OCR PYTHON

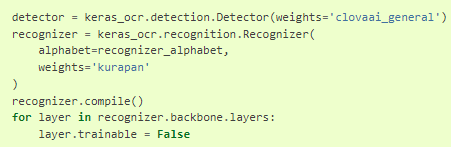
Keras-OCR :

Installer en python : pip install keras-ocr

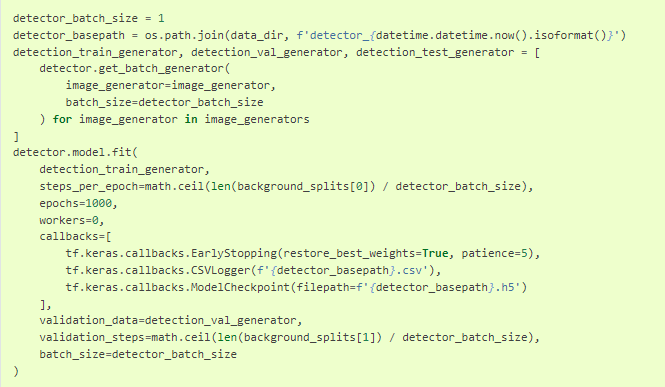
Construire modèle de détecteurs et de reconnaissance de base :

Les deux modèles (détecteur et reconnaissance de base) seront pré-entrainé, on fige donc les poids dans le backbone (toutes les couches sauf la couche de classification finale).

Création du détecteur et reconnaissance :



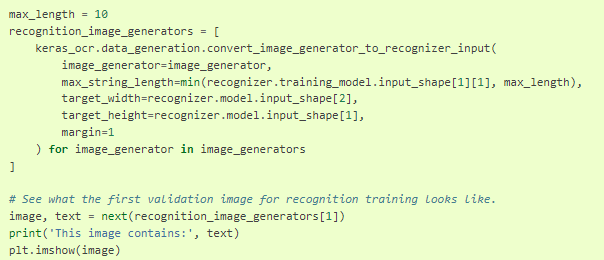
Entrainer le détecteur :



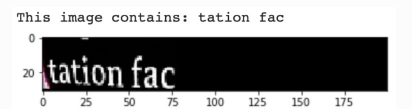
L’detector objet a une get\_batch\_generator méthode qui convertit le image\_generator (qui renvoie les images et les annotations associées) en un batch\_generator qui renvoie des paires pour l’entrainement avec .X, y fit\_generator.

Former le reconnaisseur :

Maintenant nous entrainons le module de reconnaissance ce dernier s’attend à ce que les images soient déjà recadrées en une seule ligne de texte. Keras-ocr fournit une méthode pratique pour convertir notre générateur existant en générateur a ligne unique.

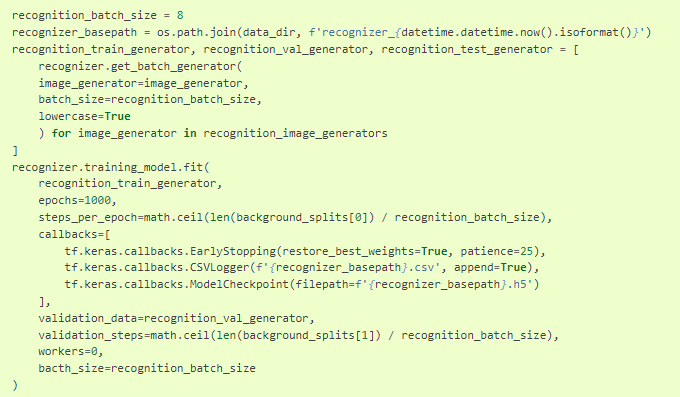


Le résultat ressemble à ceci :



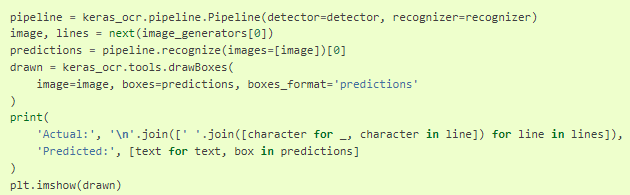
Tout comme le detector, le recognizer dispose d’une méthode pour convertir le générateur d’images en un batch\_generator que Keras fit\_generator peut utiliser.

Les mêmes rappels sont utilisés pour l’arrêt anticipé et la journalisation qu’auparavant.



Utiliser les modèles pour l’inférence :

Une fois que la formation est finie, on peut utiliser la pipeline recognize pour extraire du texte.

