Pictwriters

Soutenance 1
Projet OCR

Archer Stéphane, Eychenne Mélanie,
Lorenzini Ugo et Petrov Daniel

24 Octobre 2013

Table des matières

1 Introduction

Après avoir fait l'implémentation d'un jeu vidéo en SUP, pour notre année de SPE à EPITA, nous devons réaliser un **O**ptical **C**haracter **R**ecognition (Reconnaissance Optique de Caractères en français) que l'on nommera : O.C.R. Ce projet est codé en Objective Caml (aussi nommé OCaml) et est développé sous FreeBSD.

1.1 Qu'est ce qu'un O.C.R ?

Un O.C.R est un logiciel permettant le traitement d'un texte présent dans une image telle que : une photographie, un scan d'un livre, etc. Après ce traitement, l'O.C.R permet d'obtenir un document avec le texte brut modifiable.

Pour ce faire, nous devons implémenter des fonctions permettant de reconnaitre un caractère même s'il est incliné ou composé de couleurs différentes, etc.

De plus, on implémentera également un dictionnaire qui permettra d'analyser le mot retranscrit pour une diminution des risques de fautes de traitement d'un caractère.

1.2 Principe d'utilisation

L'utilisateur souhaite récupérer un fichier contenant le texte brut d'une image ou d'une numérisation d'une page. Ainsi, lorsqu'il va scanner son document, notre O.C.R. va lui permettre de transformer celui ci en un texte brut grâce à la liste des symboles

2 Présentation des membres de la Pictwriters Team

Nous avons choisi comme nom de groupe la Pictwriters Team car nous allons permettre aux utilisateurs de notre logiciel de transformer leurs images contenant du texte ("Pict = picture) en un document contenant ce texte pouvant être modifié.

2.1 Stéphane

Bonjour, je m'appelle Stéphane ARCHER et je suis en SPE à EPITA. L'informatique a toujours été ma passion. De plus, ce projet est l'occasion pour moi de vérifier mes compétences de gestion d'un projet car on m'a nommé chef de projet. En début d'année je ne connaissais ni Daniel ni Ugo mais j'ai trouvé deux coéquipiers très sympathiques et sérieux tout comme Mélanie. Chaque membre du groupe a su apporter sa contribution dans la joie et la bonne humeur. J'attends essentiellement de ce projet de pouvoir perfectionner ma programmation, apprendre à structurer un projet, répartir les tâches et vérifier que tous s'imbrique. Ce travail d'équipe devrait permettre à chacun de développer ses talents. On est tous pressés de pouvoir utiliser notre O.C.R et de le partager.

2.2 Mélanie

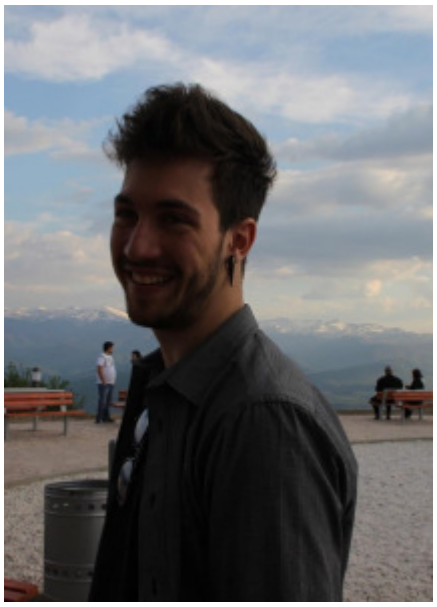
Seule représentante de la gente féminine du groupe, j'espère apporter un petit plus du côté de l'interface graphique dont on s'occupera pour la dernière soutenance. Ce projet est, selon moi, très intéressant car il m'est arrivé de nombreuses fois de me retrouver devant un scan d'une feuille sur internet que j'aurais aimé ajouter dans mon document avec quelques modifications. Mais, à chaque fois, j'étais obligée de recopier ce dont j'avais besoin. Grâce à ce projet, j'aurais "mon propre" O.C.R que je pourrai utiliser quand bon me semble !

2.3 Ugo

Moi c'est Ugo, j'ai commencé l'informatique avec l'ordinateur familial et je me suis tout de suite intéressé aux jeux vidéo. Plus tard j'ai eu envie de comprendre

comment les choses marchaient, en particulier dans un ordinateur, et ce comment étaient conçus les programmes et jeux que j'utilisais. J'ai fait un peu de C quand j'étais encore qu'un débutant en informatique mais j'ai vite arrêté car faire des jeux en mode console sous Windows n'était pas si facile que ça et pas non plus très passionnant. L'année dernière j'ai été très emballé par le projet de SUP et plutôt satisfait du résultat. Cette année, le projet d'OCR à l'air d'être un projet intéressant, en partie la partie sur le réseau de neurones car je me suis toujours demandé comment fonctionnait une "intelligence artificielle" même si je suis conscient que cela sera dur et que je devrai acquérir de nouvelles compétences pour y arriver.

