La numérisation de documents : principes et évaluation des performances

Abdel Belaïd, Laurent Pierron, Laurent Najman, Dominique Reyren

La numérisation de documents :
Principe et évaluation des performances

A. Belaïd[1], L. Pierron[1], L. Najman[2] et D. Reyren[3]
[1]LORIA-INRIA, Campus scientifique, B.P. 239, 54506 Vandœuvre-Lès-Nancy
[2]Océ Industries SA, 1 rue Jean Lemoine, 94015 Créteil
[3]Berger-Levrault GTI SA, 18 rue des Glacis, 54017 Nancy

Résumé

Le but de cet article est d'initier à l'élaboration d'une plate-forme industrielle de numérisation de documents papier. Par plate-forme industrielle, nous entendons une architecture logicielle basée sur l'intégration générique des outils de reconnaissance de caractères du commerce (OCR). En effet, l'évolution aujourd'hui de ces outils entrouvre une alternative sérieuse à la saisie manuelle dans un objectif de gain de productivité. Nous présenterons les capacités de ces outils et leurs limites. Nous nous attarderons sur les techniques d'évaluation automatiques de ces outils dans un triple objectif: (1) adaptation optimum des paramètres, (2) sélection du meilleur outil disponible à chaque instant à cause de l'évolution rapide du marché, (3) tirer un meilleur parti de ces outils en combinaison avec une double correction humaine. Enfin, nous ouvrirons l'architecture vers le futur en effleurant les techniques de combinaison pour augmenter la performance globale du système.

1 Introduction

1.1 Position du problème

La numérisation de documents est une étape importante dans la mise en place d'un système de gestion électronique de documents (GED). Le choix de la solution de numérisation doit prendre en compte toutes les étapes de traitement des documents depuis l'acquisition, la conversion du contenu jusqu'à la correction et la mise en exploitation du document final. Le but de cette numérisation est une utilisation du contenu converti, par exemple pour effectuer une recherche d'information.

La partie centrale de la numérisation concerne la reconnaissance de caractères et la structuration du contenu. Elle peut s'effectuer de deux manières, l'une n'étant pas exclusive de l'autre :

- Il est aujourd'hui possible de faire réaliser cette étape manuellement par des opérateurs, à un coût réduit. En effet, on trouve de plus en plus dans les pays en voie de développement une main d'œuvre qualifiée pouvant parfaitement effectuer ce travail. Cela peut cependant être difficile à gérer, à la fois pour des raisons de distances, et parce qu'il faut des personnes sachant lire le français. Il ne faut pas négliger non plus les problèmes de confidentialité qui peuvent se poser.
- On peut le faire automatiquement avec des outils. C'est cet aspect que nous allons développer. Notons cependant qu'on ne pourra jamais retirer définitivement l'utilisateur, car il faudra toujours une étape de validation et de vérification du résultat fourni. Les outils (pas plus que l'être humain) ne seront jamais parfaits, et on tolère moins d'erreur de la part d'une machine que de la part d'un être humain. Par contre, on peut espérer un gain de productivité non négligeable par rapport à une saisie purement manuelle : en effet, les ordinateurs peuvent travailler 24h/24, et être achetés en nombre suffisant. Cette combinaison opérateur/machine ne peut donc être que bénéfique, d'autant plus que les erreurs effectuées par une machine (donc sans interprétation du contenu) ne seront pas du même type que celles effectuées par un opérateur.

Les techniques de lecture automatique de documents ont beaucoup évolué et mûri cette dernière décennie et l'on voit fleurir sur le marché des progiciels de moins en moins chers, de plus en plus complets et fiables offrant des solutions de rétroconversion très fidèles. Ces progiciels, appartenant à la famille des OCR (Optical Character Recognition), sont aujourd'hui capables de distinguer les différents médias dans le document (texte, graphique et photographie), d'identifier les structures linéaires et tabulaires, de faire face à une variation importante de la typographie, d'interpréter et de restituer plusieurs styles éditoriaux. Des bancs d'essais sont effectués couramment sur les dernières versions de ces progiciels montrant les nouvelles possibilités offertes et donnant une évaluation correcte et précise de leur capacité de reconnaissance en termes de taux de confiance, de précision et de vitesse d'exécution, par type de document et de typographie utilisée [1-4].

Ces outils ont également progressé sur le plan de l'intégration puisqu'ils existent aujourd'hui sous la forme d'API facilement intégrables dans un système ou sous la forme de kits de développement offrant des possibilités d'extension et de coopération avec d'autres API permettant l'augmentation de l'efficacité de l'ensemble.

Si ces techniques sont aujourd'hui utilisées sans trop de difficulté dans plusieurs domaines, leur intégration dans une chaîne de numérisation réelle, en milieu industriel,

nécessite de prendre en compte d'autres contraintes liées à la production telles que la volumétrie, la cadence de numérisation et surtout la haute qualité des résultats exigée en bout de chaîne. Peu d'expériences ont été rapportées pour rendre compte de l'efficacité de ces intégrations, mais il est évident d'après les premiers essais que nous avons effectués en entreprise, que leur apport est considérable en termes de gain de productivité, comparé par exemple à la double saisie manuelle.

1.2 Problème de l'évaluation du résultat

Un des principaux problèmes dans le choix d'une méthode est de savoir évaluer le résultat, en fonction de ses besoins. En d'autres termes, la sûreté attendue de la reconnaissance, ou le taux d'erreur toléré est dépendant de l'application visée. Deux exemples vont nous permettre d'illustrer ce fait :

- Dans l'absolue, un taux d'erreur de 90% est correct. Cependant, supposons que l'application concerne la numérisation du Bottin. Un taux de reconnaissance de 90% est catastrophique car cela signifie une erreur tous les 10 caractères ; un numéro de téléphone ayant dix caractères, tous les numéros seront erronés, et donc la reconnaissance sera inutile.
- Dans un autre contexte, AcrobatCapture fait de la recherche de mots « pleine page » sur un texte scanné. Il utilise pour effectuer sa recherche un moteur approché, qui trouve les mots à des erreurs près, et présente comme résultat à l'utilisateur l'image scannée. Dans ce cas, le taux d'erreur est donc moins important.

Actuellement on se dirige vers une reconnaissance mixte, combinant intervention humaine, à la fois pour la préparation des documents, la vérification, la correction et compléter les documents (par exemple : formules) et traitement numérique.

1.3 Organisation de l'article

Cet article présente la technique de numérisation face à l'exigence industrielle, en répondant aux critères économiques relatifs à la volumétrie, la cadence, l'amélioration de la productivité par rapport à la saisie manuelle, la polyvalence et la généricité des plates-formes de numérisation. Il est organisé comme suit :

- la section 2 expose les étapes essentielles d'une chaîne de saisie industrielle
- Nous étudierons dans la section 3 les éléments importants de l'évaluation des performances des OCR. L'accent sera mis sur la performance en reconnaissance de texte; la reconnaissance de la structure ne sera pas développée car elle n'est pas au niveau industriel aujourd'hui.
- Nous montrerons dans la section 4 la réalisation de cette évaluation avec des expérimentations réelles sur des documents.
- La section 5 sera consacrée à la combinaison des OCR.
- La section 6 illustrera sur des exemples les principes exposés dans ce papier.
- Enfin, nous exposerons dans la section 7 les éléments d'une architecture générique d'intégration des outils OCR.

2 Chaîne de numérisation

Les principales étapes d'une chaîne de numérisation sont :

- **L'acquisition** permettant la conversion du document papier sous la forme d'une image numérique (bitmap). Cette étape est importante car elle se préoccupe de la préparation des documents à saisir, du choix et du paramétrage du matériel de saisie (scanner), ainsi que du format de stockage des images.
- **Le pré-traitement** dont le rôle est de préparer l'image du document au traitement. Les opérations de pré-traitement sont relatives au redressement de l'image, à la suppression du bruit et de l'information redondante, et enfin à la sélection des zones de traitement utiles.
- **La reconnaissance du contenu** qui conduit le plus souvent à la reconnaissance du texte et à l'extraction de la structure logique. Ces traitements s'accompagnent le plus souvent d'opérations préparatoires de segmentation en blocs et de classification des médias (graphiques, tableaux, images, etc.).
- **La correction des résultats** de la reconnaissance en vue de valider l'opération de numérisation. Cette opération peut se faire soit automatiquement par l'utilisation de dictionnaires et de méthodes de correction linguistiques, ou manuellement au travers d'interfaces dédiées.

Seules l'acquisition et la reconnaissance du contenu seront détaillées dans la suite. Le pré-traitement est aujourd'hui intégré dans les OCR et est considéré comme suffisant en première approximation. Quant à la reconnaissance des résultats, c'est un problème en soi qui mériterait d'être détaillé dans un autre contexte.

2.1 Acquisition

2.1.1 Principe

L'acquisition du document est opérée par balayage optique. Le résultat est rangé dans un fichier de points, appelés pixels, dont la taille dépend de la résolution. Les pixels peuvent avoir comme valeurs : 0 (éteint) ou 1 (actif) pour des images binaires, 0 (blanc) à 255 (noir) pour des images de niveau de gris, et trois canaux de valeurs de couleurs entre 0 et 255 pour des images en couleur. La résolution est exprimée en nombre de points par pouce (ppp). Les valeurs courantes utilisées couramment vont de 100 à 400 ppp. Par exemple, en 200 ppp, la taille d'un pixel est de 0,12 mm, ce qui représente 8 points par mm. Pour un format classique A4 et une résolution de 300 ppp, le fichier image contient 2 520 $\times$ 3 564 pixels. Il est important de noter que l'image n'a à ce niveau qu'une simple structure de lignes de pixels qu'il faudra exploiter pour retrouver l'information.

2.1.2 Matériel

La technicité des matériels d'acquisition (scanner) a fait un bond considérable ces dernières années. On trouve aujourd'hui des scanners pour des documents de différents types (feuilles, revues, livres, photos, etc.). Leur champ d'application va du "scan" de textes au "scan" de photos en 16 millions de couleurs (et même plus pour certains). La résolution est classiquement de l'ordre de 300 à 1200 ppp selon les modèles.

2.1.3 Formats

Il existe différents formats de représentation des fichiers images : TIFF[1], JPEG[2], GIF[3], PNG[4], etc. dépendant des propriétés de l'image, comme le chromatisme et l'animation, et de la tolérance à la perte de l'information dans le processus de compression. La structuration des données est différente pour chaque format d'image. Un format d'image comprend habituellement une en-tête, contenant des informations générales sur l'ensemble du fichier (par ex. n° de version, ordre des octets, etc.), un ou plusieurs répertoires de paramètres, caractérisant la représentation bitmap de l'image, suivis par les données de l'image qui seront lues et interprétées suivant la valeur des paramètres. Un paramètre particulier est dédié au type de compression autorisé sur le format, avec ou sans perte d'information.

Il est important, dans la mesure du possible, d'écarter les formats propriétaires (comme GIF, sous Licence Unisys) et de leur préférer des formats libres de tous droits. D'autres précautions sont également à prendre en compte concernant les changements de version que peut recouvrir un format, comme c'est le cas du format TIFF dont certaines versions peuvent ne pas être reconnues par certains logiciels.

En attendant la généralisation du PNG, qui est l'émanation de recommandations du consortium W3C (1996), le format TIFF est pour l'instant le format le plus répandu pour les documents textuels, avec le mode de compression sans perte CCITT groupe IV. Lancé par la société Aldus Corporation qui visait le marché de la PAO, il est devenu un standard de fait sur le marché de la microinformatique et permet d'échanger des images en mode point entre systèmes hétérogènes de PAO (PageMaker, Xpress), PréAO (Freelance, PowerPoint), ou éditeur raster (Photoshop, Corel Draw). Il a été spécifié pour prendre en compte différents types d'images (monochromes, à niveaux de gris ou en couleurs), différentes caractéristiques qui dépendent du mode d'acquisition et divers modes de compression. Quasiment tous les produits du marché pour la manipulation des images raster offrent aujourd'hui une interface TIFF, et des librairies de manipulation d'images TIFF, libres de droits, existent [5].

