

Projet
d'Approfondissement
et d'Ouverture

ASI4 2014-2015

Manon ANSART
Anthony COURTIN

RECONNAISSANCE DE CARACTÈRES ET DÉTECTION DE LANGUE *RAPPORT DE PROJET*

Table des matières

1	Introduction	2
2	Compréhension des outils	3
2.1	Ocropus	3
2.1.1	Généralités	3
2.1.2	Fonctionnement	3
2.1.3	Résultats	4
2.2	Cld2	5
2.2.1	Généralités	5
2.2.2	Nettoyage des données	6
2.2.3	Analyse statistique	6
2.2.4	Résultats	6
3	Conception de l'application	8
3.1	L'application Android	8
3.1.1	Conception	8
3.1.2	Installation	9
3.1.3	Fonctionnement	9
3.1.4	Tests	9
3.1.5	Problèmes rencontrés	10
3.2	La communication avec le serveur	11
3.2.1	Connexion au serveur	11
3.2.2	Installation des outils	11
3.2.3	PHP	12
3.2.4	Problèmes rencontrés	12
4	Conclusion	13

Chapitre 1

Introduction

Le but initial de ce projet était de travailler avec la librairie Olena d'Epita afin de détecter des photos dans des images de documents. Après le téléchargement et la compilation de la librairie, nous avons tenté de la tester et de rechercher les fonctions utiles pour l'application que nous voulions en faire. Nous avons découvert que la librairie était bien plus complexe que ce que nous pensions, et qu'elle ne fournissait pas de fonction permettant de trouver directement une photo dans une image. Nous avons donc décidé de choisir un autre sujet, toujours dans le cadre de la reconnaissance d'image de document. Pendant ce Projet d'Approfondissement et d'Ouverture, nous allons utiliser la bibliothèque ocropus pour reconnaître les caractères à partir d'une image de documents puis en détecter la langue grâce au module de détection de langue cld2 développé par google.

Chapitre 2

Compréhension des outils

2.1 Ocropus

2.1.1 Généralités

OCropus est un logiciel libre de reconnaissance optique de caractères (ROC) ou optical character recognition (OCR) en anglais. Il a pour but de pouvoir être utilisé à la fois par des chercheurs et des entreprises[?], c'est pourquoi il a été développé sous licence Apache 2, qui permet une utilisation commerciale facile.

OCropus permet d'utiliser plusieurs modules de reconnaissance de texte. En particulier, il implémente Tesseract, qui est considéré comme un des modules les plus exactes dans le domaine des logiciels libres. Ces performances se rapprochent de celles des modules commerciaux peu connus.

2.1.2 Fonctionnement

OCropus, comme n'importe quel logiciel d'OCR, fonctionne selon 5 étapes principales [?] : Le pré-traitement, l'analyse de page, la reconnaissance des caractères, le post-traitement puis la mise en forme du résultat

Pré-traitement

Cette étape a pour but de traiter l'image d'entrée afin de faciliter les étapes suivantes. Cela implique par exemple de redresser une image inclinée, de travailler sur le contraste... OCropus fourni une toolbox pour les images en noir et blanc ou niveau de gris.

Analyse de page

Une fois l'image optimisée pour le traitement, elle est découpée en lignes de texte. Pour cela, OCropus recherche d'abord la présence de colonnes, puis découpe chacune de ces colonnes en lignes. Ces lignes pourront ensuite être traitées séparément

Reconnaissance de caractère

Les lignes en sortie de l'étape précédente sont analysées afin d'en reconnaître les différents caractères. Pour se faire, les images de caractères sont analysées une à une et comparées par différentes méthodes a des images de caractères connus dans une

base de donné. Le ou les caractères les plus proches sont choisis. C'est dans cette partie que Tesseract est utilisé.

Post-traitement

Une analyse statistique est ensuite utilisée pour améliorer le résultat et lever les éventuelles ambiguïtés. Des dictionnaires sont utilisés pour calculer la probabilité de séquences de lettres, ce qui permet de poser un modèle linguistique utilisé dans cette étape.

Mise en forme

Les lignes ayant été analysées une par une, une dernière étape est nécessaire afin de mettre en forme les différents résultats obtenus afin d'obtenir une résultat final unique : une page html contenant le texte de sortie. Un script bash nous permet de convertir cet page html en un fichier txt.

2.1.3 Résultats

Nous avons tenté d'utiliser OCRopus pour obtenir le texte présent dans différentes images.

