Télécom SudParis

Année scolaire 2014/2015

Projet Informatique 1ère Année

CSC 3502

Reconnaissance de Texte sur une image

Besga Simon

Guillem Maxime

Malnuit Lucie

Rousset Antonin.

Enseignant responsable : **Parrot Christian**

Date de la version :

5 Mars 2015

Sommaire

Cahier des charges

Extracteur de texte à partir d'une image

-Motivation :

Dans une société où la numérisation fait partie intégrante de la vie professionnelle, la possibilité de traiter des documents administratifs papier via un logiciel de traitement de texte offrirait un gain de temps non négligeable pour le secteur administratif.

-But du Projet :

Créer un programme permettant d'extraire le texte d'une page numérisée i.e. une image sur fond uni ne possédant que du texte.

-Périmètre :

Le fond de l'image doit être uni, les caractères à traiter aussi, et le logiciel ne traitera que des caractères ASCII, ou Unicode (restreints aux langues anglaise et française). Il ne traitera pas les images (non texte) ou les tableaux.

-Contraintes :

Certains documents sont protégés (pdf ou livres), le logiciel permettrait la recopie de ces documents, mais décline toute responsabilité en cas de violation des droits d'auteur.

-Fonctionnement :

L'utilisateur fournit l'image au logiciel, puis règle quelques paramètres (nombre de pages, couleurs de fond, contours de la page…) et lorsqu'il lance l'opération, le logiciel réécrit le texte en respectant la mise en page et les couleurs dans un fichier LibreOffice.

Analyse du problème

I. Le traitement de l'image

Voici les étapes que devrait suivre le logiciel afin de traiter une image donnée :

1. (Facultative) Préanalyse de l'image : lissage, corrections de contraste, passage en noir et blanc, détection de contours.

2. Segmentation de l'image : isoler les lignes, puis les caractères du texte.

Problème : Délimitation de la hauteur de la ligne. Une ligne de hauteur 1pixel pose problème pour la lettre i (le point pourrait définir une ligne), mais également dans des cas particuliers de ponctuation, casse (exposant, indice...) et d'accentuations.

3. Reconnaissance et sauvegarde dans la base de données des caractères rencontrés. Deux méthodes pour la reconnaissance :

- classification par caractéristiques (features) : la forme à reconnaître est caractérisée par un vecteur de valeurs numériques (feature). Cette méthode utilise des réseaux de neurones artificiels.

- comparaison avec une base de données (efficacité limitée)

Problème : Efficacité de la méthode. La méthode qui nous paraît la plus efficace est la plus difficile à mettre en œuvre.

Nous allons donc dans un premier temps considérer la méthode de comparaison. Si le temps nous le permet, nous étudierons l'autre méthode une fois l'architecture de notre programme mise en place et fonctionnelle.

4. Génération du fichier .doc contenant le texte reconnu.

II. L'interface graphique du logiciel

1. Importation du fichier à traiter

Le logiciel devra parcourir l'arborescence à la recherche du fichier à traiter. L'utilisateur pourra sélectionner plusieurs fichiers à la fois, qui seront traités successivement. Un historique permettra à l'utilisateur de retrouver tous les fichiers déjà traités par le logiciel.

2. Traitement

Une fois sélectionné, une jauge indiquera en pourcentage l'évolution du traitement du fichier. L'utilisateur aura éventuellement la possibiité en fin de traitement de choisir le logiciel (LibreOffice, OpenOffice) de traitement de texte qu'il souhaite utiliser.

3. Message d'erreur

Si l'utilisateur sélectionne un fichier dont le format n'est pas pris en charge, ou si l'image sélectionnée n'est pas reconnue comme un texte, le logiciel renverra un message d'erreur.

Analyse fonctionnelle

I. Pré-traitement de l'image.

La première étape que devra exécuter sera de prendre en entrée le fichier et de le pré-traiter pour faire en sorte qu'il corresponde aux spécifications des fonctions suivantes.

- L'utilisateur devra entrer un fichier à traiter à l'aide de l'interface utilisateur.

- La fonction "typeFichier" testera si le fichier d'entrée est bien d'un des types suivants : JPEG,PNG ou PDF. Elle renverra un booléen.

- La fonction suivante "tab2D" transformera le fichier en un tableau de pixels.

- La fonction suivante "recadrageImage" prendra l'image en entrée et la recadrera pour éviter les erreurs lors du scannage (notamment pour les pages de livres dont les bords pourraient être assombris). Cette fonction aura la nouvelle image en sortie.

- La dernière fonction "noirEtBlanc" de cette première étape prendra l'image traitée en entrée et ressortira la même image en noir et blanc.

II. Recherche des pixels pour délimitation des lignes et caractères.

La deuxième étape consiste en la séparation de l'image en lignes puis en blocs contenant chacun un caractère à reconnaître. Pour cela on recherchera les pixels noirs de l'image pour délimiter les lignes , puis les colonnes qui nous seront utiles pour délimiter les caractères contenus dans l'image.

