Analyse de classes de formes pour la transcription de textes imprimés anciens

Sylvain Hocquet*,— Jean Yves Ramel***

**Université François Rabelais de Tours*
Laboratoire d'informatique
64 avenue Jean Portalis
37200 Tours
{sylvain.hocquet, jean-yves.ramel}@univ-tours.fr
***Ecole Nationale d'Ingénieurs du Val de Loire*
Rue de la Chocolaterie - BP 3410
41034 BLOIS CEDEX

RÉSUMÉ. *Ce travail se situe dans le contexte de la numérisation et de l'indexation de documents imprimés anciens. Il décrit un logiciel intitulé Retro, permettant de transcrire semi automatiquement les zones de texte préalablement localisées et extraites à l'aide d'un autre logiciel nommé Agora. Agora réalise simultanément l'analyse de structure des pages et une extraction de toutes les composantes connexes présentes dans chaque page. Une classification non-supervisée de ces composantes connexes est effectuée et conduit à la création de classes regroupant des composantes semblables. Nous proposons dans cet article une étude, puis une exploitation des classes issues d'Agora pour obtenir une transcription du texte (OCR). Nous présentons une analyse statistique et qualitative des classes produites, avant de proposer une méthode de fusion des classes basées sur l'étude de leur relation de voisinage qui nous permet d'étiqueter rapidement 60% des caractères d'un ouvrage sans utiliser de méthode couteuse en temps de calcul.*

ABSTRACT. *This work deals with the digitization and indexing of old printed documents. It describes a software called Retro, which allows to transcript semi-automatically the text boxes previously localized and extracted using another software called Agora. Agora realizes simultaneously the analysis of the pages structure and the extraction of all the connected components present in each page. An unsupervised classification of these connected components is performed and led to the creation of classes of similar connected components. In this paper we propose a study and an utilization, of these classes to transcript text (OCR). We present a statistical and qualitative study of the produced classes, before proposing a method of merging classes based on the study of neighbor relationship allows us to quickly label 60% of characters in a book without using expensive method in computation time.*

MOTS-CLÉS : *OCR, transcription, documents anciens, clustering*

KEYWORDS: *OCR, old printed documents, clustering*

Sylvain Hocquet – Jean Yves Ramel

1. Introduction

Un coup d'accélérateur a été donné aux projets visant à numériser des catalogues d'ouvrages notamment via le projet de Google ou le projet Europeana. Ces projets de numérisation mettent l'accent sur les problématiques d'extraction d'information dans les ouvrages en vue de les archiver, de les indexer ou de faire des recherches sur leur contenu textuel (OCR). Dans ce cadre, nos travaux s'intéressent au cas particulier des ouvrages imprimés anciens. Ces ouvrages utilisent en général une ou plusieurs polices de caractères assez stables permettant l'utilisation de techniques d'OCR, mais devant prendre impérativement en compte des contraintes spécifiques de l'imprimerie du XVI^{e} siècle (transparence des pages, caractères mal imprimés ((Gupta 2007), (Emptoz 2003)). La Figure 1 illustre un certain nombre de ces problématiques.

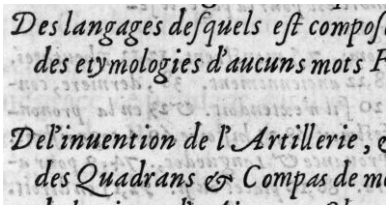

Figure 1: Exemple de document ancien imprimé

Un partenariat entre plusieurs équipes notamment l'équipe Reconnaissance des Formes et Analyses d'Images du laboratoire de Tours et le Centre d'Etudes Supérieures de la Renaissance a permis la création d'un logiciel : AGORA (Ramel 2005), (Ramel 2006) permettant d'extraire la structure physique des pages numérisées. Ce logiciel extrait la structure d'un document en le découpant en zones et associe un label à chacune de ces zones (texte, graphique, bruit, titre). Agora analyse également les formes présentes dans les zones classées comme texte et en s'inspirant des travaux réalisés par Le Bourgeois (Le Bourgeois 2003) regroupe les formes en classes de caractères semblables. Nous avons décidé d'utiliser les résultats issus d'Agora afin mettre en place un module d'OCR original.