2.4 Daniel



Je me nomme Daniel PETROV et je suis un étudiant à l'EPITA. On peut dire que dès ma plus petite enfance j'ai été baigné dans le monde des ordinateurs. A mes trois ans j'étais déjà planté devant un ordinateur et j'avais soif d'apprendre et de m'amuser grâce aux logiciels ADIBOU. Et d'ailleurs, c'était l'unique façon pour mes parents de calmer mes pleurs et mon mauvais tempérament (et surtout de donner des moments de répits aux voisins qui m'entendaient crier et pleurer à tue-tête et je peux vous dire que j'avais de la voix). J'ai toujours été intrigué par les ordinateurs, par leur fonctionnement, par leur utilisation et tout ce qu'on peut faire avec. Ils m'ouvraient de nouveaux horizons qui m'étaient encore inconnus. J'y passais des journées et des nuits entières et combien de fois je me suis fait gronder par mes parents car j'avais complètement mis en pièces mon ordinateur pour justement comprendre ses composantes. Peu de temps après, j'ai commencé à m'intéresser au multimédia et tout ce qui en découle ; le son, les images, le design, etc. Arrivé en seconde année à L'EPITA, j'ai vu que le projet de l'année était de faire un OCR (Optical Character Recognition) et j'en ai été véritablement ravi.

Cela me donnerait la possibilité de travailler sur cette problématique et surtout de réfléchir et réaliser le traitement de l'image du projet OCR.

3 Organisation

3.1 Répartition des tâches

Nous avons rapidement effectué une répartition des tâches afin de pouvoir commencer notre projet le plus tôt possible. Voici un tableau montrant quelle personne s'est le plus occupé de telle ou telle partie.

Soutenance 1	
Pré-traitement de l'image	Daniel
Passage en niveau de gris	Mélanie
Rotation de l'image	Stéphane
Détection des zones de textes	Ugo
Site web	Mélanie
Rapport de la soutenance	Mélanie

3.2 Organisation générale

Après avoir fait la répartition des tâches nous avons fait un dépôt de notre code source sur GIT. Nous sommes restés en contact permanent grâce à Facebook mais aussi Skype et nous nous sommes donnés quelques rendez-vous à EPITA pour voir la progression du projet et apporter de l'aide aux personnes en ayant besoin.

Il faut l'avouer, il a été difficile pour nous de nous voir car certaines personnes de l'équipe habitent loin d'EPITA. Ainsi, par un souci de rentabilité de temps, nous avons décidé de nous donner rendez-vous dans une salle à EPITA que dans la semaine à la fin des cours (le plus souvent le vendredi car c'est le jour où nous terminons le plus tôt). Mais de toutes façons, nous savons que nous pouvions avoir un contact permanent grâce à nos portables et aux réseaux sociaux comme dit précédemment.

4 Pré-traitement de l'image

4.1 Mise en bouche : Vocabulaire

Voici des termes important dont nous allons vous parler dans cette partie du rapport :

Une image : On peut comparer une image à une matrice composée de pixels. En effet, pour modifier une image, il suffit de parcourir chaque pixel de l'image grâce à deux boucles for permettant d'accéder à la position (w,h) avec w : la largeur et h : la hauteur de l'image (comme pour accéder à une valeur d'une matrice).

Un pixel : La couleur d'un pixel peut être modifié grâce à ses composantes : r, g et b (r représentant le rouge, g représentant le vert et b représentant le bleu). On peut également modifier l'opacité de l'image grâce à rgba (lorsque a = 1 l'image est opaque).

Le bruit d'une image : Le bruit représente tout ce dont on ne veut pas garder dans l'image. Nous, nous souhaitons récupérer les caractères présents dans une image (et seulement les caractères).

4.2 But de l'étape

Pour faire un O.C.R il faut, tout d'abord, commencer par transformer l'image d'origine pour que la reconnaissance des caractères soit plus simple. En effet, notre but est de faire ressortir le tracé des caractères.

4.3 Principe

Pour faire ressortir le tracé des caractères, il faut éliminer (et même plus : éradiquer ! !) les bruits de fond et amplifier les marques des caractères. Pour cela, il existe plusieurs méthodes d'implémentation.

4.3.1 Chargement de l'image

Avant de pouvoir traiter notre image, il faut qu'on la charge. Pour cela nous avons utilisé la bibliothèque OCamlSDL car elle permet facilement de convertir une image en une matrice de pixels grâce à cette fonction :

```
let image = Sdlloader.load_image filename;;
```

4.3.2 Passage en niveau de gris

Tout d'abord, nous avons passé notre image en niveau de gris. La technique est simple : il suffit d'appliquer à chaque pixel de l'image la formule de luminosité moyenne suivante : 0.3 * rouge + 0.59 * vert + 0.11 * bleu.