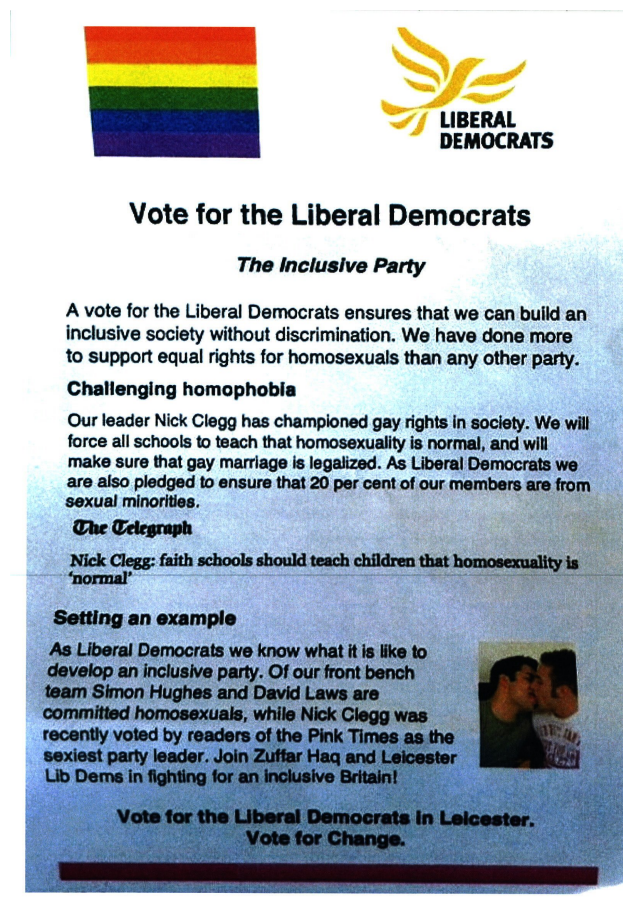


FIGURE 2.1 – Image à traiter par OCRopus

```

--
-.
t,iaeau
DEMOCRAS
Vote for the Llberal Democrats
T7e icluslve Party
Avote for the Liberal Democrats ensures that we can bulld an
inclusive society without discrimination. We have done more
to support equal rights for homosexuals than any other party.
Challenglng homophobla
Our leader Nick Clegg has champloned gay rights in soclety. We wili
force all schools to teach that homosexuality is normal, and wl
make sure that gay marlage ls legslizad. As Lberal Democrats we
are also pledged to ensure that 20 per cent of our members are from
sexulal minoritles.
tae tCetesraph
Nick Clegg: faith schools should teach children that homosexuality is
Tomal'
Setting an example
As Llberal Democrats we know what tis ske to
zpy,
team Sion Hughes and Davld Laws are
recently voted by readers of the Pink Tlmes as the
sexiest party leader. Join Zuffar Haq and Leicester
Lib Dems in fighting tor an inclusive Britainl
-r'7
L- -
Vote for thr Lib=P=] f)wmrncrst In Lelrester.
-
Vc;' y ir ('k -eyrxs,
1
H-|

```

FIGURE 2.2 – Texte trouvé par OCRopus

Avec cette image, on remarque que les lettres se ressemblant (i et l) sont souvent confondues, et que la qualité de l'image est importante : le bas de la page, moins bien éclairé, est mal reconnu et le mot "normal", situé sur la pliure, est interprété comme Tomal'. On remarque également que la police est importante : The Telegraph est interprété comme tae tCetesraph.

Des essais sur d'autres images nous ont montré que la rotation est importante (une image tournée à 90 deg n'est pas du tout reconnue par OCRopus) ainsi que l'espacement entre les caractères, qui peut rendre la police plus ou moins "lisible". Enfin, les écritures manuscrites ne sont pas du tout reconnues par OCRopus.

2.2 Cld2

2.2.1 Généralités

CLD2, ou Compact Language Detection 2, est un package proposé par google [?] pour détecter la langue d'un texte, fourni sous la forme d'une chaîne ou d'une page HTML/XML. Comme son nom l'indique, il fait suite à CLD en y ajoutant plusieurs améliorations et des nouveautés, comme des langues supplémentaires (CLD2 peut en détecter 83 différentes, nous ne les listerons pas toutes ici) et la possibilité d'obtenir en sortie le texte correspondant aux 3 langues principales quand plusieurs langues sont détectées, pour pouvoir par exemple tout traduire dans une même langue.