L'objectif de ce présent article est de procéder à une étude statistique et qualitative des classes de caractères produites par Agora avant de proposer des stratégies de fusion des classes afin de faciliter l'étape d'OCR. Cet article rappelle tous d'abord brièvement le fonctionnement d'Agora. La partie suivante décrit les résultats de l'analyse statistique que nous avons réalisée sur les classes obtenues. Nous proposons enfin une nouvelle stratégie basée sur l'étude des voisinages existants entre les caractères appartenant à une même classe. Pour conclure, nous présentons les résultats obtenus à l'aide de cette stratégie.

2. Traitement des zones textuelles par Agora

Agora est né de la volonté du CESR "Centre d'Etudes Supérieures de la Renaissance" de Tours (CESR http://www.cesr.univ--tours.fr) de rendre accessible leur catalogue d'ouvrages rares datant des 16eme et 17eme siècles. Ces ouvrages datant du début de l'imprimerie ont une mise en page proche de celle des textes manuscrits. Ces ouvrages peuvent être écrits en latin, grec ou en français de l'époque. L'objectif n'est pas seulement de mettre à disposition les images des ouvrages mais également de mettre à disposition, d'une part, une version texte, et d'autre part, un ensemble de métadonnées d'indexation pour faciliter les recherches et rendre la lecture des ouvrages numérisés plus conviviale. Le rôle d'Agora est de réaliser une segmentation des images et une labellisation de chacune des zones extraites (texte, note de bas de page, lettrine, …) selon les objectifs et règles définis par l'utilisateur. Agora ne réalise pas la reconnaissance du texte, un logiciel complémentaire nommé Retro a été produit pour effectuer cette transcription.

2.1. Détection des caractères et clustering

Agora (Ramel 2005) réalise une détection des composantes connexes dans l'image binarisée et les regroupe en blocs "logiques". Les composantes connexes correspondant à des caractères sont regroupées pour constituer des blocs de texte. Afin de préparer l'étape d'OCR, Agora réalise une classification non supervisée (clustering) des composantes connexes afin de les regrouper en un nombre (limité) de classes. Une composante connexe est définie par un rectangle englobant la forme obtenue après binarisation de l'image avec l'algorithme Niblack (Niblack 1996).

Agora utilise un algorithme basique pour réaliser cette classification. La distance entre deux formes est calculée en comptant le nombre de pixels différents entre les deux formes après avoirs aligné leurs centre des gravités. Ces formes seront ensuite regroupées en classe, L'algorithme de clustering utilisé pour produire les classes est le suivant :

- Pour chaque forme (composante connexe) rencontrée :
    a. Comparer la forme au représentant de chaque classe existante
    b. Si on trouve une classe suffisamment similaire, on ajoute la nouvelle forme dans la classe déjà existante
    c. Sinon on crée une nouvelle classe avec la nouvelle forme comme représentant

Agora affecte ensuite un label aux caractères en fonction de leurs positions par rapport à sa position relativement aux caractères pouvant appartenir à la même ligne de texte. Plusieurs types ont été définis : Caractère Normal, Caractère

Problématique, Accent, Ponctuation. Les caractères accents sont ensuite fusionnés avec le caractère normal le plus proche.

2.2. Analyse statistique et qualitative du clustering

Avant d'envisager une étape d'OCR à partir des données générées par Agora, nous avons procédé à l'analyse critique des résultats de classification produit. Pour ce faire, nous sommes parti d'un ouvrage datant de 1557, que nous avons traité à l'aide d'Agora. Nous avons exporté uniquement les zones classées comme texte. Cet ouvrage comporte 119 pages. Agora a détecté 3260 zones de texte. Ce grand nombre de zones aurait pu être réduit avec l'utilisation de scénarii plus poussés et mieux adaptés à l'ouvrage, mais notre objectif n'était pas d'obtenir une bonne segmentation en blocs.

Agora a extrait 125 744 formes regroupées dans 29 943 classes, un étiquetage manuel de chaque classe est donc impossible. Néanmoins, un grand nombre de classes ne comporte qu'une forme. Nous allons donc nous intéresser à la répartition des formes dans les classes. Un premier examen nous indique que 79% des classes ne contiennent qu'une forme alors que la classe la plus volumineuse contient 3% du total des formes. La Figure 2 montre le pourcentage cumulé de formes dans les classes.

