partir de cas, analyse de documents, réseaux de neurones incrémentaux.

Abstract

This thesis deals with administrative document analysis and recognition. The continuous arrival of documents lead us to choose a methodology taking into account the previous processing experiences. We chose case-based reasoning for this reason. After extracting the document's structures like adresses, amount zones and tables, a document model is built as a graph, representing the problem to be solved.

This problem is then compared to a document case base using graph probing. If a similar case exists, it is then adapted to analyze and interpret the current case. Otherwise, a structure by structure analysis is done using a document structure case base. The continuous arrival of data requires an incremental learning scheme that could be done as processing goes on. For this purpose, we proposed an improvement of an already existing neural network called Incremental Growing Neural Gas. This improvement consisted in taking into account only the local neighborhood of the nearest neuron while creating a new neuron. The proposed neural network was successfully tested on real documents (invoices, forms) and other synthetic data.

This thesis was done thanks to a collaboration with the company ITESOFT. All the steps of the proposed approach were tested on real cases.

Keywords: Case-based reasoning, document analysis, incremental neural networks.