Il existe également une autre technique qui est de faire la somme de chaque composante d'un pixel (c'est à dire r, g et b dont les valeurs sont comprises entre 0 et 255) et de la diviser par trois (ce qui donne un nouveau nombre compris entre 0 et 255). Mais elle ne prend pas en compte que notre œil n'a pas la même sensibilité pour chaque couleur. C'est pour cela qu'il vaut mieux éviter de mettre les mêmes valeurs pour les trois composantes d'un pixel d'une image pour la passer en niveau de gris.

$$Gris = \frac{(Rouge + Vert + Bleu)}{3}$$

4.3.3 Binarisation

Puis nous avons passé l'image en noir et blanc. Il existe différentes façons pour implémenter cette fonction. Nous avons décidé de faire la plus efficace mais également la plus simple. Il suffit de partir de l'image passée en niveau de gris et en fonction d'un certain seuil de luminosité on choisit de passer le pixel en noir "pur" ou en blanc "pur". Nous avons choisit comme valeur de seuil : 220. Ainsi, pour chaque pixel rencontré lors du parcours de l'image si le pixel courant et supérieur à ce seuil on le met en blanc sinon on le passe en noir.

4.3.4 Élimination du bruit de l'image

Et enfin, nous avons détecté puis éliminé le bruit de l'image c'est à dire que nous avons éliminé les éléments inutiles de l'image pour faire ressortir les caractères. Pour cela, nous avons implémenté une fonction qui lorsqu'elle parcourt chaque pixel d'une image, si ce pixel est noir et entouré de pixels blancs alors il devient blanc. Plus concrètement, la fonction parcourt l'image : si le pixel est noir (à une valeur de 0) elle vérifie si les pixels qui l'entourent sont blancs (ont une valeur de 255). Si c'est le cas, elle modifie le pixel noir et le met en blanc, sinon elle passe au pixel suivant.

Nous améliorerons cette fonction pour notre prochaine soutenance car si le pixel est noir et que les pixels qui l'entourent ne sont pas tous blanc alors il est inutile de tester le pixel suivant mais il faut aller sur le suivant du pixel suivant du pixel courant (vous comprenez ? ;))

4.3.5 Bilan des étapes du pré-traitement de l'image

Voici l'image sans modification :

Tout d'abord, on fait le passage en niveau de gris de l'image :

Puis on passe l'image en noir et blanc :

5 Rotation de l'image

5.1 Son but

Quand on scanne une page, celle ci n'est pas toujours "droite" ou l'écriture n'est pas forcément mis à l'horizontale.

Ainsi la reconnaissance des caractères est difficile.

C'est pour cela qu'en pré-traitement nous avons implémenté une fonction permettant une rotation de l'image.

Pour cela, il faut détecter l'angle d'inclinaison des lignes de texte puis se servir des informations récupérées afin de faire la rotation de l'image.

5.2 Son principe

Pour faire une rotation d'image il faut tout d'abord trouver l'angle d'inclinaison du texte et utiliser cet angle de rotation pour mettre le texte "droit".

5.2.1 Trouver l'angle d'inclinaison

La partie la plus difficile de la rotation est sans conteste de trouver l'angle d'inclinaison.

Pour cela nous avons calculé le nombre de pixels noirs sur une ligne de pixels de l'image, fait la moyenne du nombre de pixels par ligne et enfin, calculé la moyenne générale du nombre de pixels par lignes. Par la suite, nous avons calculé la variance du nombre de pixels par ligne. Plus la variance est forte plus il y a de pixels noirs sur une même ligne (respectivement de pixels blancs) et donc plus l'image est droite.

Ainsi, tant que la valeur de la variance est supérieure à la précédente valeur, la fonction refait les calculs des moyennes et de la variance après avoir effectué une rotation de l'image vers la droite de 1° (si la valeur de la variance est inférieure, elle fera une rotation vers la gauche d'1°).

Après avoir trouvé l'angle d'inclinaison, nous effectuons la rotation de l'image avec l'angle obtenu grâce à la fonction expliquée dans le paragraphe suivant.

5.2.2 Utilisation de l'angle trouvé

Après avoir trouvé l'angle d'inclinaison, il nous a fallut l'utiliser pour faire tourner notre image.

Pour notre premier essai, nous avons utilisé ces formules pour le calcule des nouvelles coordonnées :

x' = x*cos(angle) – y*sin(angle)
y' = x*sin(angle) +y*cos(angle)

Le problème de ces deux équations est qu'après la transformation, l'image était coupé sur les bords et donc il y avait une grande perte de texte.