CLD2 a été créé pour travailler sur des pages web comprenant plus de 200 caractères. De ce fait, des "indices", tels que des balises html concernant la langue ou le Top Level Domain de l'URL (.fr, .de ...) sont utilisés pour influencer les résultats du package. Étant donné que nous n'utiliserons pas CLD2 sur des pages web mais sur des chaînes de textes extraites par Ocropus, ces indices ne seront pas utilisés dans notre cas.

2.2.2 Nettoyage des données

La première étape effectuée par CLD2 afin de travailler sur le texte et détecter la langue est de nettoyer le texte fourni pour pouvoir le traiter.

Tout d'abord, tout le texte est passé en minuscule. Ensuite, tous les nombres, la ponctuation et les tags html, non utiles pour la détections de la langue, sont supprimés. Les mots d'une seule lettre sont également ignorés.

Les mots sont séparés en séquences de 4 lettres, appelés quadrigramme. Le signe `_` est ajouté au début ou à la fin du quadrigramme pour indiquer que ce quadrigramme se trouve au début ou à la fin du mot, ce qui peut être caractéristique d'une langue.

Les mots qui ne sont pas caractéristiques d'une langue et pouvant biaiser la detection, tels que les extensions (.jpg, .gif, ...) sont supprimés.

2.2.3 Analyse statistique

Une fois les données nettoyées, une analyse statistique peut être effectuée sur le texte afin d'en détecter la langue. On travaillera sur les quadrigramme créé précédemment.

La méthode utilisée est probabiliste. Une table contenant des entrées de 4 bytes est utilisée afin d'associer à chaque quadrigramme entre 3 et 6 langues les plus probables. Un "score" est alors calculé à partir de la probabilité logarithmique du quadrigramme pour ces langues.

La table contenant les langues et les probabilités a été créée à partir d'un corpus de texte. Dans un premier temps, des pages web en chacune des langues ont été choisies humainement et traitées. 100 millions de pages web choisies automatiquement ont ensuite été ajoutées.

2.2.4 Résultats

- "Bonjour, vive notre PAO" : FRENCH(95% 1291p)
- "Nous sommes ravis de vous rencontrer." : FRENCH(97% 885p)
- "Hello, nice to mee you too" : ENGLISH(96% 1175p)
- "Nous sommes ravis de vous rencontrer. Hello, nice to meet you too. " : Unknown(un-reliable). On constate que 2 phrases dans deux langues différentes qui sont bien détectées séparément ne sont pas bien détectées lorsqu'elles sont ensemble.
- "Nous sommes ravis de vous rencontrer. Visiblement il faut beaucoup plus de texte pour pouvoir detecter la langue. Hello, nice to meet you too." : FRENCH(99% 633p)
- "Nous sommes ravis de vous rencontrer. Visiblement il faut beaucoup plus de texte pour pouvoir detecter la langue. Hello, nice to meet you too. We need to

put a little more text here too. Let's hope it works." : FRENCH(56% 772p), ENGLISH(43% 1314p). Il nous fout plus de texte dans chaque langue quand 2 langues sont mélangées.

- "Il se rague roupète drôle emparouille endosque pratèle libucque barouffle ouillais tocarde marmines manage" : Unknown(un-reliable). En mettant des mots n'existant pas mais à sonorité française (extraits du poème Le Grand Combat d'Henri Michaux), on s'attendrait à ce que CLD2 trouve la langue française mais ce n'est pas le cas. CLD2 n'est pas si facile à berner !
- "Essayons de trouver" : FRENCH(95% 870p). 3 mots français peuvent suffire pour détecter la langue, ce qui est mieux que les 200 mots minimum prévus.
- "esrdtu Essayons eghedu de eedef trouver kjhgvb jjkl zzzz" : FRENCH(98% 300p). L'ajout de séquences de lettres aléatoires ne perturbent pas beaucoup CLD2
- "Essayons de troujver" : Unknown(un-reliable). Le fait de modifier un mot perturbe en revanche beaucoup le résultat.

Chapitre 3

Conception de l'application

3.1 L'application Android

3.1.1 Conception

Nous avons développé cette application pour les systèmes d'exploitation Android. La programmation pour Android est basée sur le langage java que nous avons vu l'année dernière en cours de programmation avancée. Néanmoins, même si nous possédions de bonnes bases java, il nous a fallu plusieurs semaines afin de savoir développer sur Android car celui-ci fonctionne différemment du langage java que nous avons pu voir.