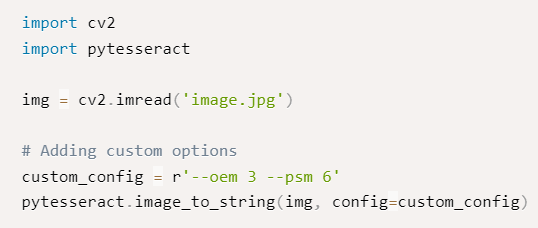


Résultats :



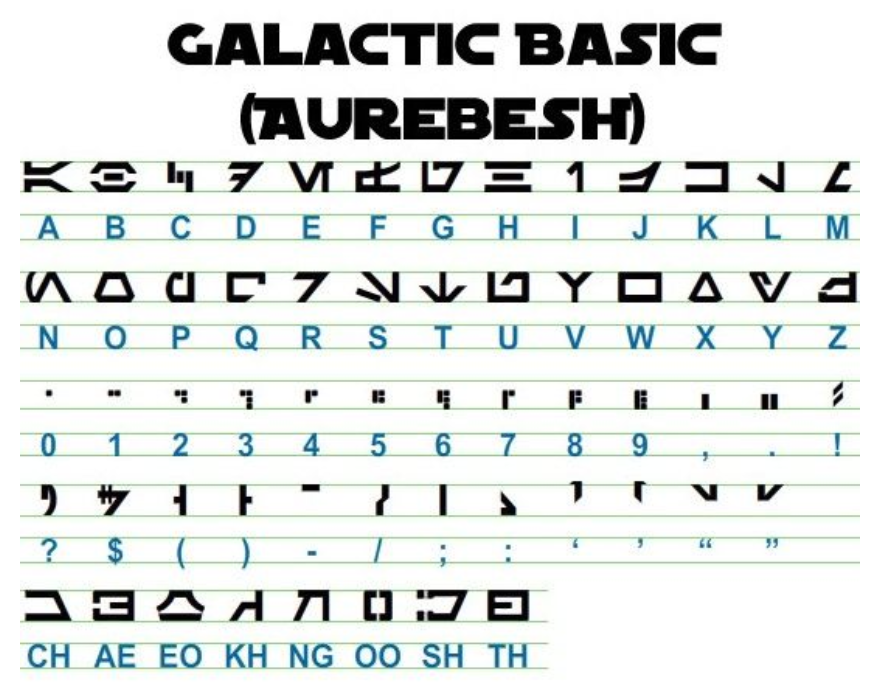
**PyTesseract**

Pytesseract ou Python-tesseract est un outil OCR pour Python qui sert également de wrapper pour le moteur Tesseract-OCR. Il peut lire et reconnaître le texte dans les images et est couramment utilisé dans les cas d'utilisation d'images ocr python en texte.

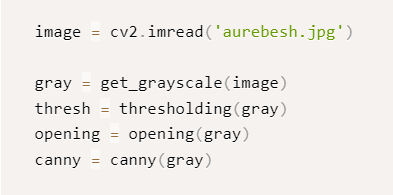


Pré-traitement sur l’image pour Tesseract :

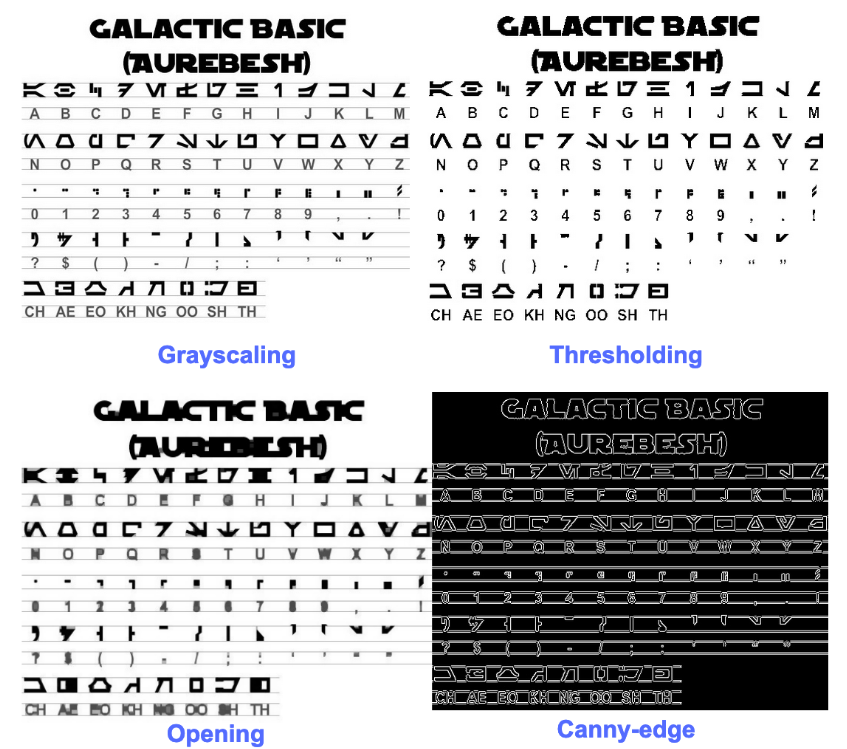
Image originale :



Application du pré-traitement :