Pour résoudre ce problème, nous avons fait des recherches et nous avons trouvé notre erreur fatale : notre rotation débutait sur la coordonnée (0,0) de la matrice (le premier pixel en haut à gauche de la matrice).

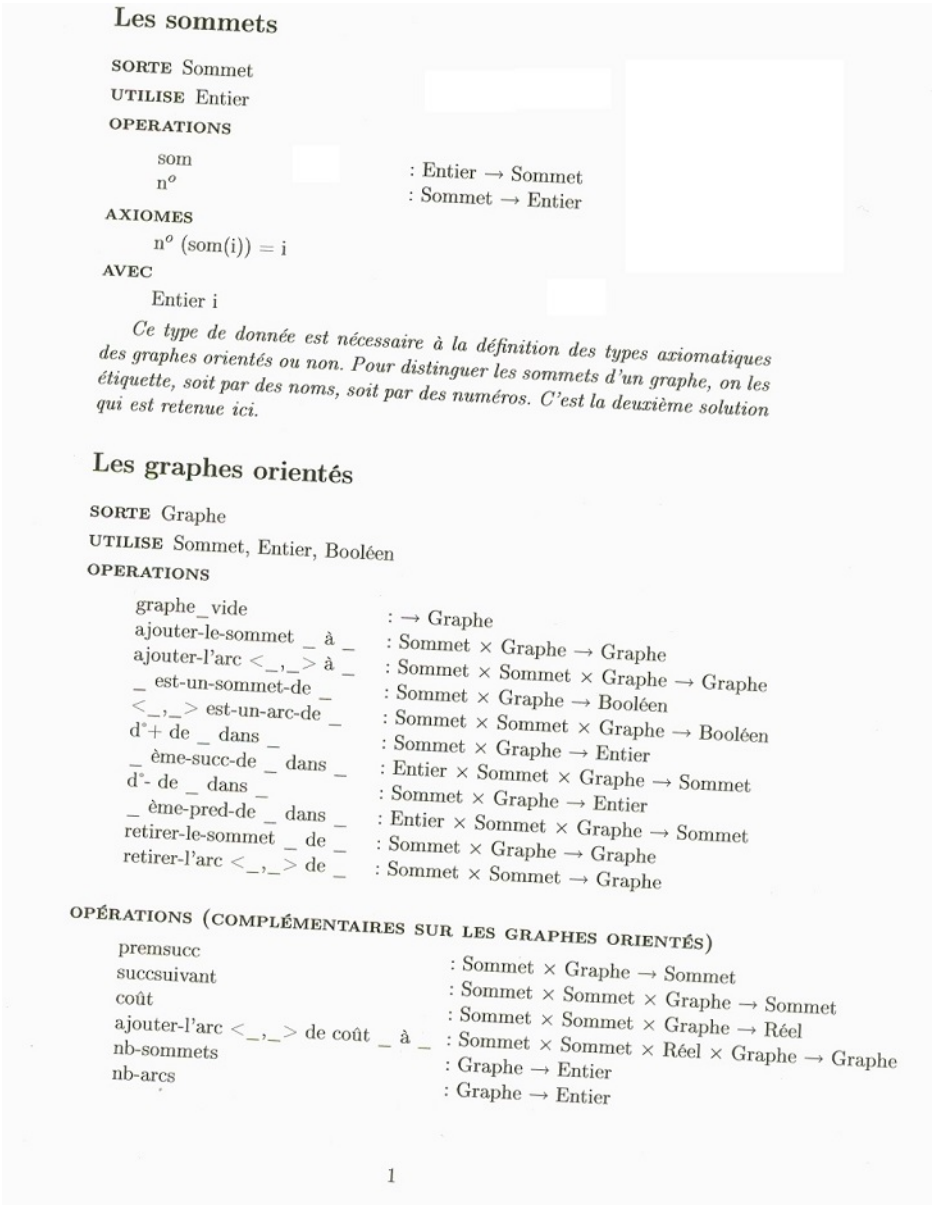
Or il fallait débuter par le pixel situé au centre de l'image : en effet, il faut prendre le pixel du centre comme centre de la rotation.

Ainsi les bonnes formules a utiliser sont :

x' = centre + (x – centre) * cos(angle) – (y – centre) * sin(angle)
y' = centre + (x – centre) * sin(angle) + (y – centre) * cos(angle)

5.2.3 Application de la rotation

Voici un exemple de ce que nous avons pu obtenir grâce à la fonction de rotation d'une image :



Les sommets

SORTE Sommet
UTILISE Entier
OPERATIONS

som	: Entier → Sommet
n^o	: Sommet → Entier

AXIOMES

n^o (som(i)) = i

AVEC

Entier i

Ce type de donnée est nécessaire à la définition des types axiomatiques des graphes orientés ou non. Pour distinguer les sommets d'un graphe, on les étiquette, soit par des noms, soit par des numéros. C'est la deuxième solution qui est retenue ici.