- La première fonction "séparerLignes" va séparer l'image en lignes de textes. Cette fonction prendra l'image finale de l'étape 1 et retournera en sortie un tableau à une dimension de booléen indiquant pour chaque ligne de l'image si elle contient ou non un pixel noir, signe qu'un caractère passe sur cette ligne.

→ Cette fonction appellera la fonction "parcourirLignes" qui prendra l'image, et le numéro de la ligne lue en entrée et qui retournera un booléen en sortira selon si oui ou non la ligne contient un pixel noir.

- On appellera alors la fonction "rectifierTableau", en effet les caractères accentués, les i,j ainsi que les ponctuations telles que : ; ! ? etc... fausseraient la reconnaissance des lignes et donneraient un mauvais découpage des caractères au final.

- Ensuite pour séparer les caractères on utilise la fonction "séparerColonnes" qui prend en entrée l'image ainsi que le tableau de booléen retourné par séparerLignes, et pour chaque colonnes de chaque lignes non vide, cette fonction va vérifier si elle contient un pixel noir ou non et retourner un tableau de tableau de booléen : pour chaque ligne on aura un tableau de booléen indiquant pour chaque colonne où se trouve les pixels noirs.

→ La fonction parcourirColonne fonctionne sur le même principe que parcourirLignes, à ceci près qu'elle prend en entrée non seulement l'image et la colonne à traiter mais aussi le tableau de booléen retourné par séparerLignes. cette fonction retourne un booléen.

- La fonction créationTableauCaractère prend en entrée l'image et les deux tableaux créés par les fonctions séparerLignes et séparerColonnes et va créer un tableau d'images contenant l'image des caractères à reconnaître. Pour cela cette fonction va appeler les fonctions suivantes.

→ La fonction "découpage" va prendre l'image et les deux tableaux des fonctions séparerLignes et séparerColonnes en argument et utiliser les coordonnées données par les tableaux pour extraire des carrés de l'image, qui sont censés contenir les caractères.

→ Ensuite on va utiliser la fonction "enleverTableau" qui va enlever les éventuels tableaux qui faussent les résultats des fonctions séparerLignes et séparerColonnes : pour cela si un des carrés extrait à l'étape précédente a tous ses bords noirs, on enlève ces bords.

→ On retraite chaque carré avec séparerLignes et séparerColonnes, déjà pour retraiter les caractères dans les éventuels tableaux et ensuite pour affiner le premier découpage grossier des caractères, on réutilise ensuite la fonction découpage pour redécouper des caractères.

→ Pendant toutes ces étapes on récupères des carrés de caractères qu'il faut mettre dans un tableau et donc il faut les numéroter.

III. Reconnaissance de caractère

La troisième étape consiste en la reconnaissance des caractères, pour cela nous utilisons une méthode dite naïve consistant en la comparaison de chaque caractère reconnu avec une banque de donné que nous auront implémentés au préalable (correspondant à l'UNICODE).

- La première fonction "créerTableauMatrices" créera un tableau contenant une seule itération de chaque caractère reconnu, que l'on appellera sa "matrice", c'est cette "matrice" qui sera comparé aux caractère UNICODE. Cette fonction prendra en entrée le tableau retourné par la fonction créationTableauCaractère et retournera un tableau d'image.

- Ensuite la fonction "comparerCaractères" va prendre en entrée la liste UNICODE et le tableau retourné par "créerTableauMatrice", comparera chaque caractère à l'UNICODE et renverra un tableau dont chaque élément sera le caractère UNICODE correspondant.

- La fonction "remplacerCaractère" va prendre les tableaux retournés par comparerCaractères et créationTableauCaractère et remplacer chaque caractère de ce dernier par son caractère UNICODE correspondant.

IV. Génération du .doc.

Enfin la quatrième et dernière étape consiste en la création d'un fichier .doc à partir de ce tableau de caractères UNICODE.

Les Test Unitaires

I. Pré-traitement de l'image.

- Le test de la fonction "typefichier" sera assez simple, il suffira de lui donner un des fichiers JPEG, PNG ou PDF et voir si le résultat renvoyé correspond bien au type de fichier donné. On pourra aussi vérifier qu'elle renvoie un code d'erreur si l'utilisateur ne lui donne pas un fichier correspondant à un des formats cités ci-desus.

- La fonction tab2D pourra être testée en envoyant en entrée des images soit entièrement noires soit entièrement blanche, puis des images rayées (en ligne ou/et en colonne), car la localistaion exacte d'un pixel et donc l'exactitude de la réponse de la fonction sera difficile à vérifier si l'on envoie une image au hasard en donnée.Ainsi on pourra au moins vérifier que le nombre et l'ordre des bandes blanche et noire sera respectée. Il faudra aussi essayer de vérifier que le code fonctionne pour la première et dernière ligne/colonne, afin de vérifier par exemple que les bornes du code soient correctes par exemple.