2.1.4 Qualité des données

La qualité de la saisie est tributaire de plusieurs facteurs déterminants, comme:

- **La résolution**. Le choix de la résolution de numérisation est fonction de la qualité du contenu en termes typographiques. La difficulté souvent rencontrée est de pouvoir adapter la résolution aux différentes tailles de caractères et épaisseurs de graphiques présents dans le document, ne nécessitant pas le même niveau de précision (échantillonnage). Par exemple, si la résolution de 200 ppp est suffisante pour du texte en corps 12, elle l'est forcément moins pour des corps plus petits ou du graphique, et excessive pour des corps plus importants ou de la photographie. Le mauvais choix de la résolution, peut conduire, en sous échantillonnage, à un manque d'information (cassure de caractères, élimination de parties de graphiques, etc.), et en sur échantillonnage, à un surplus d'information, souvent traduit par la présence d'un bruit plus abondant.
- **Le contraste et la luminosité**. La luminosité permet de jouer sur l'éclairage du document à capturer : de plus clair à plus sombre. Le contraste permet de faire varier l'accentuation ou l'atténuation des transitions Noir/Blanc. Ces deux paramètres sont

[1] Tagged Image Format File
[2] Joint Photographic Expert Group
[3] Compuserve Graphics
[4] Portable Network Graphics

souvent corrélés entre eux et jouent un grand rôle dans la qualité de reconnaissance. L'expérience montre que l'on peut passer d'un taux de reconnaissance égal à zéro à un taux de reconnaissance supérieur à 99% avec un petit changement de ces paramètres. On peut ainsi éliminer la transparence d'un document, le noircir judicieusement ou le dégraisser comme il faut. Il est assez difficile de trouver le bon réglage pour une classe de documents donnés. On procède assez souvent par la méthode « essais-erreurs » : 1er essai en automatique, examen de la qualité de l'image à l'écran en utilisant le zoom, modifications du contraste et de la luminosité si nécessaire afin de trouver le bon compromis. L'usage d'une mire peut permettre de faire converger plus vite ce processus de réglage. Selon le degré d'hétérogénéité des fonds à traiter, il peut s'avérer impossible de mettre en ouvre un processus industriel automatique. Dans ce cas, il faudrait faire un traitement à façon si on voulait obtenir une qualité optimale. Mais ce n'est souvent pas possible pour des raisons économiques, et on se contente généralement d'un réglage moyen. Il faut noter que certains pilotes logiciels de scanners proposent maintenant une fonction spéciale « Virtual Scan » qui permet d'améliorer les résultats en effectuant un premier réglage suivi d'une comparaison avec un modèle (seuils de tolérance) puis un deuxième réglage plus optimisé : l'établissement des modèles et des règles de choix sont basés sur une procédure d'apprentissage piloté par un opérateur humain dans la phase de mise au point. Enfin, il faut noter que les paramètres de luminosité/contraste sont spécifiques à un modèle physique de scanner et varient dans le temps en fonction du degré d'usure de la lampe, par exemple, ou des capteurs CCD.

- **La qualité du support** influe sur la qualité du résultat. Le choix de la résolution peut permettre de corriger certains de ses défauts comme ceux relatifs, par exemple, au grain du papier. En effet, un bruit à 200 ppp est plus important qu'à 300 ppp. Aujourd'hui, les logiciels d'OCR peuvent être paramétrés pour prendre en compte la qualité du support.
- **L'inclinaison** est une source d'erreur classique, relativement gênante pour les systèmes de reconnaissance qui utilisent l'horizontale comme référentiel de base pour l'extraction des lignes de texte et la modélisation de la forme des lettres. Elle est de plus en plus maîtrisée grâce à l'existence de logiciels de redressement appliqués systématiquement sur les documents à leur entrée.

2.2 Reconnaissance du texte

2.2.1 Principe

La reconnaissance de caractères est réalisée à l'aide de systèmes dédiés appelés OCR. Son but est de convertir l'image du texte en un texte lisible par ordinateur, en faisant le moins de fautes possibles sur la conversion des caractères. L'existence aujourd'hui de plusieurs outils de ce type a conduit peu à peu à définir des critères de choix pour sélectionner l'OCR le plus efficace et surtout le mieux adapté à son application. Longtemps, le critère d'efficacité était lié à un taux de reconnaissance élevé, pensant qu'une technologie efficace est une technologie sans défaut. En effet, il faut admettre que le taux de 100 reste un objectif à atteindre. Mais réussir une opération de numérisation exploitant la technologie d'OCR nécessite un certain nombre de règles dans la mise en œuvre de ces applications. Il est confirmé que le taux de reconnaissance ne dépend pas du seul moteur de reconnaissance mais d'un ensemble de précautions à prendre lors de la mise en œuvre de l'application.

Parmi ces précautions, nous trouvons :

- L’aide dans la préparation en amont du document papier pour réussir une bonne acquisition (ex : réglage du couple contraste/luminosité) et disposer ainsi d’une image facilement traitable,
- L’aide du moteur d’OCR dans le choix de ses paramètres pour mieux s’adapter au type du contenu, en prenant en compte la qualité du document, la langue du texte, la mise en page employée, etc.
- La mise en place de plusieurs moteurs de reconnaissance de caractères, permettant en travaillant sur plusieurs seuils de confiance et sur les résultats apportés par chacun des moteurs d’OCR, de prendre une décision confortée sur le caractère analysé.

2.2.2 Fonctionnement

Un texte est une association de caractères appartenant à un alphabet, réunis dans des mots d’un vocabulaire donné. L’OCR doit retrouver ces caractères, les reconnaître d’abord individuellement, puis les valider par reconnaissance lexicale des mots qui les contiennent. Cette tâche n’est pas triviale car si l’OCR doit apprendre à distinguer la forme de chaque caractère dans un vocabulaire de taille souvent importante, il doit en plus être capable de la distinguer dans chacun des styles typographiques (polices), chaque corps et chaque langue, proposés dans le même document. Cette généralisation omnifonte et multilingue n'est pas toujours facile à cerner par les OCRs et reste génératrice de leurs principales erreurs.

Un système de reconnaissance de textes est composé de plusieurs modules : segmentation, apprentissage, reconnaissance et vérification lexicale[6].

- **La segmentation** permet d’isoler les éléments textuels, mots et caractères, pour la reconnaissance. Elle se base sur des mesures de plages blanches (interlignes et inter caractères) pour faire la séparation. La multiplicité des polices et la variation des justifications empêchent de stabiliser les seuils de séparation, conduisant à la génération de blancs inexistants ou au contraire à l’ignorance de blancs séparateurs de mots. Ce type d’erreur est très fréquent, d’après une récente étude réalisée par Nagy et al.[7] .
- La **reconnaissance de caractères** permet de se prononcer sur l’identité d’un caractère à partir d’un apprentissage de sa forme. Cette étape nécessite une étape préalable de paramétrisation de la forme, définissant des données, des mesures, ou des indices visuels sur lesquels s'appuie la méthode de reconnaissance. Suivant la nature de ces informations, il existe plusieurs catégories de méthodes : syntaxique (description par une grammaire), structurelle (description par un graphe), ou statistique (description par partionnement de l'espace). Ces dernières ont de loin le plus grand intérêt avec les méthodes à base de réseaux de neurones, ou de modèles stochastiques. La complexité de la tâche vient de l'apprentissage qui nécessite, pour sa stabilité, d'un très grand nombre d'échantillons par classe, et de la recherche d'indices visuels discriminants, ce qui n'est pas aisé dans un contexte omnifonte comme celui concerné par la numérisation automatique.
- Le **post-traitement** est effectué quand le processus de reconnaissance aboutit à la génération d'une liste de lettres ou de mots possibles, éventuellement classés par ordre décroissant de vraisemblance. Le but principal est d'améliorer le taux de reconnaissance en faisant des corrections orthographiques ou morphologiques à l'aide de dictionnaires de digrammes, tri-grammes ou n-grammes. Quand il s'agit de la reconnaissance de phrases entières, on fait intervenir des contraintes de niveaux successifs : lexical, syntaxique ou sémantique.

3 Evaluation des performances des OCR

3.1 Principe

L'emploi d'un OCR dans une chaîne de numérisation nécessite une évaluation de ses performances par rapport au jeu de documents à reconnaître. Une approche simpliste d'un système d'OCR serait d'imaginer que sa tâche se limite à localiser l'image de chaque caractère et à le reconnaître indépendamment de toute autre information.

Les logiciels d'OCR sont devenus de plus en plus des systèmes experts à bases de règles intégrant différentes techniques de reconnaissance et entraînés **pour maximiser la performance globale**. Parce que ces systèmes sont à bases de règles, leur performance n'est pas prévisible de manière théorique.

Ce type de système a un comportement qui peut changer de manière imprévisible voyant la performance parfois chuter sur des cas limites. Il suffit parfois d'une petite variation pour que deux caractères, en général bien différenciés, soient confondus. Ces systèmes utilisent des sources d'informations différentes et étagées (taille des caractères, inclinaison des caractères, etc.) pour reconnaître. A cause de la multiplicité de la typographie, ces sources d'information ne peuvent pas être fiables, conduisant à des défaillances dans les cas limites. Ces défaillances sont en plus amplifiées d'étage en étage car chaque étage prend pour acquis les résultats de l'étage précédent.

3.2 Types des erreurs

3.2.1 Erreurs de segmentation

La segmentation du document conduit à la décomposition du document en unités structurelles telles que des régions textuelles ou des graphiques. Une mauvaise application de la méthode de segmentation conduit à des erreurs [20]. Ces erreurs sont :

- **Fusion horizontale de régions textuelles :** cette erreur conduit à la fusion de lignes adjacentes appartenant à des colonnes différentes. Elle influe sur l'ordre de lecture comme le montre la

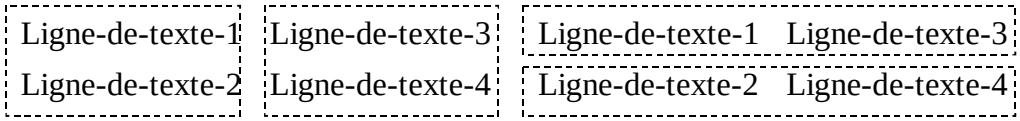
Ligne-de-texte-1	Ligne-de-texte-3	Ligne-de-texte-1 Ligne-de-texte-3
Ligne-de-texte-2	Ligne-de-texte-4	Ligne-de-texte-2 Ligne-de-texte-4

(a) Segmentation correcte (b) Segmentation incorrecte

- Figure 1 où l'ordre normal : 1, 2, 3, 4 est transformé en 1, 3, 2, 4.
- **Fusion verticale de régions textuelles :** cette erreur conduit à regrouper deux paragraphes. Elle n'altère pas l'ordre de lecture mais sa correction est nécessaire pour la classification du texte, par ex. le titre ou des notes de bas de page collés au texte.
- **Scission horizontale de régions textuelles :** cette erreur conduit à un faux ordre de lecture.
- **Scission verticale de régions textuelles :** ce cas est similaire au précédent; il ne cause pas d'erreur grave car l'ordre de lecture n'est pas changé.
- **Région non détectée** : cette erreur indique la non détection d'une région de texte, assimilée sans doute à un graphique ou à du bruit.
- **Graphique/bruit confondu à du texte** : Ceci indique que l'OCR a dû interpréter un graphique ou du bruit comme du texte. Cette erreur conduit à des séquences

perturbées de caractères dans le texte. Cette erreur concerne également les formules mathématiques.

- **Fusion horizontale avec graphique/bruit** : ceci conduit, comme dans le cas précédent, à l'insertion de séquences erronées de caractères dans le texte.
- **Fusion verticale avec graphique/bruit :** Ce cas est identique au cas précédent, sauf qu'il se produit verticalement.



(a) Segmentation correcte (b) Segmentation incorrecte

Figure 1 :Fusion horizontale des régions.

3.2.2 Erreurs de reconnaissance de caractères

Un OCR peut faire quatre types d'erreur sur la reconnaissance des caractères :

- Une confusion, en remplaçant un caractère par un autre, si les caractères sont morphologiquement proches (par ex. « o,0 », « c,(», « n,h », « s,5 ») ;
- Une suppression, en ignorant un caractère, considéré comme un bruit de l'image,
- Un rejet, en refusant un caractère soit parce qu'il n'est pas connu par le système, soit parce que le système n'est pas sûr de sa reconnaissance ; dans ce cas, le système propose un caractère spécial, en général le ~ car celui-ci apparaît rarement dans les documents papier.
- Un ajout, en dédoublant un caractère par deux autres dont la morphologie de leurs formes accolées peut être proche du caractère (par ex. « m, rn», « d,cl », « w,vv »).

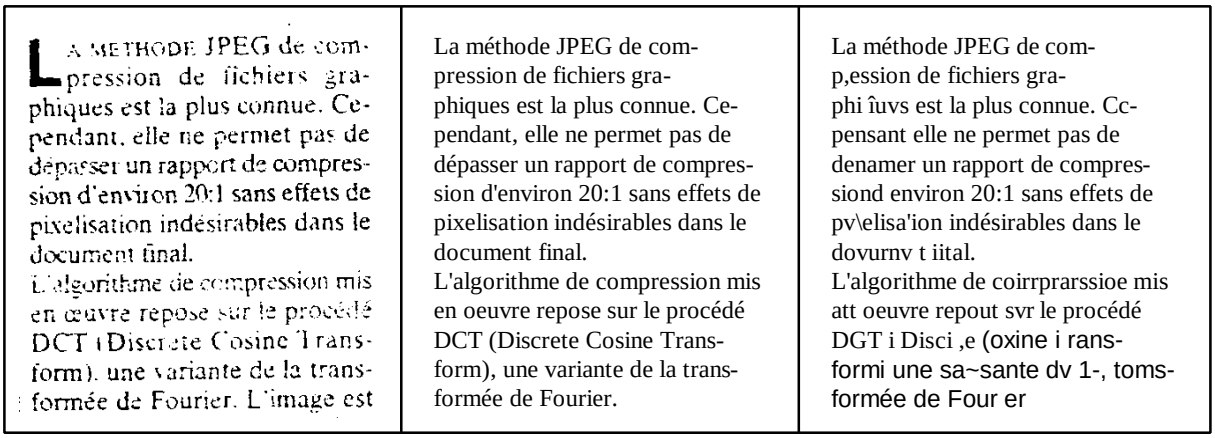
En plus de ces erreurs, les OCR peuvent indiquer les doutes qu'ils ont eu sur certains caractères et certains mots. Ces doutes peuvent servir lors de la phase de correction.