Les graphes orientés

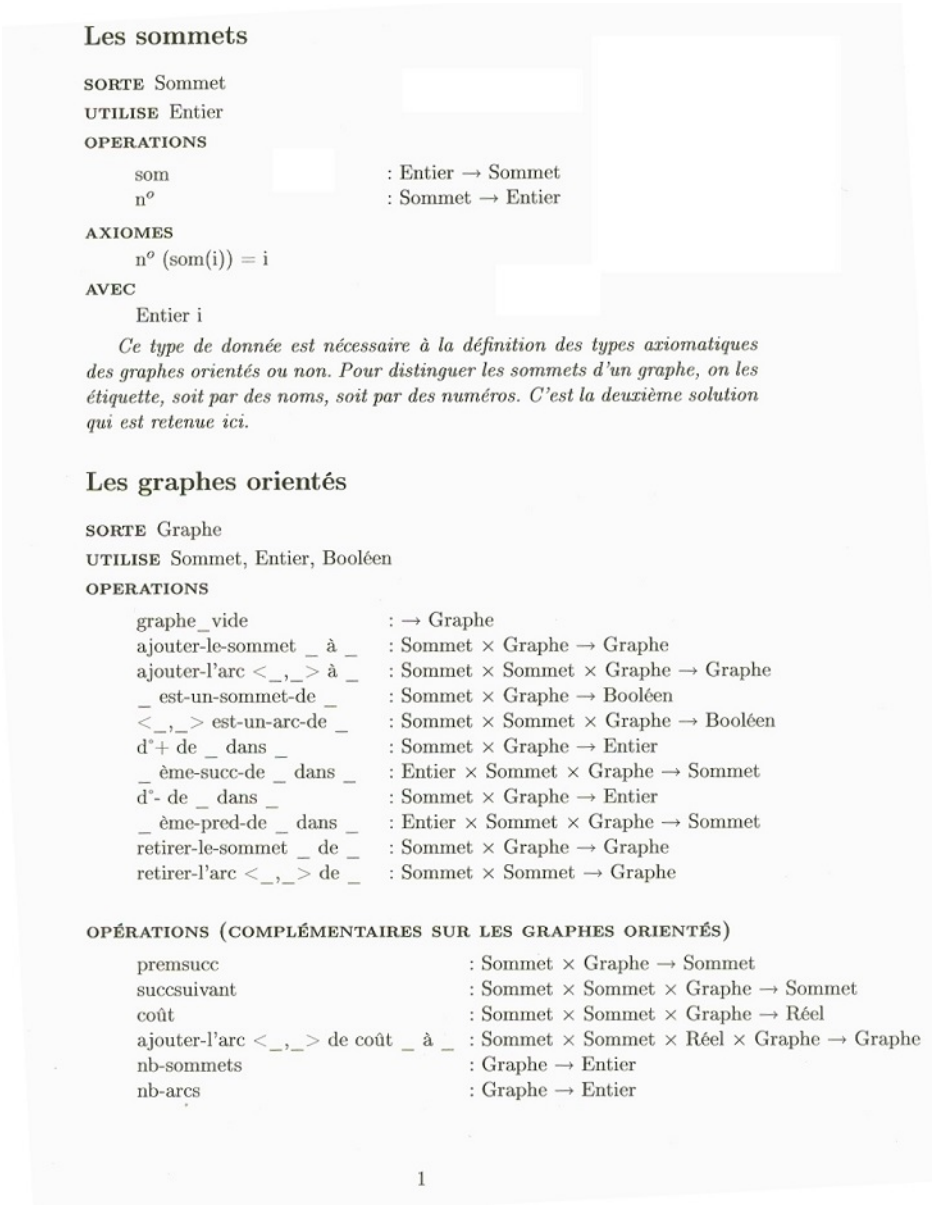
SORTE Graphe
UTILISE Sommet, Entier, Booléen
OPERATIONS

graphe_vide	: → Graphe
ajouter-le-sommet _ à _	: Sommet × Graphe → Graphe
ajouter-l'arc <_,_> à _	: Sommet × Sommet × Graphe → Graphe
_ est-un-sommet-de _	: Sommet × Graphe → Booléen
<_,_> est-un-arc-de _	: Sommet × Sommet × Graphe → Booléen
$d^\circ+$ de _ dans _	: Sommet × Graphe → Entier
_ ème-succ-de _ dans _	: Entier × Sommet × Graphe → Sommet
$d^\circ-$ de _ dans _	: Sommet × Graphe → Entier
_ ème-pred-de _ dans _	: Entier × Sommet × Graphe → Sommet
retirer-le-sommet _ de _	: Sommet × Graphe → Graphe
retirer-l'arc <_,_> de _	: Sommet × Sommet → Graphe