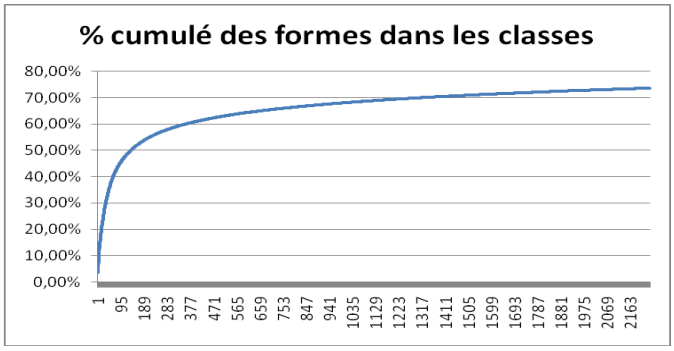

Figure 2 : Répartition des formes dans les classes

Une autre interprétation de la Figure 2 est le pourcentage du texte transcrit en fonction du nombre de classes étiquetées. Au début de la courbe, nous avons une très forte progression du pourcentage du texte étiqueté puis cela ralentit de plus en plus. Ainsi, en étiquetant seulement les 25 classes les plus volumineuses, on étiquette 25% du texte. Mais pour étiqueter 75% du texte, il faut étiqueter 4 000 classes. Il semble possible d'étiqueter manuellement et rapidement 25% ou même 50% du texte. Par contre, il est difficile et très couteux en temps de demander à un opérateur de faire plus manuellement. Les sections suivantes proposent donc des approches automatiques ou semi-automatiques simplifiant la transcription.

2.3 Erreurs classiques du clustering

L'étude des classes produites par Agora nous a permis de détecter un certain nombre de types d'erreurs classiques dues au fonctionnement d'Agora, erreurs pour lesquelles nous proposons des solutions dans la suite de cet article. Ces erreurs sont, pour la plupart dues à la mesure de distance utilisées.

Le premier type d'erreurs correspond à la création de plusieurs classes pour une même forme, Cette création de classes en surnombre a plusieurs explications : des tailles différentes pour un même caractère, mauvais centrage, caractère abimé…

D'autres erreurs apparaissent quand il y a des superpositions de formes dans les rectangles englobant les formes. L'exemple le plus courant concerne la lettre ***f***. La conséquence de cette erreur sera la création d'au moins une classe par type de voisin de la lettre ***f*** **(**Figure 3**).**

Figure 3 : Trois f avec un voisin différent donnant lieu à la création de 3 classes

D'autres erreurs plus gênantes mais heureusement moins fréquentes correspondent à la mauvaise classification de certaines formes dans des classes incorrectes. Cette mauvaise classification est causée par des ressemblances entre formes. Par exemple : les caractères ***h*** et ***b*** ou ***r*** et ***t*** sont fréquemment confondues. Ces erreurs sont concentrées sur un nombre restreint de classes mais avec des taux de mal classés dans ces classes de l'ordre de 10%. On peut donc espérer qu'elles n'auront pas une trop grande influence sur l'étape d'OCR qui suivra. Globalement, les erreurs de mauvaise classification peuvent être estimées à environ 1,2% de formes affectées à des mauvaises classes

Un autre type d'erreurs provient d'une mauvaise extraction des composantes connexes. On observe un certain nombre de caractères coupés en deux formes ou bien non séparé (Figure 4).

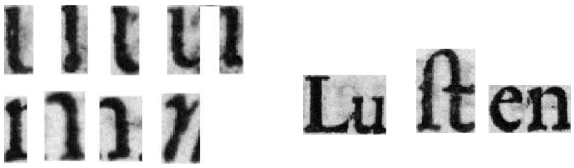

Figure 4 : Exemples de caractères mal isolés

L'étape de fusion des accents génère également un nombre significatif d'erreurs causées par une fausse classification d'un caractère comme accent (Figure 5).