La Figure 2 donne des exemples d'erreurs de reconnaissance possibles. Notons que même l'utilisation d'un dictionnaire ne permet pas de choisir entre les alternatives.

Château	Chanteau ^	Chat eau ^	Chapeau |	Gâteau |v
Original	*Ajout*	*Ajout*	*Confusion*	*Confusion & Suppression*

Figure 2 : Différents cas d'erreurs de reconnaissance possibles sur le mot "château". Toutes les alternatives appartiennent au dictionnaire.

La Figure 3 montre un texte particulièrement bruité et sa reconnaissance par OCR, illustrant tous ces types d'erreur.



a) Image	b) Texte original	c) Résultat OCR
LA METHODE JPEG de compression de fichiers graphiques est la plus connue. Cependant, elle ne permet pas de dépasser un rapport de compression d'environ 20:1 sans effets de pixelisation indésirables dans le document final. L'algorithme de compression mis en œuvre repose sur le procédé DCT (Discrete Cosine Transform), une variante de la transformée de Fourier. L'image est	La méthode JPEG de compression de fichiers graphiques est la plus connue. Cependant, elle ne permet pas de dépasser un rapport de compression d'environ 20:1 sans effets de pixelisation indésirables dans le document final. L'algorithme de compression mis en oeuvre repose sur le procédé DCT (Discrete Cosine Transform), une variante de la transformée de Fourier.	La méthode JPEG de comp,ession de fichiers graphi îuvs est la plus connue. Ccpensant elle ne permet pas de denamer un rapport de compressiond environ 20:1 sans effets de pv\elisa'ion indésirables dans le dovurnv t iital. L'algorithme de coirrprarssioe mis att oeuvre repout svr le procédé DGT i Disci ,e (oxine i ransformi une sa~sante dv 1-, tomsformée de Four er

Figure 3 : Texte bruité et sa reconnaissance OCR.

Nous avons numéroté les erreurs dans la Figure 4 et reporté ces erreurs dans le
.

La méthode JPEG de comp{1}ession de fichiers graphi{2}u{3}s est la plus connue. C{4}pendant, elle ne permet pas de d{5}a{6}er un rapport de compression{7} d{8}environ 20:1 sans effets de p{9}elisa{10}ion indésirables dans le do{11}u{12}t {13}i{14}al. L'algorithme de co{15}pr{16}ssio{17} mis {18} oeuvre repo{19} s{20}r le procédé DC{21}T ({22}Discret{23}e C{24}os{25}ine T{26}ransform), {27} une v{28}ari{29}ante de{3} la{30}tran{31}sformée de Fouri{32}er.

Figure 4 : Détection des erreurs.

Numéro	Correct	Généré	Numéro	Correct	Généré	Numéro	Correct	Généré
1	r	,	12	men	Rnv<espace>	23	ret	i<espace>
2	q	<espace>î	13	f	i	24	C	(
3	e	v	14	n	t	25	s	x
4	e	c	15	m	irr	26	T	<espace>i
5	ép	en	16	e	ar	27	),	i
6	ss	m	17	n	e	28	v	s
7	<espace>	nie	18	en	att	29	ri	vs
8	'	<espace>	19	se	ut	30	a	-,
9	ix	v/	20	u	v	31	ran	om
10	t	'	21	C	G	32	i	<espace>
11	c	v	22	(	i			

Tableau 1 : Catalogue des erreurs répertoriées dans le texte de la Figure 4.

3.2.3 Erreurs de reconnaissance de mots

Une cause d'erreur fréquente est la mauvaise interprétation par l'OCR de la largeur des espaces. Cela peut conduire soit à la fusion de deux mots, soit à la scission d'un mot. La cause principale de suppression correspond à une mauvaise image du mot, comme le surlignage , le

fond grisé, le raturage, etc. ne permettant pas à l'OCR de prendre de décision même partielle sur les caractères, le conduisant à rejeter le mot entier. La suppression peut même toucher des lignes entières, comme les lignes d'en-tête à cause de l'inverse vidéo, de la pliure du papier, etc.

3.3 Mesures de performances sur la reconnaissance de texte

La première mesure à laquelle on peut penser est le taux d'erreur global Γ_{err} qui représente le pourcentage de caractères erronés. Il est exprimé par la formule : $\Gamma_{err} = 100\frac{n_e}{n_c}$ où n_e est le nombre d'erreurs produites *et* n_c est le nombre de caractères dans le texte de référence. Une erreur est signalée si le caractère reconnu ne correspond pas au caractère attendu, souvent confondu avec un autre. Pour cette raison, le taux d'erreur est souvent synonyme de taux de confusion.

En complément au taux d'erreur, on peut penser au taux de rejet Γ_{rej} qui représente le pourcentage de caractères qui ont été considérés douteux par le système. Il est exprimé par la formule : $\Gamma_{rej} = 100\frac{n_r}{n_c}$ où n_r est le nombre de rejets.

Le taux de reconnaissance Γ_{rec} , taux le plus affiché, représente le pourcentage decaractères correctement reconnus. Il est exprimé par le rapport suivant :

$$\Gamma_{rec} = 100\frac{n_c - n_r - n_e}{n_c}$$

Ces trois taux vérifient l'égalité suivante :

$$(1)\ \Gamma_{rej} + \Gamma_{err} + \Gamma_{rec} = 100\%$$

On parle également de taux de confiance ou fiabilité (ang. *Reliability*) Γ_{conf}, un taux de reconnaissance obtenu en faisant du rejet suivant la formule :

$$\Gamma_{conf} = 100\frac{n_c - n_r - n_e}{n_c - n_r} = \frac{\Gamma_{rec}}{\Gamma_{rec} + \Gamma_{err}}$$

En milieu industriel, le seul critère de performance utilisée est le taux d'erreur final $\Gamma_{rec} + \Gamma_{err}$ le plus bas possible. Idéalement, on cherche à maximiser Γ_{rec} et à minimiser $\Gamma_{err,}$ mais cela est très difficile à faire en pratique. On peut également augmenter le rejet pour diminuer la confusion.

A titre d'indication, le standard de qualité défini dans le monde de l'édition est de 1/10000. Généralement, un OCR du commerce présente un taux de reconnaissance total Γ_{rec} + Γ_{err} de l'ordre de 99%, soit pour une page moyenne de 3500 caractères : 35 erreurs à corriger égal au taux de rejet Γ_{rej}. Le problème est que le taux d'erreur n'est pas connu et peut être important, de l'ordre de 1% à 5%.

3.3.1 Contrôle de l'OCR

Il n'est pas pensable d'éliminer le taux de rejet et de forcer le système à décider de l'identité de chaque échantillon puisque les décisions faites dans le cas incertain peuvent causer une augmentation disproportionnée du taux d'erreur. Ceci étant, il serait naturel pour la

mesure de la performance d'un OCR de contenir quelques relations entre les taux de reconnaissance/rejet et le taux d'erreur.

Les taux d'erreur et de rejet sont corrélés (en première approximation) en raison inverse selon la formule [13]:

$$\Gamma_{err} = 1/\ \Gamma_{rej}$$

Ce qui veut dire que la seule façon de diminuer le taux de confusion est d'augmenter le taux de rejet. Il y a trois moyens plus ou moins fins de contrôler le taux de rejet :

- Si on dispose d'un taux de confiance par caractère, on peut contrôler le taux de rejet par rapport à cette confiance. Plus le seuil de confiance est élevé plus le taux de rejet est important de l'ordre de 3% à 5% mais le taux de confusion reste relativement significatif de l'ordre de 1 pour cent à 1 pour mille. La conséquence est que le nombre de caractères à corriger devient important et qu'au-delà d'un certain taux de caractères à corriger, il devient économiquement plus rentable de ressaisir la totalité du texte, ce qui rend inutile l'utilisation de l'OCR.
- Si au contraire, on dispose d'une appréciation dans une échelle donnée, le contrôle se fait de manière approximative par rapport à l'un des paliers proposés
- Si on a le contrôle sur l'apprentissage, et plus précisément sur la distribution des échantillons, on peut agir plus finement sur le rejet en fonction d'un taux de recouvrement plus important des classes.

Dans le cas des OCR du commerce, on ne peut pas parler de contrôle du rejet, car d'une part, on a peu d'information sur la reconnaissance, et d'autre part on a peu de connaissance sur le fonctionnement de l'apprentissage. Donc, on peut soit utiliser les appréciations ou taux de confiance sachant que l'on ne sait même pas à quoi ils correspondent réellement, soit simuler un taux de confiance a posteriori. On utilise une matrice de confusion calculée à partir de la distance d'édition. Mais ces taux de confiance seront une simple estimation car un même caractère aura toujours le même taux de confiance.

3.3.2 Prise en compte d'une correction humaine

Dans le cas d'une correction des erreurs par un être humain, on doit tenir compte du taux d'erreur intrinsèque Γ_{hum} de celui-ci, qui est d'ordre de 1 pour mille à 3 pour mille.

Soit Γ_{cor} le taux de caractères correctement corrigés par l'opérateur humain, on a la relation suivante :

$$(2)\ \Gamma_{rej} = \Gamma_{cor} + \Gamma_{hum}$$

qui exprime le fait que la correction des caractères rejetés par l'OCR est entâchée d'un taux d'erreur résiduel égal au taux d'erreur intrinsèque de l'opérateur.

Ces considérations expliquent que pour obtenir un niveau de qualité correspondant à un taux d'erreur inférieur à 1 caractère erroné pour 2 000 ou 5 000 ou 10 000 caractères saisis, les seules procédures de production satisfaisantes sont basées soit sur des processus de double saisie avec comparaison ou des processus de simple saisie mais avec relecture systématique : le choix de la procédure dépendant du niveau de qualité exigé.

En conclusion, on peut résumer le problème à résoudre sous la forme d'une contrainte déduite du rapprochement des formules (1) et (2).

$$\Gamma_{rec} + \Gamma_{err} + \Gamma_{cor} + \Gamma_{hum} = 100\%$$

$$\text{soit } (\Gamma_{rec} + \Gamma_{cor}) + (\Gamma_{err} + \Gamma_{hum}) = 100\%$$

$$\text{soit encore } G_{err} + G_{hum} < \mathbf{0{,}0001}$$

Ce qui veut dire que les taux à minimiser reviennent, dans le cas industriel, au taux d'erreur humaine et à celui de la confusion du système. Les réduire à moins de 1/ 10 000 veut dire avoir un OCR faisant peu de confusion, et peu de rejet et donc moins de corrections conduisant à peu d'interventions humaines et donc à moins d'erreurs (et un gain de productivité important).

3.3.3 Types d'évaluation des performances

Il y a deux manières d'évaluer les OCR suivant le type d'information que l'on peut ou que l'on souhaite extraire d'un OCR. On peut considérer l'OCR comme une boîte noire (système fermé) duquel seul le résultat de la reconnaissance des mots est accessible. Dans ce cas, l'évaluation est faite de manière globale. Dans le cas où l'on dispose d'une information plus précise (par exemple, position dans l'image des caractères extraits par l'OCR), on peut affiner la mesure de la performance.

3.3.3.1 Evaluation globale

C'est le cas où la seule information disponible est la liste des caractères et des rejets. La seule évaluation possible est de calculer le taux d'erreur Γ_{err} ou le taux de reconnaissance Γ_{rec}. Pour se faire, on a besoin de trouver les erreurs. Cela revient à mesurer**,** par mise en correspondance entre les couples de chaînes de références et de résultats, les ajouts, les suppressions et les substitutions.

Si l'OCR se comporte comme une boîte noire, il est impossible, à cause de ce problème de non unicité, de faire un choix entre deux possibilités de coût identique et donc de connaître précisément le nombre de substitutions et de suppressions. Ceci ne permet pas de détailler l'erreur et surtout de déterminer l'erreur de substitution qui dénote la confusion. Or, ce type d'erreur est souvent difficile à détecter par ailleurs car elle conduit à des mots souvent validés par un dictionnaire. Les deux autres erreurs sont moins fondamentales car il y a plus de chance que l'erreur soit flagrante. On souhaiterait utiliser le taux de confusion afin d'évaluer la performance du système. On peut en estimer un à partir de la solution de correspondance de coût minimum.

A partir de cette mise en correspondance, on peut déterminer tous les taux précédents, le taux de rejet étant donné par l'OCR.