OPÉRATIONS (COMPLÉMENTAIRES SUR LES GRAPHES ORIENTÉS)

premsucc	: Sommet × Graphe → Sommet
succsuivant	: Sommet × Sommet × Graphe → Sommet
coût	: Sommet × Sommet × Graphe → Réel
ajouter-l'arc <_,_> de coût _ à _	: Sommet × Sommet × Réel × Graphe → Graphe
nb-sommets	: Graphe → Entier
nb-arcs	: Graphe → Entier

1



Les sommets

SORTE Sommet
UTILISE Entier
OPERATIONS

som	: Entier → Sommet
n^o	: Sommet → Entier

AXIOMES
n^o (som(i)) = i
AVEC
Entier i

Ce type de donnée est nécessaire à la définition des types axiomatiques des graphes orientés ou non. Pour distinguer les sommets d'un graphe, on les étiquette, soit par des noms, soit par des numéros. C'est la deuxième solution qui est retenue ici.

Les graphes orientés

SORTE Graphe
UTILISE Sommet, Entier, Booléen
OPERATIONS

graphe_vide	: → Graphe
ajouter-le-sommet _ à _	: Sommet × Graphe → Graphe
ajouter-l'arc <_,_> à _	: Sommet × Sommet × Graphe → Graphe
_ est-un-sommet-de _	: Sommet × Graphe → Booléen
<_,_> est-un-arc-de _	: Sommet × Sommet × Graphe → Booléen
d°+ de _ dans _	: Sommet × Graphe → Entier
_ ème-succ-de _ dans _	: Entier × Sommet × Graphe → Sommet
d°- de _ dans _	: Sommet × Graphe → Entier
_ ème-pred-de _ dans _	: Entier × Sommet × Graphe → Sommet
retirer-le-sommet _ de _	: Sommet × Graphe → Graphe
retirer-l'arc <_,_> de _	: Sommet × Sommet → Graphe

OPÉRATIONS (COMPLÉMENTAIRES SUR LES GRAPHES ORIENTÉS)

premsucc	: Sommet × Graphe → Sommet
succsuivant	: Sommet × Sommet × Graphe → Sommet
coût	: Sommet × Sommet × Graphe → Réel
ajouter-l'arc <_,_> de coût _ à _	: Sommet × Sommet × Réel × Graphe → Graphe
nb-sommets	: Graphe → Entier
nb-arcs	: Graphe → Entier

1

6 Détection des zones de textes

Après avoir "nettoyé" au maximum l'image de toutes ses impuretés, il faut s'attaquer à la localisation des lignes et à la détection des bornes supérieure et inférieure de chaque ligne.

6.1 But

Le but de cette détection des zones de texte est de permettre de localiser chaque caractère d'une phrase afin de les retranscrire dans le document texte final.

6.2 Principe de l'algorithme X/Y cut

Afin de localiser les zones de texte, l'algorithme appelé X/Y cut peut être utilisé. Cet algorithme permet de découper le fichier en plusieurs zones en commençant tout d'abord par les plus grandes surfaces puis en restreignant le découpage dans chaque bloc trouvé jusqu'à ce qu'il n'y ait plus de possibilité de faire d'autres découpages.

Ainsi l'algorithme X/Y cut parcourt l'image d'abord horizontalement et détecte les changements de paragraphes en hauteur (c'est à dire lorsqu'il y a plusieurs lignes composées entièrement de pixels blancs les uns à la suite des autres).

Puis sur chaque sous bloc détecté on applique l'algorithme verticalement pour détecter les paragraphes "parallèles" sur une même hauteur (c'est à dire pour localiser les colonnes de texte).

On recommence tant que l'on continue à détecter des séparations. A la fin on se retrouve avec une liste de blocs et de sous blocs qui correspondent au découpage de l'image.