Une fois le langage assimilé, nous nous sommes attaqués à la conception de l'application. Nous avons tout d'abord élaboré un dessin papier des différentes vues de notre application. Au fur et à mesure que nous avançons dans le développement nous nous sommes rendu compte qu'il serait plus compliqué que prévu de réaliser toutes ces vues. Nous avons donc opté pour une application plus simple mais qui répond à nos attentes.

La figure ci-dessous présente le menu de démarrage de l'application :

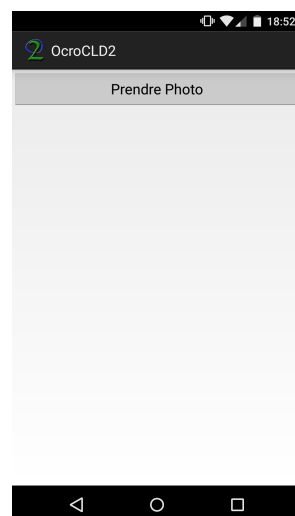


FIGURE 3.1 – Menu principal

3.1.2 Installation

Pour installer l'application sur mobile Android, il suffit juste de copier ocro-CLD2.apk sur le smartphone. Ensuite, activer l'installation d'applications non certifiées et installer l'application.

3.1.3 Fonctionnement

L'intégralité du code de l'application est disponible dans l'archive fournie.

Lorsque nous sommes dans le menu principal, il nous suffit d'appuyer sur le bouton "prendre une photo" afin que l'application fasse appel à l'application appareil photo de notre mobile. Pour faire ceci, nous utilisons un listener qui lorsque l'on clique sur le bouton, l'application crée un intent afin d'utiliser l'application en charge des photos. Ceci nous a permis de ne pas redévelopper l'appareil photo déjà présent.

Une fois que nous avons pris la photo, il nous suffit de valider et nous retournons à l'accueil. L'application envoie alors grâce à notre classe DataSender l'image par protocole Http sur l'adresse du script php de notre serveur. Une fois cela fait, l'application reste à l'écoute du serveur afin de récupérer la chaîne de caractère renvoyée. Cette chaîne contient le résultat du traitement par Ocropus puis CLD2.

Nous pouvons alors reprendre une photo ou bien quitter l'application.

3.1.4 Tests

Nous avons effectué quelques tests sur notre application dont voici l'un des résultats :