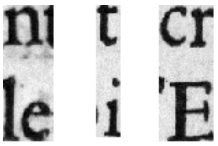

Figure 5 : Mauvaises fusions d'un caractère et de son accent

2.4 Conclusion sur le clustering

En conclusion, les problèmes devant être résolus après l'étape du clustering des formes sont les suivantes :

- Un très grand nombre de classes de formes : Ce grand nombre de classes interdit un étiquetage purement manuel et impose des traitements automatiques ou semi-automatiques d'optimisation.
- L'existence de caractères coupés ou, au contraire fusionnés, doit être prise en compte : Ces caractères entrainent l'apparition de classes qui ne correspondront pas à des caractères usuels et posent donc problème si on souhaite utiliser des dictionnaires de mots ou de trigrammes afin de faciliter l'étape d'OCR.
- Une mauvaise détection de la ponctuation et des caractères accentués :. Ces regroupements font apparaitre des caractères cassés qui pourront avoir une influence sur la segmentation des lignes et des mots du texte.

Malgré ces problèmes Agora fournit pour chaque forme des informations intéressantes comme sa classe (ponctuation, caractère normal, accent….), son voisinage (formes précédentes, suivantes), et surtout un ensemble d'occurrences de forme semblables dispersé au sein de l'ouvrage… Toutes ces informations pourront être utilisées pour simplifier l'étape de transcription.

3. Propositions d'optimisation pour la phase de transcription

L'objectif de nos travaux est de proposer une méthode de transcription qui s'appuie au minimum sur des caractéristiques issues des images et le plus possible sur des informations contextuelle via les classes de formes issues du clustering des formes. Ce choix a été fait suite à l'utilisation d'Agora au CESR de Tours et à la demande des usagers du centre. Nous avons également décidé dans un premier temps de ne pas alourdir le logiciel Agora (déjà conséquent), mais de proposer des solutions pour remédier à certains problèmes que nous avons énuméré en proposant un deuxième logiciel dédié à l'étape de transcription. Ce logiciel (Retro) travaille à partir des données associées aux ouvrages par Agora.

Retro propose une démarche de transcription semi-automatique. Nous nous autorisons donc à faire appel à l'utilisateur bien avant la phase plus contextuelle de post traitement (correction des erreurs). La transcription s'effectue en trois phases :

1. Etiquetage manuel de quelques classes (les plus volumineuses)
2. Fusion des caractères coupés
3. Fusion de certaines classes sur des critères contextuels (étude des voisinages entre formes)

Nos propositions sont basées sur une exploitation poussée des voisinages entre les éléments des classes : une forme d'une classe possède à priori deux voisins : un à gauche et un à droite. Ces voisins peuvent être de plusieurs types :

1. Une forme d'une autre classe de type "Caractère Normal"
2. Un début ou une fin de ligne
3. Un espace
4. Une forme d'une autre classe de type "Ponctuation".

Pour traiter les cas 2, 3 et 4, nous avons choisi de créer une classe fictive SEPARATION, qui indique que la forme est au début ou à la fin d'un mot. Toutes les autres classes étant définies par un identifiant numérique unique (un numéro de cluster).

3.1 Etiquetage manuel

Avant d'effectuer les traitements exploitant ces informations contextuelles, il est nécessaire de faire étiqueter par l'utilisateur les quelques classes les plus fréquentes. L'objectif est d'amorcer le processus en injectant suffisamment d'information au système pour faciliter les traitements automatiques. Les traitements que nous proposons sont basés sur l'étude du voisinage des formes en se limitant aux m classes les plus fréquentes, l'étiquetage par l'utilisateur peut faire gagner en fiabilité et en robustesse. Pour l'instant, il se limite à l'étiquetage des m=25 classes les plus volumineuses, ce qui représente 25% du total des formes dans l'ouvrage choisi pour nos tests.

Pour réaliser l'étiquetage de ces classes, l'utilisateur affiche à l'aide de Rétro un échantillon des formes appartenant à la classe. Il peut en visualiser d'autres et lorsqu'il estime avoir suffisamment d'exemples, indique le label (caractère) correspondant aux éléments présents dans la classe. Lors de nos tests, nous avons constaté que cette étape prend environ 5 minutes à l'utilisateur.