3.3.3.2 Evaluation locale

Dans ce cas, on dispose de la position des caractères (glyphes) dans l'image. On maîtrise la réponse sur chaque caractère et on peut déterminer précisément le type d'erreur commise sur chaque caractère. Cela permet par conséquent de calculer le taux de substitution qui posait problème dans le cas de l'évaluation globale.

3.4 Mesures de performances sur la segmentation

Ainsi que nous l'avions signalé dans l'introduction, les outils de segmentation ne sont pas un enjeu industriel aujourd'hui. Néanmoins, la mesure de performances des OCR peut se poursuivre sur la qualité de la segmentation physique du document en régions. Cette opération est en effet déterminante pour rétroconvertir le texte, essentiellement dans le cas où la

structure du document est multi- colonne, comprend des tableaux et des graphiques. Une erreur sur la segmentation des régions peut fausser l'ordre de leur lecture.

L'algorithme de segmentation n'est pas une tâche facile. En effet, une segmentation optimale n'est pas bien définie et surtout non unique pour une même page. Deux résultats de segmentation en régions avec des caractéristiques différentes peuvent aboutir à des textes identiques. La littérature propose plusieurs articles sur l'étude de la segmentation [19-26].

Deux méthodes de mesure de performance de la segmentation sont proposées dans la littérature: l'une est basée sur l'image et l'autre est basée sur le texte. La première évalue la segmentation au niveau du pixel, tandis que la deuxième l'évalue au niveau du caractère.

- **Approche image :** L'évaluation utilise l'image du document et le zonage du document de référence dans lequel les régions sont décrites par des polygones. L'évaluation effectue une comparaison géométrique entre les deux en testant l'affiliation de chaque pixel aux régions correspondantes [20,21]. La comparaison se base sur des opérations de fusion-scission de régions dans les deux directions principales. La méthode classe les erreurs dans 19 catégories différentes. La qualité de la segmentation est déterminée par le nombre de pixels ou caractères dans les régions mal segmentées du document.
- **Approche textuelle :** L'évaluation opère sur la sortie textuelle de l'OCR. L'OCR est d'abord appliqué sur le document initial, générant des erreurs à la fois sur la segmentation et sur les caractères. Ensuite, l'OCR est de nouveau appliqué sur la même image du document, mais produite par le zonage manuel du document de référence. Le résultat du second traitement contient uniquement des erreurs d'OCR. Pour les deux textes, les coûts d'erreurs de correction sont calculés par des algorithmes de mise en correspondance de chaînes (basée sur le principe de la distance d'édition) [9]

  Kanai et al. [17,19] ont introduit une métrique du calcul de la performance du zonage, basée sur le nombre d'opérations d'édition nécessaires pour transformer la sortie d'un OCR en une chaîne de texte lisible. La chaîne de caractères produite par l'OCR est comparée à la chaîne de texte correct et les plus longues sous-chaînes communes sont d'abord identifiées et étiquetées. Toutes les sous-chaînes non mises en correspondance sont ensuite identifiées par un algorithme de mise en correspondance opérant par fusion-scission de chaînes. Enfin, le système calcule le nombre minimal d'opérations d'édition (insertions, suppressions et déplacements) nécessaires pour corriger la sortie de l'OCR et avoir la même disposition des sous-chaînes [30].

  L'avantage d'une telle méthode est que, étant basée uniquement sur le texte, elle ne nécessite de spécifier aucun format particulier pour la représentation zonale. Cependant, la sortie est un nombre d'opérations d'édition à partir desquelles, il est difficile de dériver beaucoup d'information sur les erreurs de segmentation.

  Agne et al. [20] propose une méthode basée sur des comparaisons approximatives de chaînes, ce qui permet de comparer directement le texte de référence au résultat de l'OCR sans effectuer de segmentation manuelle comme dans l'étude précédente. Cependant, cette approche ne permet que d'approcher le vrai nombre d'erreurs de segmentation.

L'approche image ne peut être utilisée que quand le zonage automatique est disponible, ce qui n'est pas toujours fourni par les OCR. L'approche texte suppose que les erreurs de segmentation et les erreurs d'OCR sont indépendantes, ce qui n'est pas toujours vrai. De plus, cette approche ne permet aucune classification des erreurs de segmentation; Aussi, Les deux approches nécessitent la préparation manuelle d'un document de référence pour chaque document, ce qui implique beaucoup d'effort et donc un coût.

4 Réalisation de l’évaluation

4.1 Principe

Comme nous l'avions déjà signalé, nous ne nous intéressons qu'au résultat de la reconnaissance de texte et nous supposerons que celui de la segmentation est correct. Nous étudierons dans cette section la classe des OCR capables de reconnaître des textes imprimés ou dactylographiés, ayant une structure normalisée et un ordre de lecture logique. C'est le cas des moteurs d'OCR disponibles sur le marché.

Nous entendons par réalisation de l'évaluation la mesure de l'écart qui sépare le document exact du résultat de l'OCR. La technique la plus directe pour le faire est de synchroniser les deux textes afin de repérer et de compter les erreurs.

4.2 Synchronisation et alignement

Que ce soit pour évaluer des OCR ou pour combiner leurs résultats il faut procéder à un alignement entre les documents, qui soit le plus exact possible, i.e. mettant en relief les erreurs potentielles et leurs positions.

On appelle alignement de deux chaînes, la superposition des deux chaînes dans lesquelles on a ajouté des caractères de synchronisation en début, en fin ou au milieu des deux chaînes, de telle sorte que les deux chaînes obtenues soient alignées et de même longueur**.**

Exemple : un alignement des deux textes : *Un chêne* et *Des chaînes*

```
-Un chê--ne-
Des ch-aînes
```

Dans l'exemple, le caractère d'alignement est le tiret : -

On dira que l'alignement est le meilleur alignement possible, s'il existe une métrique pour mesurer l'alignement, et si la valeur obtenue par cette métrique est la plus petite possible.

4.3 Document de référence

On appelle évaluation, la comparaison du document obtenu par passage d'une image à un OCR, avec un document de référence (DR) qui correspond au document attendu par l'OCR.

4.3.1 Représentativité du DR

L'évaluation impose de préparer soigneusement un jeu de test le plus représentatif possible des données manipulées[16,17]. La représentativité se traduit par un bon échantillonnage aléatoire d'unités de traitement : document, page ou partie de page, en fonction des unités qui seront fournies au logiciel.

Par exemple, dans le cas de documents de type catalogues de notices bibliographiques, un bon échantillonnage serait de donner comme unités de traitement au système, des blocs de notices pris au hasard sans considération de la structure du contenu, mais d'un nombre suffisant pour espérer une bonne représentativité du contenu.

Un autre exemple concernant les documents composites, un bon échantillonnage serait de séparer les parties du document en fonction des médias qu'elles représentent : notes, corps de texte, tableaux, etc.

L'erreur que l'on fait souvent consiste à mettre en avant les cas difficiles qui sont souvent les plus rares. Donc calculer la performance du système sur ces cas extrêmes ne permet pas de juger de l'efficacité d'un logiciel. Supposons que les documents traités contiennent deux polices, par exemple, 90% de Times et 10% de Courier. En prenant un échantillonnage équilibré : 50% de chaque police, et supposons qu'un logiciel OCR reconnaît à 100% le Times et à 50% le Courier, on obtiendra sur notre échantillon un taux de reconnaissance de 75%, alors que sur nos documents, le taux de reconnaissance sera de 95%. Si on prend maintenant un autre logiciel qui reconnaît Times à 80% et Courier à 80%, le taux de reconnaissance sur l'échantillon sera de 80% meilleur que pour le premier OCR, mais le taux sur le document sera aussi de 80% et donc largement inférieur au premier document. L'idéal est de combiner les deux logiciels, permettant d'atteindre les performances les plus élevées, soit en théorie : 90% sur l'échantillon et 98% sur les documents.

Ce raisonnement peut conduire à penser qu'il suffit de présenter au logiciel d'OCR, 900% caractères en Times et 100% en Courier pour déterminer la performance du logiciel. Mais, comme on l'a dit précédemment, le logiciel d'OCR utilise le contexte pour reconnaître les caractères et ainsi la performance ne peut pas être facilement maîtrisée. Il faut donc fournir aux logiciels plutôt des unités de traitement que des détails sur le contenu.

4.3.2 Obtention

Le document de référence (DR) peut être obtenu de trois manières différentes :

- Soit sous forme électronique déjà constituée et accompagnant les documents à reconnaître; par exemple un journal informatisé depuis un an et les documents ont été sauvegardés sous forme électronique (PAO, Xpress, SGML, etc.). Un travail de préparation (reformatage) peut être parfois nécessaire pour disposer d'un document au format désiré pour l'évaluation.
- Soit en le tapant si le document source n'existe pas. Une double saisie est souvent nécessaire pour disposer d'un document correct.
- Soit enfin en le saisissant par un OCR, si on a confiance dans un OCR donné, en portant beaucoup de soin à la correction des résultats de l'OCR, de manière à ce qu'il y ait le moins d'erreurs résiduelles possibles.

4.3.3 Evaluation de l'OCR bien segmenté

On dit que le résultat de l'OCR est bien segmenté si le découpage opéré par cet OCR correspond à l'ordre de lecture souhaité. Cette segmentation correcte peut être obtenue :

- soit manuellement en définissant des zones sur l'image et en donnant l'ordre de succession des zones (tous les logiciels d'OCR permettent cette fonctionnalité), ce qui permet d'assurer l'uniformité des résultats de tous les OCR;
- soit automatiquement, dans le cas de documents simples ne contenant qu'une seule colonne ou plusieurs colonnes bien séparées avec un ordre naturel de lecture (de haut en bas et de gauche à droite pour du texte, et ligne par ligne et colonne par colonne pour un tableau), ou dans certains cas complexes où nous avons l'assurance que l'OCR est capable de réaliser une segmentation fidèle.

4.3.4 Structure identique

Un document est une suite de pages découpées en colonnes composées de paragraphes et de lignes. Deux documents ont une structure identique s'il y a une correspondance entre

tous les éléments à tous les niveaux de la structure. Cependant, les documents issus de traitement de texte n'ont pas la notion de paragraphe, ce qui peut poser des problèmes de synchronisation avec le document de référence. Un travail de préparation supplémentaire par balisage des paragraphes doit être effectué.

Dans ce cas, on dispose de points de synchronisation ou repères (page, colonne, paragraphe et ligne). Pour réaliser l'évaluation, on va synchroniser les documents aux limites de ces repères.
Il y a trois niveaux :

- Un premier niveau (en général la page ou la colonne) où l'on sait exactement où se trouvent les points de synchronisation, c'est-à-dire que l'on sait que l'élément n du document OCR correspond à l'élément n du DR. Cela veut dire qu'il y a autant d'éléments dans chaque document numérotés de manière identique.

  Pour i de 1 à n
  synchroniser Dr_i avec OCR_i

  Un deuxième niveau (en général les paragraphes et les lignes) où l'on peut avoir des doutes sur la séparation des paragraphes et des lignes. Les erreurs possibles sont des insertions, des suppressions et fusions d'éléments. On utilise la programmation dynamique (cf. §10.2) permettant un recalage des lignes autorisant les contractions et les éclatements. La programmation dynamique nécessite le calcul d'une distance d'édition entre les lignes.

- Un troisième niveau correspond à la synchronisation des caractères à l'intérieur des lignes. Une fois que l'on a la correspondance entre les paragraphes et les lignes de documents, il faut faire la synchronisation des caractères à l'intérieur des lignes par la distance d'édition, la programmation dynamique ou par la méthode de la plus longue sous-séquence commune (cf. §10.4).

Il y a deux justifications dans le choix des méthodes de synchronisation : le coût et la précision locale. La programmation dynamique donne la meilleure synchronisation globale, mais avec un coût relativement important et une solution ne correspondant pas forcément à la solution recherchée. Si on synchronise caractère par caractère les deux documents dans leur globalité, on obtiendra une synchronisation globale optimale, mais des erreurs peuvent se répercuter d'un élément de segmentation à un autre, n'assurant pas une synchronisation optimale locale.

4.3.5 Structure différente

On entend par structure différente, un DR où on a uniquement la notion de paragraphe (cas des documents produits par des traitements de texte). C'est en général le cas où l'on dispose des DR sous forme électronique.

Dans ce cas, on serait tenté d'utiliser les paragraphes comme points de synchronisation mais cela conduirait à un échec car ils correspondent à des erreurs fréquentes des OCR. Donc, il faut supprimer dans l'OCR, les retours à la ligne, et dans le DR, les débuts de paragraphes, et synchroniser les deux documents obtenus comme deux longues chaînes.

Pour synchroniser les deux documents, la méthode qui s'impose est la recherche des plus longues sous-chaînes communes (cf. §10.5). Mais cela n'évite pas les synchronisations aberrantes en cas de problèmes sur les plus longues chaînes. Ces problèmes conduisent à faire

la synchronisation sur des chaînes plus courtes qui ne correspondent pas forcément. La Figure 5 montre une mauvaise synchronisation basée sur la plus longue sous-chaîne commune.