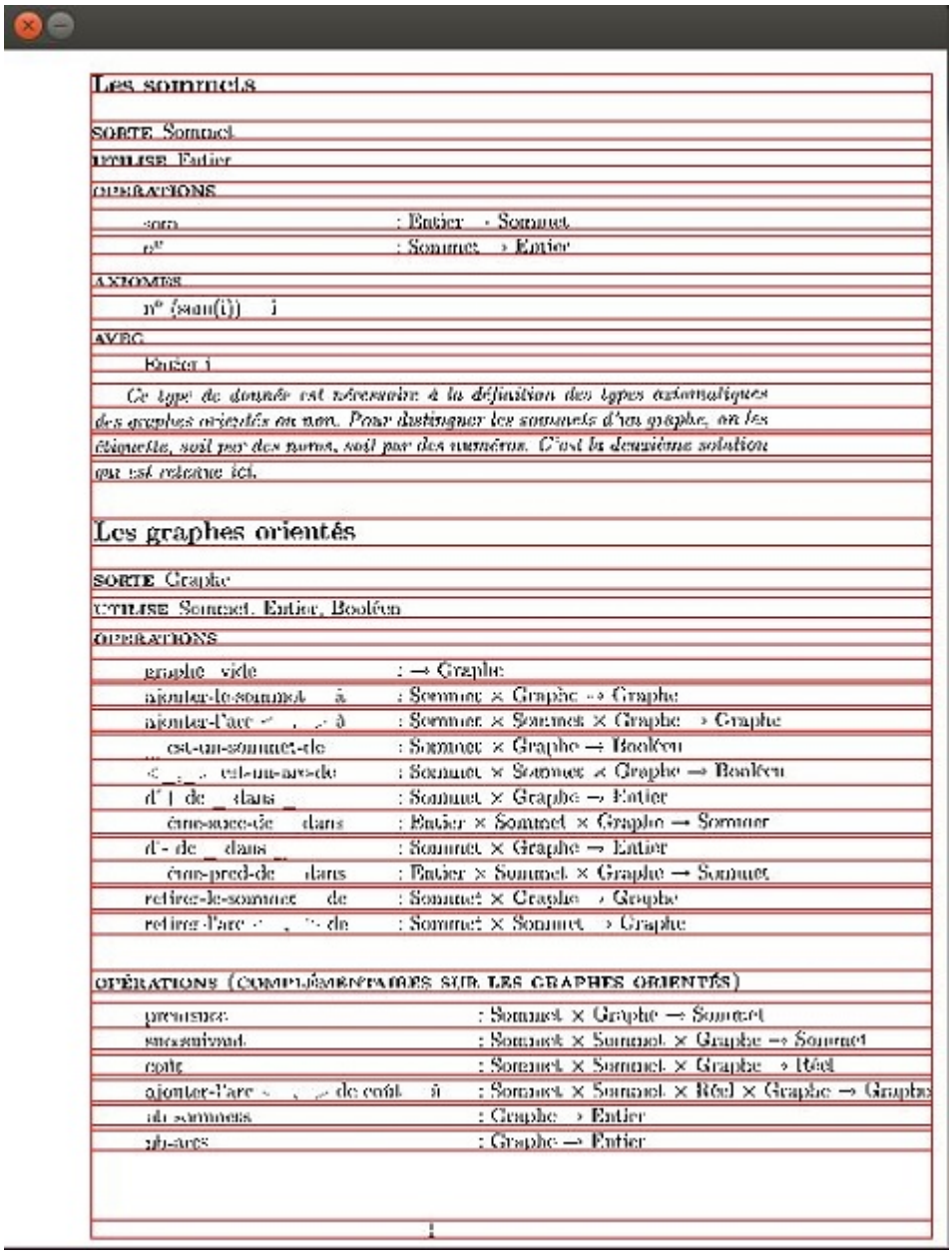
Pour bien faire il faudrait placer les blocs et sous blocs dans un arbre général avec pour racine l'image de départ (sans les pixels blancs autour) et où chaque fils serait le sous bloc détecté par l'algorithme. Cette technique permettrait que les paragraphes soient tous en feuille, ce qui sera plus pratique pour la détection des lignes mais aussi pour savoir dans quel ordre la lecture devra se faire pour retranscrire le texte.

Et pour la détection des lignes à chaque fois que l'on a une ligne de pixel entièrement blanche c'est qu'il y a une nouvelle ligne donc on stocke les coordonnées dans une liste.

6.3 Principe que nous avons implémenté

Nous n'avons pas réussi à implémenter l'algorithme X/Y cut. Mais nous avons utilisé un autre principe et si les résultats de celui-ci sont satisfaisants nous le garderons sinon nous envisagerons de le faire pour la prochaine soutenance.

Notre principe est donc de détecter les lignes et les blocs "en même temps" puis de diviser les blocs trouvés s'ils sont trop grands afin de faire de nouvelles lignes. Les résultats sont plutôt bons comme vous pouvez le voir sur cette image :



Les sommets

SORTE Sommet

UTILISE Entier

OPÉRATIONS

som	: Entier → Sommet
n°	: Sommet → Entier

AXIOMES

n° (som(i)) = i

AVEC

Entier i

Ce type de donnée est nécessaire à la définition des types axiomatiques des graphes orientés en nom. Pour distinguer les sommets d'un graphe, on les étiquette, soit par des noms, soit par des numéros. C'est la deuxième solution qui est retenue ici.