3.2 Détection des caractères coupés

Le traitement suivant est la fusion des classes de caractères qui ont été coupés lors de l'étape de détection des composantes connexes. Des méthodes "basées images" existent, décrites par exemple dans (Droettboom 2003) ou (Allier 2006).

Nous proposons une méthode uniquement basés sur l'étude des voisinages entres formes similaires.

Les hypothèses sur lesquelles nous nous appuyons pour proposer une solution à ce problème sont :

- Les caractères coupées sont généralement toujours les mêmes (***n***,***u***).
- Les morceaux de ces caractères se retrouvent dans les mêmes classes (création de classes de morceaux de ***u*** par exemple).
- Une forme appartenant à une classe d'un morceau de caractère aura comme voisin une forme appartenant à la classe contenant les morceaux du caractère correspondant.

Nous définissons les voisinages potentiels (possibles) d'une classe par deux listes de classes voisines, une pour les voisins de gauche et une pour les voisins de droite. Dans chaque liste, nous stockons la fréquence d'apparition des autres classes dans le voisinage considéré. Par exemple, le **tableau 1** indique la liste des voisins de gauche et fréquences associé pour la classe 39 représentant la lettre *e*.

Tableau 1 : Voisins gauche de la classe 39

classe	SEPARATION	240	1023	…
fréquence	0,65	0,01	0,02	….

Afin d'avoir des informations suffisamment fiables, nous ne pouvons pas travailler sur les classes avec trop peu de formes, c'est pourquoi nous nous limitons aux classes comportant plus de 25 formes (434 classes représentant 62% des formes dans le livre choisi). Notons également que le voisinage que nous proposons est très dépendant de la bonne qualité de la détection des espaces entre caractères effectué par Agora. L'algorithme que nous proposons pour détecter les caractères coupés est le suivant :

1. Calculer le voisinage de chaque classe et trier les listes par fréquences décroissantes
2. Pour une classe *i* :

   Identifier la classe voisine droite la plus fréquente *j*

   Si (*j* n'est pas SEPARATION) alors

   a) Tester si *i* est la classe voisine gauche la plus fréquente de *j*

   b) Tester si les deux fréquences sont supérieures à un seuil α

   Si (a et b sont vrais) alors

   Fusionner les formes de *i* avec leurs voisins de droite qui appartient à la classe *j*

   Fin Si

   Fin Si

3.3 Fusion de classes par étude du voisinage

L'objectif de cette étape est de réduire au maximum le nombre de classes à étiqueter manuellement. Nous proposons donc une méthode afin détecter et fusionner les classes de formes représentant un même caractère. L'hypothèse sur laquelle nous basons notre approche est que deux classes représentant le même caractère doivent avoir des voisinages similaires.

Comme il y a un grand nombre de classes presque vides, nous avons choisi de limiter cette étude aux *n* classes les plus volumineuses. De plus, nous avons choisi de ne pas autoriser la fusion des classes trop petites c'est à dire d'une taille inférieure à *t*. Pour ce faire, nous proposons une mesure simple de distance entre les deux voisinages (de la classe *i* de la classe *j)* contenant chacun les fréquences des *n* classes choisies pour le calcul du voisinage (f_{ik} est la fréquence de la classe *k* dans le voisinage de la classe *i*). Nous avons choisi d'utiliser comme distance, après avoir effectué quelques tests, de prendre la somme des valeurs absolues des écarts entre fréquences observées dans les deux voisinages.

$$dissimilarité(i,j) = \sum_{k=1}^{n} \left| f_{ik} - f_{jk} \right|$$

L'algorithme retenu pour cette fusion est le suivant :

1. Pour une classe *i* chercher la classe *j* avec *taille(j)>t* la plus proche en termes de similarité entre voisinages.
2. Si cette similarité est inférieure à un seuil β alors fusionner les classes i et j

Cet algorithme est contrôlé par trois paramètres. Le premier est le seuil de dissimilarité β qui permet de décider de la fusion, le second le nombre de classes utilisées pour le calcul de la distance et le troisième est la taille *t* minimale des classes à fusionner.

4. Résultats et discussions

4.1 Détection de caractères coupés

Les premiers résultats auxquels nous nous sommes intéressés concernent l'algorithme de détection contextuelle de caractères coupés.