- Pour tester la fonction "recadrageImage", on pourra envoyer une image dont les bords sont noirs, puis si elle fonctionne, avec des bords constitués de niveaux de gris de plus en plus clairs. Enfin, on pourra aussi tester avec une image contenant des caractères très près de tous les bords et/ou compris entre des bandes noires, c'est à dire une bande sur un bord mais coupée par exemple en son milieu avec un mot à l'intérieur. Imaginons qu'un "L" se trouve au milieu d'une bande verticale et collé au bord, la fonction retirera-t-elle le L de l'image ?

- La fonction "noiretblanc " pourra être testée en utilisant différents niveau de gris pour les caractères et le fond. Mais tant qu'on ne sait pas exactement comment la fonction procède, des tests spécifiques à la fonction seront difficiles à prévoir. On peut tout de même vérifier que les caractères ressortent bien avec un fond blanc sur des images où les caractères sont de plus en plus difficiles à distinguer.

Il faudra enfin vérifier que le module fonctionne entièrement après que chaque fonction ait été testée, i.e qu'en envoyant une image type utilisée pour le test de la fonction "noiretblanc" en donnée dans le programme, toutes les étapes se sont correctement réalisées pour les fonctions citées dans la première partie

II. Recherche des pixels pour délimitation des lignes et caractères.

- Test de la fonction "séparerLignes" : une image comprenant une colonne de lettres (« k » ou « i »par exemple).

- On testera la fonction "parcourirLigne" avec une image comprenant des pixels noirs et blancs, on vérifie que les indices de lignes sont corrects, et on remplace un pixel blanc situé sur une ligne vide par un pixel noir. On vérifie que la fonction retourne "VRAI" pour la présence d'un pixel noir sur cette ligne.

Problème qu'il faudra ne pas oublier de vérifier : Comment séparer les lignes sur image comprenant une colonne de caractères alignés ("l" par exemple), tels que chaque ligne comporte un pixel noir, ou une image correspondant à un caractère géant ? .

- Test de la fonction "rectifierTableau" : on peut tester cette fonction avec une image comprenant une ligne de « i » uniquement, puis une ligne de chaque caractère problématique, puis une ligne mélangeant majuscules et minuscules.
Enfin tester la fonction sur une ligne de texte regroupant tous les caractères à problème.

- Test de la fonction "séparerColonnes" : une image comprenant une ligne de lettres ("b" par exemple).

- On teste la fonction "parcourirColonne" avec une image comprenant des pixels noirs et blancs, on vérifie que les indices de colonnes sont corrects, et on remplace un pixel blanc situé sur une colonne vide par un pixel noir. On vérifie que le nouveau pixel noir est comptabilisé.

- Test de la fonction "creationTableauCaractère" avec une image comprenant un caractère quelconque (« k » par exemple), puis 4 caractères disposés sur 2 lignes.

III. Reconnaissance de caractère

- La fonction "créerTableauMatrice"devra être testée avec une image contenant un texte basique, puis avec des images où les caractères seront les mêmes mais légèrement différents (par exemple par leur taille, s'ils sont en gras ou en italique etc...), pour voir si la fonction est capable de reconnaître que les caractères sont les mêmes, ou vérifier plus tard que la reconnaissance de ces caractères semblables lors de la comparaison engendreront le même caractère reconnu.

- Pour la fonction "comparerCaractères" les tests unitaires devront être nombreux. Il faudra faire un test pour au moins chaque lettre de l'alphabet afin de voir si le code arrive à reconnaître le caractère en comparant avec la base de donnée. Il faudra ensuite envisager de faire un test avec plusieurs lettres "a" par exemple, de différentes tailles, polices, en gras ou en italiques sur une seule image.
Les tests unitaires sur les caractères nous permettront notamment d'améliorer notre approche sur l'implémentation du code après les premiers essais.

- La fonction "remplacerCaractère" sera elle aussi testée avec plusieurs caractères différents, notamment des caractères spéciaux qui pourraient poser des problèmes.

Conclusion :

Pour conclure, nous avons donc ici l'analyse détaillée du problème. Notre principale difficulté va sûrement être de choisir l'algorithme de reconnaissance de texte dans la mesure où celui utilisé ici est plutôt simple mais n'est pas optimisé. L'utilisation des Features va être un moyen que nous allons étudier afin de décider quel algorithme nous utiliserons avant de commencer le code en lui même.
De plus, il y aura des considérations graphiques a ne pas négliger avec l'interface du logiciel que l'on prendras en compte prochainement.
Ce premier rapport et ses objectifs peuvent bien sur évoluer si le temps nous le permet, notamment vis-à-vis des types de fichiers acceptés en entrée du logiciel mais aussi de la considération de la mise en page lors de l'écriture du fichier « .doc ».