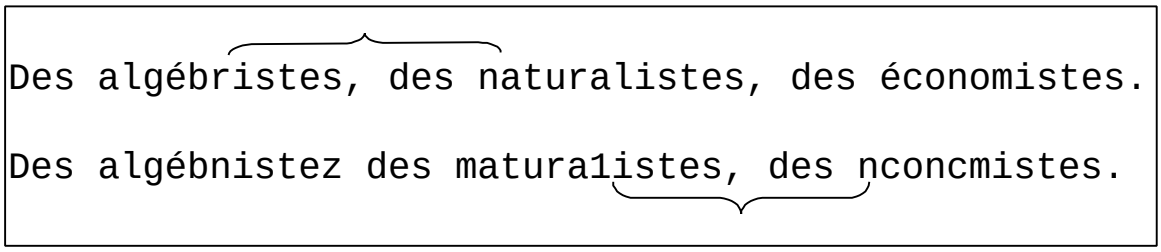


Figure 5 : Aberration de la plus longue sous-chaîne commune.

4.4 Exemples de comparaison de documents

Ces exemples ont été réalisés avec le programme *diff* de Gnu [32] . Gnu *diff* utilise l'algorithme de Myers [15] pour effectuer la comparaison ligne à ligne, qui est une optimisation de l'algorithme de programmation dynamique de Wagner et Fisher. Il existe une adaptation de *diff* appelé *wdiff* qui fait la comparaison mot à mot. Nous en avons dérivé notre propre adaptation *cdiff* pour faire la comparaison caractère par caractère.

Soient les deux fichiers suivants à comparer :

Fichier 1

```
The string string correction problem is
to determire the distance between two
by the mininnum cost sequene of 'edit operations".
```

Fichier 2

```
The string to string correction problem is
to determine the distance between two
strings as measured
by the minimum cost sequene of "edit operations".
```

4.4.1 Comparaison ligne par ligne avec Gnu Diff

```
1,3c1,4
< The string string correction problem is
< to determire the distance between two
< by the mininnum cost sequene of 'edit operations".
---
> The string to string correction problem is
> to determine the distance between two
> strings as measured
> by the minimum cost sequene of "edit operations".
```

Cela signifie qu'il faut changer toutes les lignes du fichier 1 pour obtenir le fichier 2. Dans le cas d'une correction, cela signifie qu'il faut retaper tout le document.

4.4.2 Comparaison mot par mot avec *wdiff* :

```
The string {+to+} string correction problem is
to [-determire-] {+determine+} the distance between two
{+strings as measured+}
by the [-mininnum-]{+minimum+} cost sequene of [-'edit-] {+"edit+} operations".
```

Cela signifie qu'il faut faire 5 opérations pour passer du fichier 1 au fichier 2 :

- Substituer les trois mots suivants :
  [-determire-] par {+determine+}, [-mininnum-] par {+minimum+} et [-'edit-] par {+"edit+};
- Insérer le mot suivant : {+to+};
- Insérer la ligne suivante : {+strings as measured+}.

On voit que les opérations sont plus localisées que dans le cas ligne par ligne.

4.4.3 Comparaison caractère par caractère avec *cdiff*

```
The string {+to+} string correction problem is
to determi[-r-]{+n+}e the distance between two
{+strings as measured+}
by the mini[-nn-]{+m+}um cost sequene of [-'-]{+"+}edit operations".
```

Cela signifie qu'il faut faire 5 opérations pour passer du fichier1 au fichier2:

- 3 substitutions : [-r-] par {+n+}, [-nn-]{+m+} et [-'-] par {+"+};
- 1 insertion de mot : {+to+};
- 1 insertion de ligne : {+strings as measured+}.

4.4.4 Conclusion

En conclusion, wdiff est un très bon candidat pour comparer les documents issus d'un OCR deux à deux sur la base des mots. Quand on souhaite comparer un résultat d'OCR à un document témoin, on peut se passer de la connaissance des fins de lignes dans le document témoin. La performance en temps de ce programme permet même d'envisager la comparaison complète d'un document de plusieurs dizaines de pages, si l'on ne dispose pas de la structuration en page dans le document témoin.

5 Combinaison d'OCR

Les OCR n'atteindront jamais les 100% de bonne reconnaissance. On observe par ailleurs, que différents moteurs d'OCR produisent des erreurs différentes. C'est cette idée qui a conduit les chercheurs ces dernières années à développer des techniques de combinaison de moteurs OCR afin d'améliorer le résultat global. L'objectif est de tirer parti des avantages de chaque OCR et d'écarter leur faiblesse [27].

Dans les différentes études effectuées par Rice[1-4], il est montré que de l'ordre de 50% d'erreur est éliminée par la combinaison de plusieurs OCR ayant des taux de reconnaissance individuels de l'ordre de 97%. Cela étant, ce gain ne peut être atteint que dans la mesure où les erreurs proviennent des OCR et non de la qualité de l'image, et où les OCR sont de bonne qualité.

5.1 Fondements

La combinaison d'OCR nécessite de connaître les éléments suivants :

- **Degré d'intégrité** : il faut tenir plus ou moins compte du résultat produit par l'OCR en fonction de la confiance que l'on peut avoir dans le moteur. Ceci nécessite donc de connaître ses forces et ses faiblesses.
- **Détail des réponses :** la nature des réponses pour chaque glyphe peut varier depuis un simple nom de caractère jusqu'à la fourniture d'un degré de confiance pour chaque lettre de l'alphabet, en passant par une liste ordonnée de lettres possibles. On imagine que le détail des réponses influe sur la stratégie de combinaison.
- **Indépendance des moteurs :** dans l'idéal, il faut choisir des moteurs les plus indépendants possibles de manière à ne pas trop donner d'importances aux décisions prises sur des critères identiques.

5.2 Stratégies de combinaison

Nous n'allons pas ici faire un état de l'art complet des méthodes de combinaison possibles des OCR ni parler des méthodes de post-traitement lexicaux pour augmenter la fiabilité des OCR. Rappelons que notre objectif est d'obtenir un texte le plus proche possible du texte contenu dans l'image, ce qui implique de ne pas corriger les fautes originelles éventuelles (fautes de frappe).

Nous allons examiner en fonction du niveau de détail fourni au niveau de la lettre quelques stratégies de combinaison possibles :

- **Simple caractère** : ce niveau de détail ne permet que la stratégie de combinaison la plus simpliste, celle du vote majoritaire. Le caractère retenu sera celui le plus répandu parmi toutes les réponses des OCR.
- **Liste ordonnée de caractères** : la méthode la plus répandue, appelée « Borda Count » [27] donne une note à chaque réponse en fonction du nombre de fois où la réponse est présente dans les OCR, pondérée par la position dans la liste (rang).
- **Degré de confiance** : dans ce cas, il est d'abord nécessaire de normaliser les degrés de confiance qui ne sont pas nécessairement des probabilités. Suivant la dépendance ou non des classifieurs, on pourra utiliser soit les techniques basée sur la théorie de probabilités conditionnelles (de Bayes), soit celles basées sur la combinaison de la croyance (Dempster-Schafer) [28].

Le tableau 2 présente un exemple des trois combinaisons évoquées. L'article de Dengel et al. [27] décrit plusieurs méthodes de combinaison.

Soit E la « fonction classifieur »
Soit 4 le nombre de classifieurs

(a) $e_1(x)=17$ $e_2(x)=4$ $e_3(x)=17$ $e_4(x)=21$

Classe	n.de votes
4	1
17	2
21	1

→ Sortie simple : E(x)=17

(b) $e_1(x)$ $e_2(x)$ $e_3(x)$ $e_4(x)$

$e_1(x)$	$e_2(x)$	$e_3(x)$	$e_4(x)$
17	4	17	21
4	21	21	17
21	17	9	9
9	9	4	4

$B(4)=B_1(4)+B_2(4)+B_3(4)+B_4(4)=2+3+0+0=5$
$B(9)=0+0+1+1=2$
$B(17)=3+1+3+2=9$
$B(21)=1+2+2+3=8$

→ Sortie simple : E(x)=17 ; Sortie liste : E(x)

E(x)
17
21
4
9

(c)

i	4	9	17	21
$P_1(C_i\|x)$	0.3	0.1	0.4	0.2
$P_2(C_i\|x)$	0.3	0.5	0.1	0.2
$P_3(C_i\|x)$	0.08	0.1	0.3	0.5
$P_4(C_i\|x)$	0.02	0.04	0.1	0.8

$P(C_4|x)=(0.3+0.3+0.08+0.02):4=0.175$
$P(C_9|x)=0.0725$
$P(C_{17}|x)=0.275$
$P(C_{21}|x)=0.375$

→ Sortie simple : E(x)=21 ; Sortie liste : E(x)

E(x)
21
17
4
9

Tableau 2 : Résultas de combinaison de classifieurs d'après [27].

5.3 Recalage du texte

Comme dans le cas de l'évaluation individuelle globale, on retombe sur le problème du recalage du texte pour effectuer la combinaison entre OCR. On peut soit se fier à la position des glyphes dans l'image (souvent fournie par les OCR), soit utiliser la distance d'édition pour mettre en correspondance les caractères.

L'alignement multiple permet de comparer plusieurs chaînes. L'idée est de trouver une chaîne de référence (chaîne de consensus) à partir de laquelle on initialisera l'alignement. Ensuite, cet alignement est rectifié itérativement par comparaison deux à deux. Les étapes de l'algorithme sont :

- On commence par évaluer l'alignement minimum deux à deux (par exemple par calcul de la distance d'édition) et obtenir une valeur numérique qui va en décroissant. Elle est égale à 0 si les deux chaînes sont égales.
  Soient les trois chaînes :

  1 Cette chaîne
  2 Ces chaînes
  3 Ce Chêne

Les distances d'édition entre ces chaînes sont données dans le tableau suivant :

	1	2	3	Somme
1	0	4	5	9
2	4	0	4	8
3	5	4	0	9

- Sélectionner la chaîne qui a la plus petite somme comme chaîne de référence, puis faire l'alignement avec les autres chaînes, avec n'importe quelle méthode :
  Alignement avec 1 :

  Ces-- chaînes
  Cette chaîne-

  Alignement avec 3 :

  Ces-- chaînes
  Ce--- chêne--

- Si des tirets ont été rajoutés dans la chaîne de référence, il faut les rajouter également dans les autres chaînes déjà comparées.

C'est un algorithme purement itératif. Il ne garantit pas l'optimum; il est 2 fois inférieur à l'optimum. Pour garantir l'optimum, il faut utiliser la programmation dynamique, ce qui conduit à une explosion du calcul.

Rice et al.[9] ont proposé un algorithme rapide basé sur la recherche de la plus longue chaîne commune, la chaîne initiale étant formée par la concaténation des chaînes représentant les résultats de tous les OCR.

6 Expérimentations et résultats

6.1 Principe

Nous allons tester cinq OCR du commerce (Capture, FineReader, OmniPage, TextBridge, TypeReader). Le but est d'appliquer les principes précédents pour recaler les réponses des OCR par rapport aux documents de référence. Nous en dériverons les taux d'erreur et de reconnaissance par classe de caractère et par OCR. Nous utilisons le logiciel OCR Frontiers Toolkit 1.0 de l'ISRI (Université du Nevada) [33] pour calculer le résultat.

Les documents sont présentés aux OCR sans aucun prétraitement, sans segmentation manuelle; la seule indication concerne la langue : français même pour les documents en anglais.

6.2 Données de test

Les pages numérisées sont au nombre de 15, elles sont numérisées à 300dpi. On peut répartir les 15 pages dans 4 catégories :

- 4 pages contiennent du texte français sur une seule colonne, le texte provient d'une machine à écrire et correspond à une police Courier 12 points.

- 4 pages contiennent du texte français sur deux colonnes dans la police Times 12 point.
- 3 pages contiennent du texte anglais sur deux colonnes dans la police Times 12 point.
- 4 pages contiennent du texte sur deux colonnes, une en anglais et l'autre en français, rédigées dans les polices Times 10 et 12 point.

Les pages ne contiennent ni images, ni tableaux, ni graphiques, mais par endroit les caractères sont mal imprimés. Les 15 pages ont été présentées aux 5 OCR. Il y a en tout 64313 caractères (y compris les espaces et les retours à la ligne).

6.3 Résultats

Les résultats de reconnaissance sont résumés dans les tableaux joints. Le **Tableau 3** donne les précisions de reconnaissance par type de caractères, et le **Tableau 4** donne la précision de reconnaissance par caractère.