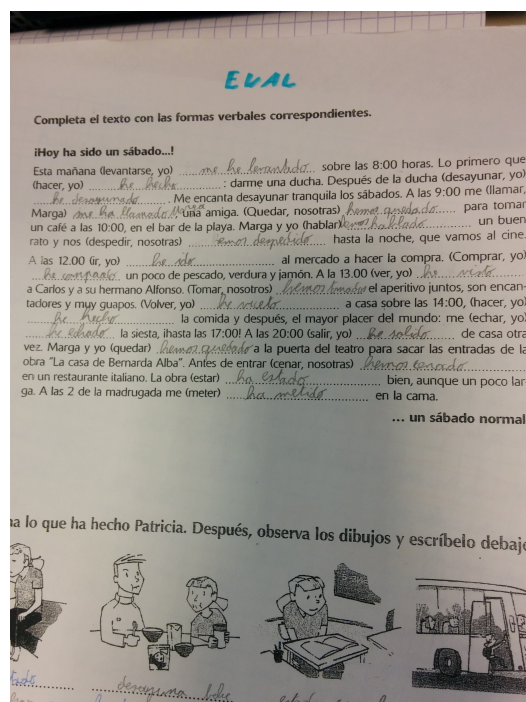


FIGURE 3.2 – Document en espagnol pris en photo

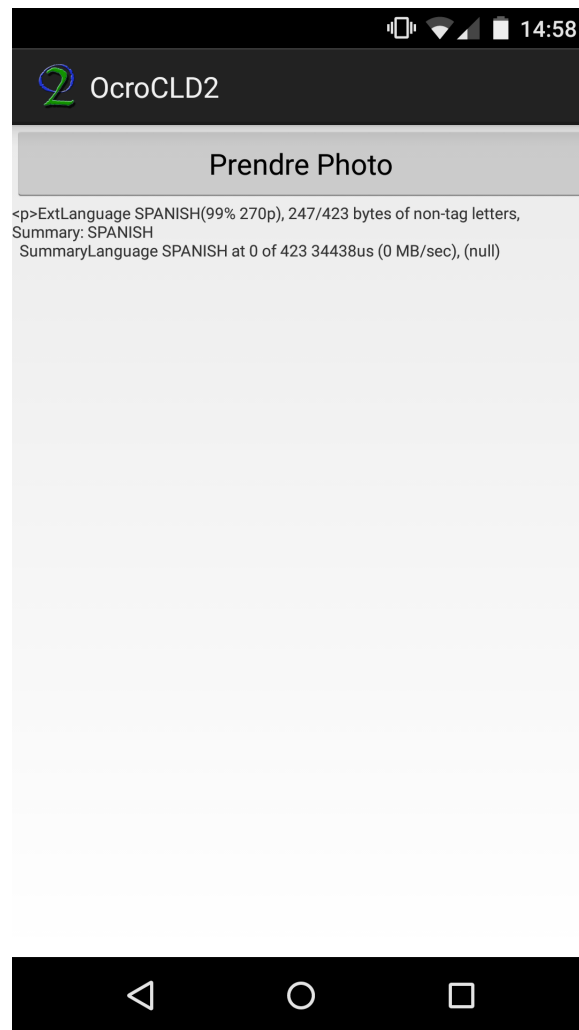


FIGURE 3.3 – Menu principal

Nous pouvons voir que l'image a bien été traitée par le serveur et que la langue renvoyée par CLD2 est la bonne.

3.1.5 Problèmes rencontrés

Tout d'abord, l'apprentissage du java pour Android est apparu plus difficile que prévu car nous ne pensions pas qu'il y avait autant de différences. De plus, nous avons eu quelques soucis pour installer le SDK android sur Eclipse. C'est pourquoi nous avons pris un peu de retard au début de ce projet.

Ensuite, il a été assez compliqué de faire communiquer l'application avec le serveur. Plus précisément, nous ne savions pas quel type de protocole était le plus adapté afin d'envoyer une photo sur un serveur. Après avoir consulté Mr Nicolas Malandin, nous avons donc opté pour le protocole Http.

Finalement, nous avons toujours des problèmes de réponse du serveur : par moment nous ne sommes pas certain de recevoir une réponse. En effet, le traitement de l'image par Ocropus prend plusieurs minutes suivant la taille du texte à traiter.

3.2 La communication avec le serveur

3.2.1 Connection au serveur

Afin d'installer et de faire nos tests nous nous connectons en ssh sur un serveur que Mr Sebastien Bonnegent nous a mis à disposition : `ssh -X paoandroid@asi-android.insa-rouen.fr`. De plus, il faut se trouver sur le réseau WEP-26 de l'INSA de Rouen pour pouvoir se connecter. Lorsque le PAO sera terminé, le serveur sera fermé, il faudra donc redemander un serveur pour pouvoir reprendre le projet.

3.2.2 Installation des outils

Ocropus

- Dans un premier temps, il faut cloner le répertoire d'Ocropus à l'aide de la commande : `hg clone -r ocropus-0.7 https://code.google.com/p/ocropus` ;
- Si cela ne fonctionne pas, il faut installer le package que linux vous suggère ;
- Ensuite, déplacez vous dans le dossier ocropy : `cd ocropus/ocropy` ;
- Installez les packages nécessaires au fonctionnement : `sudo apt-get install $(cat PACKAGES)` ;
- Téléchargez les modèles python : `python setup.py download_models` ;
- Lancez l'installation : `sudo python setup.py install` ;
- Lancez les tests : `./run-test` ;
- Si tout est correctement installé, vous devriez avoir un message d'Ocropus disant que vous pouvez visualiser les résultats en tapant une certaine ligne de commande.

CLD2

- Allez dans votre dossier tmp : `cd /tmp/` ;
- Téléchargez le répertoire de CLD2 : `svn checkout http://cld2.googlecode.com/svn/trunk/cld2` ;
- Déplacez vous dans le dossier internal : `cd cld2/internal` ;
- Copiez et collez le fichier `compile_libs_32bit.sh` dans le répertoire internal (ce fichier est joint dans l'archive que nous avons fourni). Celui-ci sert à pouvoir exécuter CLD2 dans un terminal ;
- Autorisez l'exécution de ce fichier : `chmod u+x compile_libs_32bit.sh` ;
- Exécutez le fichier : `./compile_libs_32bit.sh` ;
- Effectuez le test suivant dans votre terminal : `echo "Hello World je suis gentil trop bien my name is i don't know" | cld2` ;
- Si tout est correctement installé votre terminal affichera un message vous disant que le texte est en anglais et français.