Nous avons réalisé un test avec une fréquence seuil de $\alpha=15\%$. Cela peut sembler faible mais des tests ont montré que c'était la fréquence avec laquelle nous obtenions les meilleurs résultats. Cette basse fréquence est due à l'éparpillement des moreaux d'un même caractère dans plusieurs classes.

Dans cette configuration, nous sommes parvenus à obtenir une bonne détection des caractères coupés. Parmi l'ensemble des 400 classes de plus de 25 caractères, nous avons détecté 10 (5x2) classes correspondant effectivement à des morceaux de caractères coupés. Le problème est qu'apparaissent également des fausses détections

(au nombre de 10). Ces fausses détections sont observées sur des classes contenant des caractères superposés comme ***fa, de,…,*** mais aussi sur des couples de lettres qui se suivent fréquemment en français comme ***qu***, ***là,***...

Ces résultats démontrent qu'il n'y a pas de différence statistique significative entre le voisinage d'un caractère coupé et celui d'un couple de lettres fréquent. Il faut donc coupler cette technique contextuelle avec d'autres critères (par exemple basées image) pour détecter les caractères coupés de façon satisfaisante.

4.2 Fusion des classes

Le second test que nous avons mené a concerné la fusion des classes similaires. Après plusieurs tests nous nous sommes limités à $n=10$ classes pour le calcul de la similarité entre voisinage, et à t=25 pour la taille minimale des classes à étudier.Après analyse de ces n classes et réalisation des fusions possibles. 50% des formes (caractères) se retrouvent dans les 50 classes les plus volumineuses ce qui peut permettre un étiquetage manuel de 50 % du texte assez rapidement. Pour passer à 60% du texte il faudrait étiqueter 150 classes.

Un examen qualitatif des résultats ainsi obtenus démontre à une bonne qualité des fusions de classes. On observe néanmoins quelques erreurs dues à quelques caractères peu fréquents en français comme les ***y.*** Ce traitement peut également accentuer les confusions déjà existantes lors de la création des classes comme des mélanges de ***b*** et de ***h*** dans certaines classes. De plus, les caractères ***fi***, ***fo***, ***fe*** sont parfois mélangés dans une même classe après traitement. Nous ne disposons pas de vérité terrain pour les ouvrages étudiés. Pour effectuer une évaluation plus quantitative de nos résultats, nous avons étiqueté manuellement les classes les plus volumineuses représentant 50 % du texte. Puis, nous avons tiré aléatoirement 5000 formes dans ces classes étiquetées. Enfin, nous avons compté le nombre de formes mal étiquetées parmi ces 5000 formes.

Tableau 2 : taux d'erreurs en fonction du seuil de dissimilarité

Seuil β	0 (pas de fusion)	0,1	0,2
Nombre de classe pour étiqueter 50% du texte	136	83	49
Nombre de classe pour étiqueter 60% du texte	359	237	155
Taux d'erreurs à 50%	1,2%	2,32%	6,5%

Nous avons réalisé plusieurs tests en faisant varier le seuil β (les résultats sont présentés dans le Tableau 2). Nous obtenons un taux d'erreurs de 6,5% pour un seuil de 0,2 ce qui reste important. Les résultats avec un seuil de 0.1 sont meilleurs mais ce seuil limite le nombre de fusions, Il faut donc trouver un compromis entre

réduction du nombre de classes et taux d'erreurs. En analysant les résultats, nous avons constaté que plus de 30% des erreurs provenaient de formes associées incorrectement au caractère **&**. Ce caractère a un voisinage particulier puisqu'il apparait entouré d'espaces. Cela montre encore une fois, l'influence d'une mauvaise détection des espaces entre mots faussant les statistiques sur les voisinages. Le reste des erreurs apparait concentré sur quelques confusions h et b, p et G et p et l … Certaines confusions existaient déjà dans Agora (h et b par exemple) d'autres ont été introduites par notre proposition principalement à cause de la mauvaise détection des espaces entre mots. Une amélioration de l'algorithme de segmentation actuellement utilisé par Agora pourrait résoudre ce problème. Enfin, il nous semble également important de mentionner que les propositions faites dans cet article n'exploitent pas, comme la plupart des OCR, de dictionnaire de mots. Notons que l'analyse d'un corpus de texte datant de la Renaissance nous a permis de produire un dictionnaire des trigrammes fournissant un taux d'apparition (fréquence) pour toutes les combinaisons de trois caractères successifs. Malheureusement, les tests réalisés montrent que la qualité approximative des clusters produits entraine des biais (plusieurs classes pour un même, caractère, caractères regroupés, caractères coupés) trop forts pour exploiter efficacement ce type d'information. Nous ne pouvons donc pas utiliser cet outil pour le moment. Il existe pourtant des méthodes de vérification de l'orthographe performantes comme par exemple dans (Takahashi 1990) qui pourraient permettre de corriger certaines erreurs restant sans solution actuellement.