Précision de reconnaissance par classe de caractères

Classes	*Occurr.*	*Capture*		*FineReader*		*OmniPage*		*TextBridge*		*TypeReader*	
		Raté	*%Juste*	*Raté*	*%Juste*	*Raté*	*%Juste*	*Raté*	*%Juste*	*Raté*	*%Juste*
ASCII Chiffres	**442**	**38**	**91,40%**	**7**	**98,42%**	**29**	**93,44%**	**15**	**96,61%**	**18**	**95,93%**
ASCII Lettres minuscules	**48707**	**941**	**98,07%**	**226**	**99,54%**	**573**	**98,82%**	**412**	**99,15%**	**2483**	**94,90%**
ASCII Espaces	**10732**	**47**	**99,56%**	**44**	**99,59%**	**58**	**99,46%**	**55**	**99,49%**	**286**	**97,34%**
ASCII Symboles spéciaux	**2126**	**55**	**97,41%**	**96**	**95,48%**	**62**	**97,08%**	**83**	**96,10%**	**214**	**89,93%**
ASCII Lettres majuscules	**936**	**115**	**87,71%**	**11**	**98,82%**	**40**	**95,73%**	**40**	**95,73%**	**59**	**93,70%**
Latin1 Lettres minuscules	**1343**	**1343**	**0,00%**	**9**	**99,33%**	**44**	**96,72%**	**33**	**97,54%**	**572**	**57,41%**
Latin1 Symboles spéciaux	**26**	**26**	**0,00%**	**3**	**88,46%**	**4**	**84,62%**	**24**	**7,69%**	**11**	**57,69%**
Latin1 Lettres majuscules	**1**	**1**	**0,00%**	**1**	**0,00%**	**0**	**100,00%**	**0**	**100,00%**	**1**	**0,00%**

Tableau 3 : Précisions de reconnaissance par type de caractères.

Précision de reconnaissance triée par ordre croissant de %right

		Capture		*FineReader*		*Omnipage*		*TextBridge*		*TypeReader*	
Car.	*Occur.*	*Misd*	*%Right*	*Misd*	*%Right*	*Misd*	*%Right*	*Misd*	*%Right*	*Misd*	*%Right*
{É}	1	1	0,00%	1	0,00%	0	100,00%	0	100,00%	1	0,00%
{»}	13	13	0,00%	2	84,62%	3	76,92%	13	0,00%	7	46,15%
{«}	13	13	0,00%	1	92,31%	1	92,31%	11	15,38%	4	69,23%
{ù}	8	8	0,00%	1	87,50%	2	75,00%	1	87,50%	5	37,50%
{î}	8	8	0,00%	0	100,00%	4	50,00%	0	100,00%	5	37,50%
{â}	7	7	0,00%	0	100,00%	3	57,14%	2	71,43%	2	71,43%
{û}	10	10	0,00%	0	100,00%	5	50,00%	1	90,00%	4	60,00%
{à}	155	155	0,00%	0	100,00%	4	97,42%	8	94,84%	105	32,26%
{ê}	59	59	0,00%	0	100,00%	9	84,75%	6	89,83%	24	59,32%
{ï}	3	3	0,00%	0	100,00%	0	100,00%	0	100,00%	2	33,33%
{ô}	22	22	0,00%	0	100,00%	4	81,82%	2	90,91%	6	72,73%
{é}	951	951	0,00%	8	99,16%	12	98,74%	11	98,84%	388	59,20%
{O}	36	18	50,00%	5	86,11%	5	86,11%	10	72,22%	6	83,33%
{è}	114	114	0,00%	0	100,00%	1	99,12%	2	98,25%	30	73,68%
{?}	13	0	100,00%	0	100,00%	7	46,15%	3	76,92%	5	61,54%
{ç}	6	6	0,00%	0	100,00%	0	100,00%	0	100,00%	1	83,33%
{W}	18	9	50,00%	1	94,44%	2	88,89%	1	94,44%	5	72,22%
{ }	9395	40	99,57%	31	99,67%	50	99,47%	54	99,43%	273	97,09%
{:}	23	1	95,65%	1	95,65%	1	95,65%	16	30,43%	3	86,96%
{"}	8	0	100,00%	0	100,00%	0	100,00%	5	37,50%	2	75,00%
{I}	107	36	66,36%	0	100,00%	19	82,24%	10	90,65%	12	88,79%
{M}	51	15	70,59%	0	100,00%	4	92,16%	4	92,16%	6	88,24%
{-}	363	4	98,90%	77	78,79%	7	98,07%	8	97,80%	71	80,44%
{w}	284	56	80,28%	1	99,65%	54	80,99%	20	92,96%	11	96,13%
{H}	4	1	75,00%	0	100,00%	0	100,00%	1	75,00%	0	100,00%
{Y}	4	1	75,00%	0	100,00%	1	75,00%	0	100,00%	0	100,00%
{3}	16	2	87,50%	1	93,75%	1	93,75%	1	93,75%	2	87,50%

{5}	23	5	78,26%	1	95,65%	2	91,30%	1	95,65%	1	95,65%
{K}	11	3	72,73%	0	100,00%	0	100,00%	1	90,91%	1	90,91%
{N}	26	5	80,77%	2	92,31%	1	96,15%	1	96,15%	0	100,00%
{1}	73	9	87,67%	1	98,63%	7	90,41%	7	90,41%	4	94,52%
{;}	28	0	100,00%	1	96,43%	1	96,43%	1	96,43%	6	78,57%
{y}	381	20	94,75%	6	98,43%	22	94,23%	14	96,33%	57	85,04%
{8}	42	4	90,48%	2	95,24%	3	92,86%	2	95,24%	0	100,00%
{h}	1018	78	92,34%	63	93,81%	41	95,97%	11	98,92%	38	96,27%
{0}	118	8	93,22%	1	99,15%	9	92,37%	2	98,31%	7	94,07%
{4}	17	2	88,24%	0	100,00%	2	88,24%	0	100,00%	0	100,00%
{D}	17	0	100,00%	0	100,00%	0	100,00%	2	88,24%	2	88,24%
{E}	105	6	94,29%	2	98,10%	3	97,14%	4	96,19%	6	94,29%
{U}	25	2	92,00%	0	100,00%	1	96,00%	2	92,00%	0	100,00%
{j}	85	1	98,82%	0	100,00%	2	97,65%	3	96,47%	11	87,06%
{.}	443	11	97,52%	7	98,42%	14	96,84%	8	98,19%	42	90,52%
{'}	520	4	99,23%	2	99,62%	24	95,38%	23	95,58%	44	91,54%
{A}	105	12	88,57%	0	100,00%	3	97,14%	1	99,05%	4	96,19%
{9}	59	6	89,83%	1	98,31%	1	98,31%	1	98,31%	1	98,31%
{i}	3755	34	99,09%	9	99,76%	18	99,52%	33	99,12%	600	84,02%
{V}	11	2	81,82%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{m}	1372	56	95,92%	5	99,64%	63	95,41%	25	98,18%	94	93,15%
{,}	691	35	94,93%	8	98,84%	8	98,84%	19	97,25%	40	94,21%
{v}	595	23	96,13%	2	99,66%	23	96,13%	8	98,66%	39	93,45%
{R}	25	0	100,00%	1	96,00%	0	100,00%	1	96,00%	2	92,00%
{7}	33	1	96,97%	0	100,00%	3	90,91%	1	96,97%	0	100,00%
{g}	629	8	98,73%	2	99,68%	19	96,98%	8	98,73%	56	91,10%
{l}	2382	83	96,52%	9	99,62%	32	98,66%	49	97,94%	146	93,87%
{z}	26	0	100,00%	0	100,00%	0	100,00%	1	96,15%	2	92,31%
{x}	184	8	95,65%	0	100,00%	3	98,37%	2	98,91%	8	95,65%
{L}	54	2	96,30%	0	100,00%	0	100,00%	0	100,00%	4	92,59%
{T}	68	2	97,06%	0	100,00%	1	98,53%	0	100,00%	4	94,12%
{n}	3601	45	98,75%	21	99,42%	42	98,83%	46	98,72%	185	94,86%
{6}	21	1	95,24%	0	100,00%	1	95,24%	0	100,00%	0	100,00%
{S}	43	0	100,00%	0	100,00%	0	100,00%	2	95,35%	2	95,35%
{F}	11	1	90,91%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{u}	2479	27	98,91%	8	99,68%	32	98,71%	18	99,27%	119	95,20%
{e}	7055	230	96,74%	25	99,65%	44	99,38%	34	99,52%	208	97,05%
{b}	491	6	98,78%	3	99,39%	2	99,59%	7	98,57%	17	96,54%
{d}	1889	32	98,31%	3	99,84%	12	99,36%	22	98,84%	73	96,14%
{o}	3185	36	98,87%	8	99,75%	19	99,40%	10	99,69%	163	94,88%
{2}	40	0	100,00%	0	100,00%	0	100,00%	0	100,00%	3	92,50%
{t}	4181	32	99,23%	39	99,07%	45	98,92%	20	99,52%	146	96,51%
{r}	3178	35	98,90%	3	99,91%	34	98,93%	21	99,34%	134	95,78%
{f}	794	9	98,87%	2	99,75%	5	99,37%	4	99,50%	33	95,84%
{(}	15	0	100,00%	0	100,00%	0	100,00%	0	100,00%	1	93,33%
{c}	1615	16	99,01%	3	99,81%	12	99,26%	9	99,44%	58	96,41%
{a}	3552	48	98,65%	9	99,75%	27	99,24%	14	99,61%	92	97,41%
{s}	3971	42	98,94%	4	99,90%	15	99,62%	18	99,55%	135	96,60%
{q}	495	1	99,80%	1	99,80%	3	99,39%	4	99,19%	16	96,77%
{p}	1430	14	99,02%	0	100,00%	4	99,72%	11	99,23%	41	97,13%
{C}	127	0	100,00%	0	100,00%	0	100,00%	0	100,00%	5	96,06%
{<\n>}	1337	7	99,48%	13	99,03%	8	99,40%	1	99,93%	13	99,03%
{k}	80	1	98,75%	0	100,00%	0	100,00%	0	100,00%	1	98,75%
{/}	1	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{!}	2	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{Q}	2	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{%}	4	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{)}	15	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{B}	15	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{J}	21	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{G}	22	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%
{P}	28	0	100,00%	0	100,00%	0	100,00%	0	100,00%	0	100,00%

Tableau 4 : Précision de reconnaissance par caractère

7 Architecture d'intégration des OCR

Quand on souhaite réaliser une plate-forme de numérisation, on doit se rendre le plus indépendant possible des logiciels d'OCR proprement dit afin essentiellement de pouvoir changer facilement de logiciel suivant les types de documents à traiter et/ou la sortie de nouveaux logiciels plus performants.

L'indépendance vis à vis des logiciels d'OCR s'obtient suivant trois critères: le format d'image en entrée, le format de sortie et l'activation du logiciel (paramétrage et appel) :

- **Format d'entrée :** il faut ramener toutes les images au même format. On choisira en général TIFF qui est le format donné par la majorité des scanners et accepté par la plupart des OCR. Si cela n'est pas le cas, pour un OCR particulier, il faudra développer la conversion du format TIFF au format demandé par l'OCR. Cela devra être transparent dans la chaîne de numérisation; le programme devant être appelé par l'opérateur avec un fichier TIFF. L'indépendance du format d'entrée permet d'utiliser des logiciels de prétraitement d'images, pour enlever le bruit et redresser l'image si nécessaire, qui peuvent ne pas être fournis par le logiciel d'OCR.
- **Format de sortie :** ce format doit être normalisé pour faciliter tous les traitements ultérieurs. Si l'on souhaite juste s'occuper du texte, un format texte ISO peut suffire. Dans le cas de documents structurés, un format plus évolué, type XML, reprenant les informations physiques du document (position du texte, police, structure des tableaux) est sans doute plus approprié. Le format de sortie servira à évaluer et à qualifier des OCR, à combiner le résultat de plusieurs OCR, à corriger les erreurs des OCR, et à effectuer des conversions vers des formats cibles plus proches de la structure logique des documents. On voit la difficulté de la tâche s'il fallait reprogrammer tous ces traitements à chaque fois que l'on veut utiliser un nouveau logiciel d'OCR. Beaucoup de traitements se font en aval de l'OCR et nécessitent donc une indépendance vis à vis de ces logiciels. Il y a toujours un effort de programmation pour normaliser le format de sortie de l'OCR (chaque OCR a son propre format de sortie) vers le format de sortie désiré. Cela peut se faire soit par l'API, soit par un programme spécifique.
- **Activation et Paramétrage :** il faut proposer une interface générale pour appeler les OCR afin de faciliter la tâche de l'opérateur et l'intégration de nouveaux OCR. L'effort de programmation se situe dans l'adaptation entre la méthode d'appel de l'OCR et l'interface générale.

Les logiciels de GED, qui intègrent un module d'OCR (type Ascent de Kofax), utilisent ce type d'approche indépendante de l'OCR, ce qui leur permet d'adapter leur logiciel à la demande du client avec tel ou tel OCR.

La Figure 6 montre l'architecture que nous avons utilisée pour l'intégration de plusieurs OCR. Le paramétrage et l'activation sont réalisés par un OCRAdapter. Un Adapter [34] est un modèle conceptuel pour masquer les implantations spécifiques de différents programmes qui ont la même finalité. Il est réalisé à l'aide d'une classe C++ qui est dérivée pour chaque OCR. La classe contient les méthodes d'appel et de paramétrage de l'OCR.