Lynx

Il faut installer lynx qui permet de convertir une page html en fichier txt : `sudo apt-get install lynx`.

3.2.3 PHP

Comme dit précédemment, l'application envoie l'image au serveur par protocole Http directement sur un script php qui, lorsqu'il reçoit l'image, la copie dans le bon dossier et lance les différentes commandes nécessaires au traitement de l'image. Ce script est disponible dans l'archive fournie. Une fois que Ocropus a fini CLD2 est exécuté et renvoie une chaîne de caractère transmise à l'application par protocole Http.

3.2.4 Problèmes rencontrés

Le premier problème que nous avons rencontré avec le serveur vient d'Ocropus. En effet, lorsque celui-ci est exécuté, il ouvre une fenêtre graphique afin de traiter les images. Celle-ci n'est pas visible à l'écran mais empêche le bon fonctionnement d'Ocropus, d'où le -X dans la commande permettant de se connecter au serveur qui permet de se servir de notre écran comme écran miroir.

Afin de résoudre ce problème nous avons modifié quelques lignes de code des bibliothèques d'Ocropus afin que celui-ci n'ouvre plus de fenêtre graphique pour exécuter le script bash directement grâce à une requête HTML et ne plus avoir de message d'erreur. Afin de savoir quel fichier modifier, il faut se connecter au serveur sans le -X et exécuter Ocropus.

Voici les trois lignes de code à rajouter au tout début du fichier :

```
— import matplotlib
— matplotlib.use('Agg')
— import matplotlib.pyplot
```

Si le problème persiste après c'est que vous n'avez pas modifié le bon fichier python.

Le deuxième problème fut les droits d'accès aux fichiers. En effet, lorsque l'application android envoie la requête sur le script php du serveur, ce script est exécuté par Apache. Il faut donc que Apache possède les droits d'accès, de modification et d'exécution sur les dossiers d'Ocropus et tous ceux nécessaires au bon fonctionnement du script.

Pour résoudre ce problème nous avons donnés les droits root à Apache ce qui ne constitue pas forcément la meilleure des solutions car très risquée. Néanmoins, vu que cette application est destinée uniquement à être utilisée dans le cadre du PAO, cela nous a semblé être la solution la plus simple et rapide.

Chapitre 4

Conclusion

Ce PAO nous a permis d'approfondir nos connaissances dans le domaine du traitement d'image notamment avec l'outil d'OCR qu'est Ocropus et de voir les principes théoriques appliqués lors du traitement de l'image mais aussi comment CLD2 reconnaît une langue à partir d'un texte.

De plus, ce projet fut un bon moyen de mélanger diverses technologies telles que le php ainsi que le java orienté Android. Nous avons pu voir que faire communiquer des systèmes avec des technologies de développement différentes n'est pas forcément quelque chose de facile. Malgré cela, nous sommes parvenus à un résultat très proche de ce que l'on souhaitait faire au départ.

Pour ce qui est des améliorations à apporter au projet par la suite s'il est repris, nous avons pensé à ceci :

- Tout d'abord, il faudrait améliorer l'interface de l'application avec l'ajout d'un message d'avancement du traitement ainsi qu'une modification de la chaîne de caractère du résultat de CLD2 afin de n'avoir que la langue et le pourcentage.
- Ensuite, il serait peut être possible avec plus de connaissances en python d'essayer de diminuer le temps de traitement par Ocropus en modifiant le code. Ou bien en utilisant d'autres bibliothèques d'OCR en remplacement d'Ocropus.
- Le développement ou l'utilisation d'un explorateur de fichier sur android afin de transférer au serveur de traitement une photo de document directement présente sur le mobile.
- Enfin, il sera utile de rendre l'application disponible en dehors du réseau de l'INSA afin de pouvoir traiter une image de n'importe où.

Pour finir, nous avons beaucoup appris de ce PAO qui a été très instructif pour notre formation d'ingénieur. En effet, nous avons découvert de nouveaux outils non étudiés dans le cadre de notre scolarité tels que la programmation Android et vu des applications de nos cours théorique comme le traitement d'image par Ocropus.