5. Conclusion

En conclusion, nous avons mené une étude de faisabilité et proposé des méthodes exploitant des analyses statistiques des contextes environnants chaque caractère pour optimiser la transcription de documents anciens. Nos propositions utilisent un mécanisme de classification non-supervisée de formes (composantes connexes) pour mener ensuite une étude des voisinages. Notre objectif était de réduire au maximum le nombre de classes de caractères à étiqueter et ce, sans utiliser d'information issue des images de caractères, ni de dictionnaires de mots. Nos tests ont montré qu'il était possible de réduire le nombre de classes et, avec l'intervention d'un utilisateur, d'arriver rapidement (15 minutes) à 50% du texte étiqueté avec un taux d'erreurs de 6,5%. Ce travail préparatoire devrait permettre de simplifier et d'améliorer le fonctionnement des moteurs d' OCR exploitant cette fois ci à la fois des informations images, des méthodes classiques de reconnaissances des formes comme dans (Vlontzos 1992), des dictionnaires comme proposé dans (Koerich 2004), en complément de tous ce que nous proposons dans cette article . L'autre point important d'amélioration concerne la détection plus fine des espaces. Cette détection influence, en effet, énormément la qualité des informations contextuelles. Il faudra aussi parfaire la détection des accents et de la ponctuation qui conduit à beaucoup trop d'erreurs et perturbe significativement l'analyse statistique des voisinages.

Référence

Allier B., Bali N., Emptoz H., «Automatic accurate broken character restoration for patrimonial documents», *International Journal on Document Analysis and Recognition,* 8:246–261, 2006

Droettboom M., « Correcting broken characters in the recognition of historical printed documents », *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries,*364–366, 2003

Emptoz H., Lebourgeois F., Eglin V., Leydier Y., « La reconnaissance dans les images numérisées : OCR et transcription, reconnaissances des structures fonctionnelles et des métadonnées», *Journées d'étude Numérisation des textes et des images*, .105-129, 2003

Gupta M. R., Jacobson N. P., Garcia E. K., «OCR binarization and image pre-processing for searching historical documents», *Pattern Recognition*, 40(2): 389-397, 2007

Koerich A. L., Sabourin R., Suen C. Y. «Lexicon-driven HMM decoding for large vocabulary handwriting recognition with multiple character models» *International Journal on Document Analysis and Recognition 6(2):*2004

Le Bourgeois F. ; Emptoz H. ; Trinh E., «Compression et accessibilité aux images de documents numérisés : Application au projet DEBORA*», Document numérique,* 7(3-4) : 103-125, 2003

Niblack W., « An introduction to digital image processing»*, Hemel Hempstead, UK, Prentice Hall Ldt*, 1996

Ramel J.Y, Busson S., Demonet M.L. « Agora : the interactive document image analysis tool of the bvh project. » *Conference on Document Image Analysis for Libraries (DIAL)*, 145–155, 2006.

Ramel J. Y. ,Leriche S « Segmentation et analyse interactives de documents anciens imprimés » *Traitement du Signal* (TS), 22(3):209-222, 2005

Takahashi H., Itoh N., Amano T., Yamashita A. « A spelling correction method and its application to an OCR system» *Pattern Recognition*, 23(3-4): 363-377, 1990

Vlontzos, J.A. Kung, S.Y. .,« Hidden Markov models for character recognition» *IEEE Transactions on Image Processing,*1(4): 539-543, 1992