Le format intermédiaire, appelé XmlLayout, décrit via une DTD XML le résultat de la reconnaissance de la structure physique en termes d'une arborescence d'objets de type : *document*, *page*, *région*, *bloc*, *ligne*, *mot* et *glyphe*. Les seules relations entre les objets sont de type composition. Chacun de ces objets a une liste d'attributs : *rectangle englobant*, *police*, *coefficient de confiance* et *média*. Quand le média est une image ou un graphique, il est décrit par une URI faisant une référence extérieure à l'objet. Quand le média est un tableau, nous

avons choisi d'ignorer sa présence pour le moment. Les caractères sont codés en UNICODE. Pour les besoins de l'affichage, nous avons intégré la DTD SVG (Scalable Vector Graphics) qui permet la visualisation graphique du résultat. D'autres tentatives de normalisation de type de format existent, comme le DAFS de l'Université du Nevada [29]. La conversion du format interne de l'OCR vers XmlLayout est intégrée dans OCRAdapter.

Le format final, appelé XmlDoc, décrit via une DTD XML le résultat de la structure logique mettant en relief les objets logiques représentant le document. Cette DTD est spécifique à chaque type d'application réalisée.

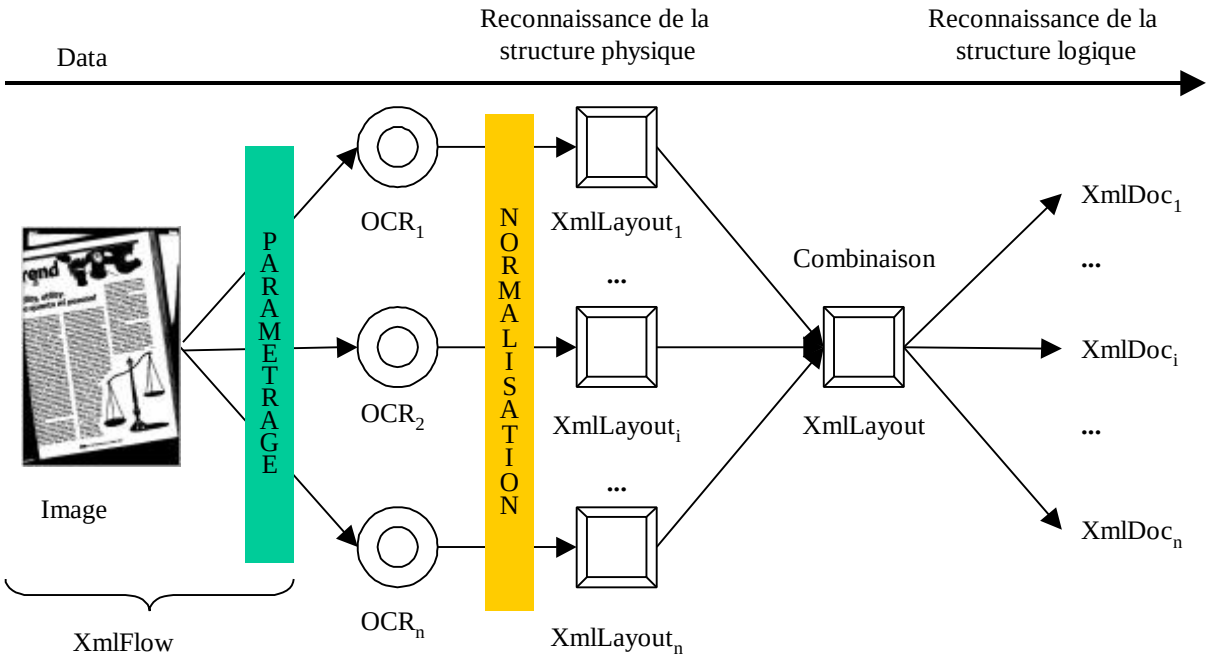


Figure 6 : Architecture de la plate-forme de numérisation.

8 Conclusion

Les OCR ont aujourd'hui atteint une maturité qui permet leur utilisation en production industrielle. Si on peut assurer un gain de productivité, on ne pourra jamais atteindre une reconnaissance parfaite. Cela peut-être important pour des documents particuliers (par ex. juridiques où il est important de conserver les erreurs de frappe). Les erreurs commises par la machine ne sont pas du même type que celles commises par un être humain, ce qui peut être considéré comme un avantage ou comme un inconvénient. En dernière analyse, le maître mot doit revenir à l'opérateur. C'est pourquoi, l'interface utilisateur est un élément essentiel de la chaîne de numérisation.

Bien que la qualité des OCR ne cesse de croître, des limites sont toujours présentes au niveau de la reconnaissance : des symboles, du manuscrit, des structures fines (formules, tableaux, bibliographies, etc.), des structures hétérogènes (formulaires, plans, dessins techniques, documents géographiques, etc.).

Nous avons montré dans cet article comment utiliser les OCR de manière générique et avantageuse. Une solution viable d'intégration, basée sur une adaptation des interfaces des divers moteurs du commerce, permet d'évaluer les performances individuelles et d'augmenter la reconnaissance totale par combinaison. La concentration par rachat des fournisseurs de

moteurs de reconnaissance va dans ce sens. Ainsi, la plupart des kits de développement du commerce proposent en fait plusieurs moteurs, et l'on commence a trouver en standard des mécanismes de vote.

La numérisation se dirige aujourd'hui vers le scan2xml, c.à.d. la rétroconversion des documents images vers des structures évoluées normalisées. Si les freins sont peu nombreux pour des documents composites (édités suivant des feuilles de style conventionnelles), car les techniques ont été largement validées en laboratoire, les défis restent importants pour des documents complexes où il n'existe pas de structure éditoriale.

Aujourd'hui l'OCR démocratisé, on pourrait alors se poser la question concernant la survivance de la technique OCR par rapport au nombre croissant de documents directement produits sous forme numérique. Un élément de réponse tient dans les techniques développées autour de l'OCR qui peuvent être utilisées à d'autres fins.

9 Bibliographie

[1] S. V. Rice, J. Kanai, and T. A. Nartker, "An Evaluation of OCR Accuracy," ISRI 1993 AnnuaI Research Report, University of Nevada, Las Vegas, April 1993, 9-31.

[2] S. V. Rice, J. Kanai, and T. A. Nartker, "The Third Annual Test of OCR Accuracy, " ISRI 1994 Annual Research Report, University of Nevada, Las Vegas, April 1994, 11-38.

[3] S. V. Rice, F. R. Jenkins, and T. A. Nartker, "The Fourth AnnuaI Test of OCR Accuracy," ISRI 1995 Annual Research Report, University of Nevada, Las Vegas, April 1995, ll- 49.

[4] S. V. Rice, 'The OCR Experimental Environment, Version 3," ISRI 1993 Annual Research Report, University of Nevada, Las Vegas, April 1993, 83-86.

[5] www.libtif.org

[6] G. Nagy, What does a machine need to read a document ? Proc. of of the 1st Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada, March 1992, University of Nevada, 1-10.

[7] S. V. Rice, G. Nagy and T. A. Nartker and, "Optical Charcater Recognition : An illustrated guide to the frontier", Kluwer Academic Publishers, 1999.

[8] H. Bunke, "String Matching for Pattern Recognition", in eds. H. Bunke and A. Sanfeliu Structural Pattern Analysis, (World Scientific Publication Co., Singapore, 1990), 119-144.

[9] S. V. Rice, J. kanai and T. A. Nartker, "An algorithm for matching OCR-generated text strings", International Journal of Pattern Recognition, vol. 8, n. 5, 1994, 1259-1266.

[10] R. A. Wagner and M. J. Fisher, "The String-to-String Correction Problem," Journal of the ACM, 21(l), (1974) 168-173.

[11] P. A. V. Hall and G. R. Dowling, "Approximate String Matching," ACM Comput. Surv. **12** (1980), 381-402.

[12] W. J. Masek and M. S. Paterson, "A Faster Algorithm Computing String Edit Machine," Journal Computer Systems Sci. **20** , 1 (1980) 18-31.

[13] C. K. Chow, "On Optimum Recognition Error and Reject Tradeoff," IEEE Transactions on Information Theory, Volume IT-16, No. 1, Jan. 1970, 41-46.

[14] T. A. Nartker, "Benchmarking DIA Systems," Handbook of Character Recognition and Document Image Analysis, Eds. H. Bunke and P.S.B. Wang, 801-820, Chapter 31.

[15] W. Miller and E.W. Myers, A File Comparison Program, Software, Practice and Experience, Vol. 15, No. 11, p 1025-40.

[l6] R. C. Camp, "BENCHMARKING: The Search for Industry Best Practices that Lead to Superior Performance," *ASQC Quality Press, Milwaukee*, WI, 1989.

[17] J. Kanai, T. A. Nartker, S. V. Rice, and G. Nagy, “Performance Metrics for Document Understanding Systems”, Proc. Second International Conference on Document Analysis and Recognition, Tsukuba Science City, Japan, Oct. 1993, 424-427.
[18] S. Chen, S. Subramaniam, R. M. Haralick, and I. T. Phillips, “Performance Evaluation of Two OCR Systems,” Proc. SDAIR ‘94, Las Vegas, NV, April 1994, 299-317.
[19] J. Kanai, S. V. Rice, and T. A. Nartker, “Automated Evaluation of OCR Zoning,” IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 17, (Jan. 1995) 86-90.
[20] S. Agne and M. Rogger, Benchmarking of Document Page Segmentation, Part of the IS&T/SPIE Conference on Document Recognition and Retrieval VII, San Jose, California, January 2000, pp. 165-171.
[21] S. Randriamasy and L. Vincent, “A region-based system for the automatic evaluation of page segmentation algorithms”, in Proceedings od DAS'94, A. Dengel and L. Spitz, eds., pp. 29-41, Kaiserslautern, Germany, October 1994.
[22] S. Randriamasy and L. Vincent, “Benchmarking Page Segmentation Algorithms”, in Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 411-416, IEEE Computer Society Press, Seattle, Washington, June, 21-23, 1994
[23] A. Yanikoglu and L. Vincent, “Ground-truthing and benchmarking document page segmentation,” in Proceedings of the Third International Conference on Document Analysis and Recognition, vol. 2, pp. 601-604, IEEE Computer Society Press, (Montréal, Canada), August 14-16 1995.
[24] M. Thulke, V. Mcgner, and A. Dengel, “A general approach to quality evaluation of document segmentation results,” in Proceedings of the International Association for Pattem Recognition Workshop on Document Analysis Systems DAS98, S.-W. Lee and Y. Nakano, eds., pp. 79-88, (Nagano, Japan), November 1998.
[25] J. Kanai, S. V. Rice, and T. A. Nartker, “A preliminary evaluation of automatic zoning,” in Annual Research Report, K. 0. Grover, ed., pp. 35-45, Information Science Research Institute, (University of Nevada, Las Vegas), 1993.
[26] J. Kanai, S. V. Rice, T. A. Nartker, and G. Nagy, “Automated evaluation of OCR zoning,” IEEE Bansactions on Pattem Analysis and Machine Intelligence 1'7, pp. 86-90, January 1995.
[27] A. Dengel, R. Hoch, F. Hönes, T. Jäger, M. Malburg and A. Weigel, “Techniques for Improving OCR Results,” Handbook of Character Recognition and Document Image Analysis, Eds. H. Bunke and P.S.B. Wang, 227-258, Chapter 38.
[28] E. Mandler and J. Schürmann, “Combining the Classification Results of Independent Classifiers based on the Dempster/Shafer theory of evidence,” In Pattern Recognition and Artificial Intelligence, ed. by E.S. Gelsema and L.N. Kanal, Elsevier Science Publishers B.V., North Holland, 1998, 381-393.
[29] RAF Technology, Inc., “DAFS: Document Attribute Format Specification,” Redmond, WA (1995).
[30] S. Latifi, “How can permutations be used in the evaluation of zoning algorithms?,” International Journal of Pattem Recognition and Artificial Intelligence 10(3), pp. 223-237, 1996.
[31] J. W. Hunt and T. G. Szymanski, “A fast Algorithm for Computing Longest Common Subsequences,” Communications of the ACM, Vol. 20, No. 5, p. 350-3, May 1997.
[32] http://www.gnu.org/software/diffutils/diffutils.html
[33] http://www. ISRI
[34] E. Gamma, Richard Helm, Ralph Johnson, and John Vlissides "Design Patterns: Elements of Reusable Object-Oriented Software", ISBN 0-201-63442-2, 416 pages, 1995

10 Annexe : Détail des méthodes de synchronisation

10.1 Distance d'édition

Levinstein propose une distance basée sur le comptage de transformations élémentaires de type SUBstitution, INSertion et DEStruction, faisant intervenir un coût par type de transformation. Soient deux chaînes a $=\{a_i\}_{i=1,n}$ et b $=\{b_j\}_{j=1,m}$

- SUB : $a_i \rightarrow b_j$ coût : $\gamma(a_i,b)$
- INS : $l \rightarrow b_j$ coût : $\gamma(\lambda,b_j)$
- DES : $a_i \rightarrow l$ coût : $\gamma(a_i,\lambda)$

Par exemple, la chaîne a = xy *peut être transformée en* b=yxz *par les opérations :* ($l \rightarrow$z), (y $\rightarrow$ *l*), (z $\rightarrow$ y), ($l \rightarrow$ z) *donnant les chaînes :* xy, zxy, zx, yx, yxz. Le coût de la transformation est : $\gamma(l,z) + \gamma(y, l) + \gamma(z,y) + \gamma(l,z)$.