Les graphes orientés

SORTE Graphe

UTILISE Sommet, Entier, Booléen

OPÉRATIONS

graphe_vide	: → Graphe
ajouter-le-sommet_à	: Sommet × Graphe → Graphe
ajouter-l'arc_,_à	: Sommet × Sommet × Graphe → Graphe
_ est-un-sommet-de	: Sommet × Graphe → Booléen
<_,_> est-un-arc-de	: Sommet × Sommet × Graphe → Booléen
d°_de_dans_	: Sommet × Graphe → Entier
ème-succ-de_dans	: Entier × Sommet × Graphe → Sommet
d-_de_dans_	: Sommet × Graphe → Entier
ème-pred-de_dans	: Entier × Sommet × Graphe → Sommet
retirer-le-sommet_de	: Sommet × Graphe → Graphe
retirer-l'arc_,_de	: Sommet × Sommet × Graphe → Graphe

OPÉRATIONS (COMPLÉMENTAIRES SUR LES GRAPHES ORIENTÉS)

precesseur	: Sommet × Graphe → Sommet
successeur	: Sommet × Sommet × Graphe → Sommet
cout	: Sommet × Sommet × Graphe → Réel
ajouter-l'arc_,_de coût_à	: Sommet × Sommet × Réel × Graphe → Graphe
nb-sommets	: Graphe → Entier
nb-arcs	: Graphe → Entier

7 Notre site internet

`http://pictwriters.wordpress.com/`

Voici notre site internet !!

Grâce à celui ci vous pouvez dès à présent accéder au PDF de notre rapport de première soutenance ainsi qu'à la présentation de chacun des membres de la Pictwriters Team !! Nous avons aussi posté différents articles indiquant certaines informations à propos de notre projet. Nous y avons également intégré un lien permettant de voir où nous en sommes dans notre code source (et même de le télécharger) grâce à GIT.

De plus, vous pouvez aller voir les sites internet des éléments utilisés pour faire notre projet grâce à l'onglet liens.

Nous avons également ajouté des "widgets" :

- un calendrier : lorsque l'on clique sur un jour spécial il affiche les articles publiés ce jour même ;
- une liste des articles récents ;

- un décompteur : il permet de calculer le nombre de jours qui sépare le jour actuel à celui d'un événement (pour nous : la soutenance) ;
- un compteur : il compte le nombre de visite sur notre page web

7.1 Avancé d'un site internet en langage HTML et CSS

Utiliser wordpress est bien beau mais apprendre à faire son propre site internet est aussi très intéressant. En effet, wordpress est assez facile d'utilisation lorsqu'on commence à faire un site internet alors qu'utiliser le langage HTML et CSS et beaucoup plus complexe et moins intuitif. Cependant, il est véritablement très intéressant de faire son site avec HTML et CSS.

C'est pour cela que nous avons commencé à faire un site internet en langage HTML et CSS. Nous avons trouvé une bonne documentation sur le site `http://fr.openclassrooms.com/` (ex. Site du Zéro). Nous espérons acquérir, grâce à ce site, de nombreuses connaissances en code HTML et CSS.

8 Ce qu'il nous reste à faire pour la prochaine soutenance

8.1 Améliorations des éléments implémentés de la première soutenance

- Correction et amélioration du pré-traitement de l'image
- Mise à jour du site web

8.2 Ajouts à faire pour la prochaine soutenance

8.2.1 Implémentation du réseau de neurones

La partie la plus importante du projet, le réseau de neurones a également l'air d'être le plus difficile à implémenté. Il va falloir que nous choisissions la représentation la plus adaptée à notre projet mais aussi la plus précise dans la reconnaissance des caractères.

8.2.2 Interface graphique

Pour l'utilisateur, nous devons faire une interface graphique simple et intuitive qui sera composée d'une zone de texte permettant à l'utilisateur de modifier le texte du document numérisé et d'un espace permettant de visualiser le document original afin de pouvoir comparer la similarité des textes.

8.2.3 Intégration d'un correcteur orthographique dans le champ de texte produit

Après la retranscription du texte, il peut y avoir des erreurs de reconnaissance d'un caractère. L'intégration d'un correcteur va permettre d'éliminer ces erreurs et de s'occuper des caractères non reconnus. Nous utiliserons la bibliothèque d'interface graphique GTK.

8.2.4 Documentation

Pour la dernière soutenance, nous devons fournir en plus du rapport de projet et d'un plan de soutenance, un dossier d'exploitation (manuel d'installation + manuel d'utilisation) ainsi que le projet.

9 Conclusion

Nous voici partis dans une nouvelle aventure ! ! Nous sommes heureux de notre petite avancé : c'est un petit pas pour l'Homme mais un grand pas pour l'humanité ! ! ! (bon peut être pas encore ;)). Nous sommes tout de même fiers de vous avoir présenté notre travail. Nous sommes conscients qu'il nous reste encore une grande partie du projet à réaliser dont le réseau de neurones qui à l'air d'être assez compliqué à implémenté d'après nos recherches.