Cette distance est appelée distance d'édition. La distance d'édition est une bonne métrique pour évaluer un alignement car elle représente l'effort qu'il faut fournir pour passer d'une chaîne à une autre. Les algorithmes d'alignement de deux chaînes vont avoir pour but de trouver l'alignement, pas forcément unique, qui minimise cette distance.

10.2 Comparaison dynamique

Wagner et Fisher [10] ont proposé une méthode d'évaluation récursive de la distance d'édition, basée sur la notion de trace. La trace permet :

- de décomposer le processus de comparaison en l'optimisant localement,
- d'avoir un processus croissant et continu.

L'algorithme utilise un tableau d'édition D(n,m) :

D(0,0)=0
Pour i:=1 **à** n **faire**
    D(i,0) := D(i-1,0) + $\gamma(a_i,\lambda)$
Pour j:=1 **à** m **faire**
    D(0,j) := D(0,j-1) + $\gamma(\lambda,b_j)$
Pour i:=1 **à** n **faire**
    Pour j:=1 **à** m **faire**

$$D(i,j) = \textbf{min} \begin{cases} D(i-1,j-1) + \gamma(a_i, b_j) \\ D(i-1,j) + \gamma(a_i, \lambda) \\ D(i,j-1) + \gamma(\lambda, b_j) \end{cases}$$

La distance d'édition $\delta(a,b) = D(n,m)$

Un exemple de calcul est montré dans le **Tableau 5** sur les chaînes "preterit" et "zeitgeist" avec les initialisions suivantes : D(i,0) = i et D(0,j) =j

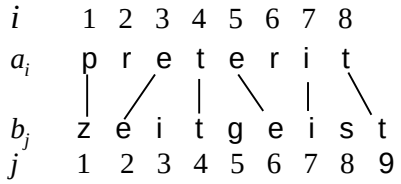
	j	0	1	2	3	4	5	6	7	8	9
i			z	e	i	t	g	e	i	s	t
0		**0**	1	2	3	4	5	6	7	8	9
1	p	1	**1**	2	3	4	5	6	7	8	9
2	r	2	**2**	2	3	4	5	6	7	8	9
3	e	3	**3**	**2**	3	4	5	5	6	7	8
4	t	4	4	3	**3**	**3**	4	5	6	7	7
5	e	5	5	4	4	4	**4**	**4**	5	6	7
6	r	6	6	5	5	5	5	5	**5**	6	7
7	i	7	7	6	5	6	6	6	**5**	6	7
8	t	8	8	7	6	5	6	7	6	**6**	**6**

Tableau 5 : Tableau des distances D entre les deux chaînes "preterit" et "zeitgeist".

Cet algorithme fonctionne en O(nm) et occupe un espace quadratique en O(nm). L'algorithme de recherche de la trace travaille à partir de l'élément D(n,m) du tableau D des distances et extrait le chemin en cherchant à chaque fois l'élément qui a conduit à obtenir l'élément courant par minimisation de la distance (éléments en gras dans le tableau). L'algorithme de calcul de la trace est le suivant :

```
i=m
j=n;
Tanque (i>0) et (j>0)
        si D(i,j) = D(i-1,j) + γ(a_i,λ)
                i = i-1;
        sinon si D(i,j) = D(i,j-1) + γ(λ,b_j)
                        j = j-1
                sinon
                        print(i,j)
                        i=i-1;
                        j=j-1
```

La trace correspondant à l'exemple précédent est :



La séquence d'édition a été obtenue de la trace en : supprimant les symboles dans a non atteints, insérant ceux de b non atteints et en substituant les autres.

10.3 Optimisation

Miller et Myers [15] ont proposé une optimisation de l'algorithme de Wagner et Fisher en évitant de calculer tous les éléments de la matrice d'édition. Le principe est de ne remplir que les diagonales.

Les étapes principales de l'algorithme peuvent être résumées comme suit :

- Initialiser le début de la première diagonale à 0 : D(0,0) = 0;
- Ajouter 1 (coût de la transformation locale) à droite et en bas des éléments déjà calculés pour insérer un tiret dans l'une ou l'autre des deux chaînes (délétion ou insertion).
- Recopier l'extrémité de la diagonale à la suite si les caractères mis en correspondance sont égaux.

Les tableaux de la **Figure 7** montrent les étapes de calcul pour les deux chaînes : abcabba et cbabac.

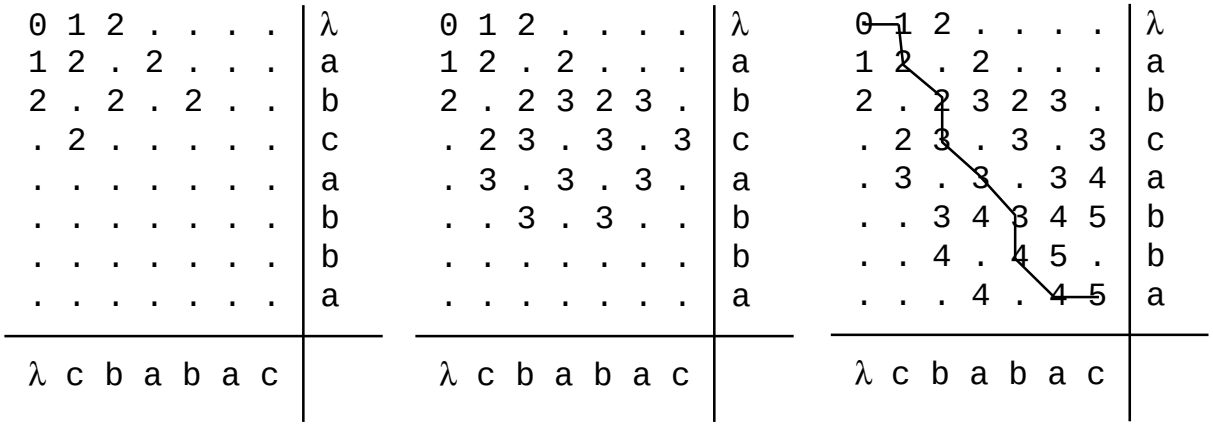


Figure 7 : Optimisation de Myers, limitant la programmation dynamique aux diagonales.

L'alignement réalisé est le suivant :

abcabba-
cb-ab-ac

L'avantage de cet algorithme est qu'il ne nécessite pas de remplir tout le tableau. De plus, seules les extrémités des diagonales sont nécessaires à chaque itération pour poursuivre les calculs. Comme l'algorithme s'arrête dès que D(m,n) est atteint, seules les diagonales entre -d et + d sont considérées, ce qui compte moins de (2d+1)min(m,n). (2d+1) est le nombre de diagonales examinées. De plus, le temps d'exécution est proportionnel à $\min(m,n)+d^2$. Cet algorithme a donné naissance à la commande Unix, *diff* (cf. §4.4) .

10.4 Plus longues sous-séquences communes

Hunt et Szymanski [31] ont proposé un algorithme plus rapide pour trouver itérativement la sous-séquence commune à deux chaînes. Cette sous-séquence correspond à la sous-chaîne commune contenant tous les éléments égaux.

La recherche se fait sur la chaîne la moins longue x, de longueur n. Elle consiste à calculer un tableau $k_{i,l}$ où i est le numéro du caractère courant dans x, et l est la longueur de la sous-séquence. Le caractère courant est comparé à chaque caractère de la chaîne y, en partant de la fin, puis plusieurs cas peuvent se produire :

- Le caractère égal se trouve à un seul endroit de y et le lien qui le lie avec celui de x ne croise pas un autre lien; dans ce cas, on reporte la ligne précédente en ajoutant en $k_{i,l+1}$ l'indice de l'élément.
- Le caractère égal se trouve à un seul endroit de y et le lien qui le lie avec celui de x croise un autre lien; dans ce cas on défait tous les liens précédents et on marque en $k_{i,l+1}$ l'indice de l'élément.
- Le caractère égal se trouve à plusieurs endroits de y; dans ce cas, on reporte la ligne précédente en ajoutant en $k_{i,l+1}$ l'indice de l'élément, mais également les autres indices aux bons endroits en respectant la croissance des $k_{i,l}$.

Si la méthode de comparaison dynamique prend un temps quadratique, celle-ci est en O((r+n)log n) pour des chaînes de longueurs égales où r est le nombre de couples de positions où les éléments correspondent. Dans le plus mauvais cas, chaque élément de x correspond avec n'importe quel élément de y, conduisant à n^2 paires de correspondance et une complexité de $O(n^2 \log n)$. Cependant, dans plusieurs applications où la différence de fichiers peut se mesurer au niveau des lignes, r peut être proche de n, donnant une performance de (O log n), une amélioration importante par rapport au temps quadratique.

Le tableau suivant montre les étapes de calcul de la sous-séquence commune des deux chaînes "preterit" et "zeitgeist".

		l								
		0	1	2	3	4	5	6	7	8
1	p	0	10	10	10	10	10	10	10	10
2	r	0	10	10	10	10	10	10	10	10
3	e	0	2	10	10	10	10	10	10	10
4	t	0	2	4	10	10	10	10	10	10
5	e	0	2	4	6	10	10	10	10	10
6	r	0	2	4	6	10	10	10	10	10
7	i	0	2	3	6	7	10	10	10	10
8	t	0	2	3	4	7	9	10	10	10

10.5 Plus longues chaînes communes

L'autre idée pour éviter l'encombrement mémoire à la construction du tableau d'édition est de rechercher les chaînes communes, et réserver la recherche de transformations uniquement aux chaînes différentes.

La recherche de chaînes communes peut se faire par la construction d'un arbre de suffixes. Le principe de construction est relativement simple: la chaîne est à chaque fois rognée d'un caractère, puis la chaîne restante (le suffixe) est insérée dans l'arbre à partir de la racine, conduisant soit à un nouveau chemin, soit à une nouvelle ramification. La suite de nœuds sans ramification est compactée faisant état d'une seule chaîne. Ces chaînes constituent les plus longues chaînes communes de leurs ramifications. Il suffit ensuite, de parcourir l'arbre pour retrouver la plus longue chaîne dans l'arbre, ce qui peut être fait en un temps linéaire. Un exemple d'arbre de suffixes partiel de la chaîne "PABCQRABCSABTU$" est donné dans la Figure 8.

Rice et al. [9] ont proposé une procédure de synchronisation de textes basée sur la recherche de plus longues chaînes communes. Elle s'applique récursivement sur les chaînes restantes jusqu'à ce qu'aucune chaîne commune ne soit plus trouvée. Cette procédure est généralisable facilement à N chaînes. La Figure 9 donne l'algorithme de synchronisation pour deux chaînes.

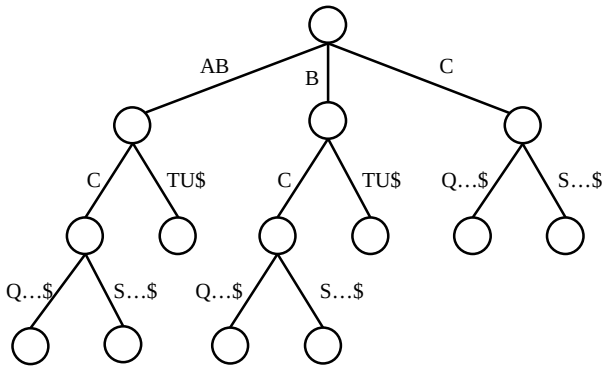


Figure 8: Arbre de suffixe partiel de la chaîne "PABCQRABCSABTU".

Procedure sync_2_strings($s_1[\alpha_1:\omega_1]$, $s_2[\alpha_2:\omega_2]$)
begin
if longest_match ($s_1[\alpha_1:\omega_1]$, $s_2[\alpha_2:\omega_2]$, α'_1, ω'_1 , α'_2, ω'_2)
then
sync_2_strings($s_1[\alpha_1: \alpha'_1-1]$, $s_2[\alpha_2: \alpha'_2-1]$)
imprimer la correspondance décrite par α'_1, ω'_1, α'_1, α'_2 et ω'_2
sync_2_strings($s_1[\omega'_1+1: \omega_1]$, $s_2[\omega'_2+1: \omega_2]$)
endif
end

Figure 9 : Algorithme de synchronisation de deux chaînes.