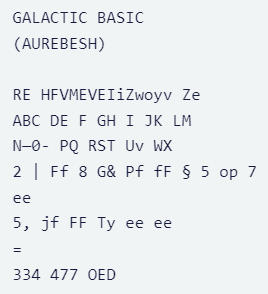


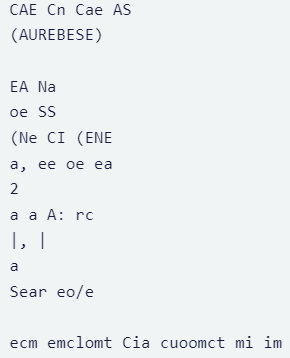
Résultat :

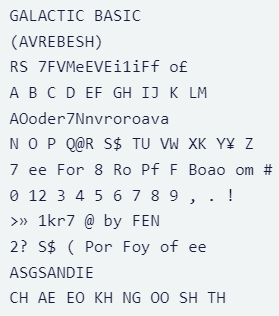


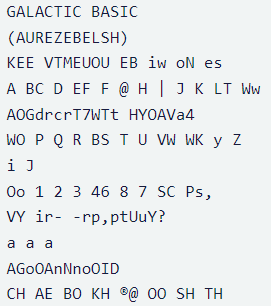
On peut remarquer que certaine images comme sur le Canny-edge c’est très peut lisible il est difficile de distinguer certaine image comparer au Grayscaling ou encore Thresholding.

Les résultats print pour chaque image sont les suivant :



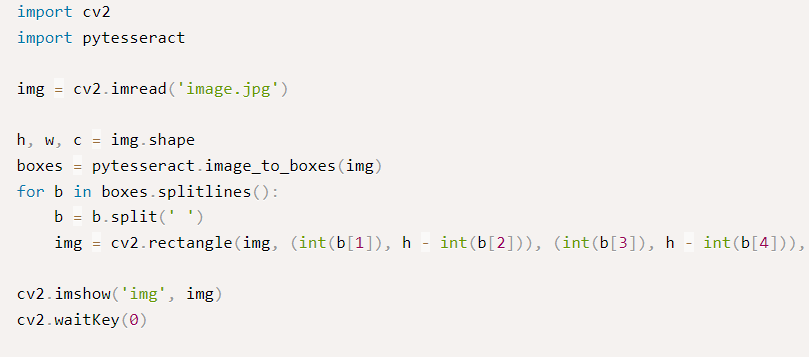






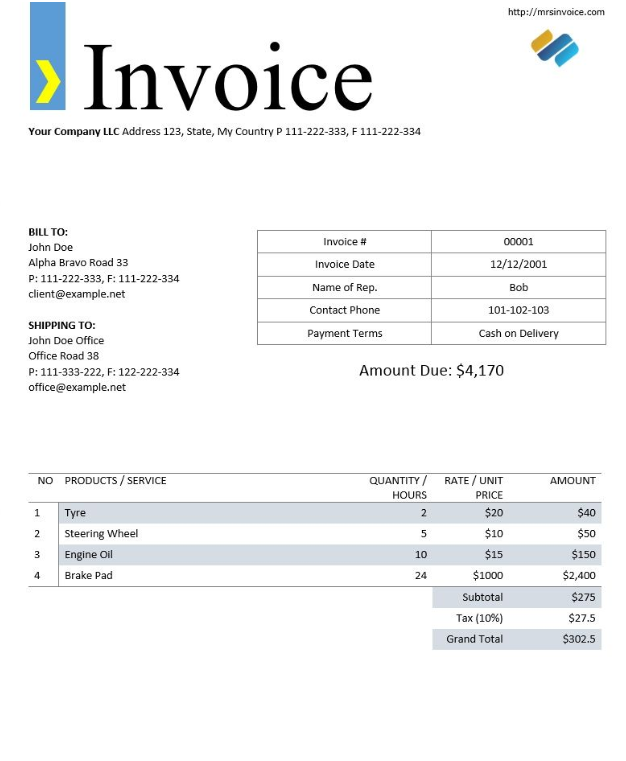
Obtenir des cadres autour du texte :

Il est possible d’obtenir des boites englobantes pour les résultats, grâce au code ci-dessous.

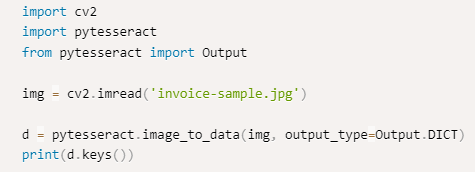


Il est possible d’entourer les mots plutôt que chaque caractère pour cela utiliser la fonction image\_to\_data.

Exemple d’une facture :



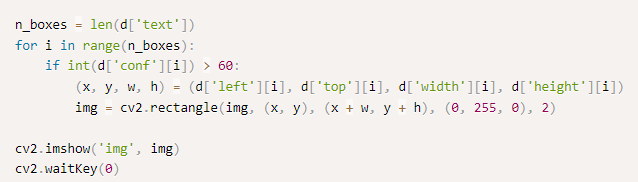
Pour avoir une sortie des mots détecter il suffit de faire cela :



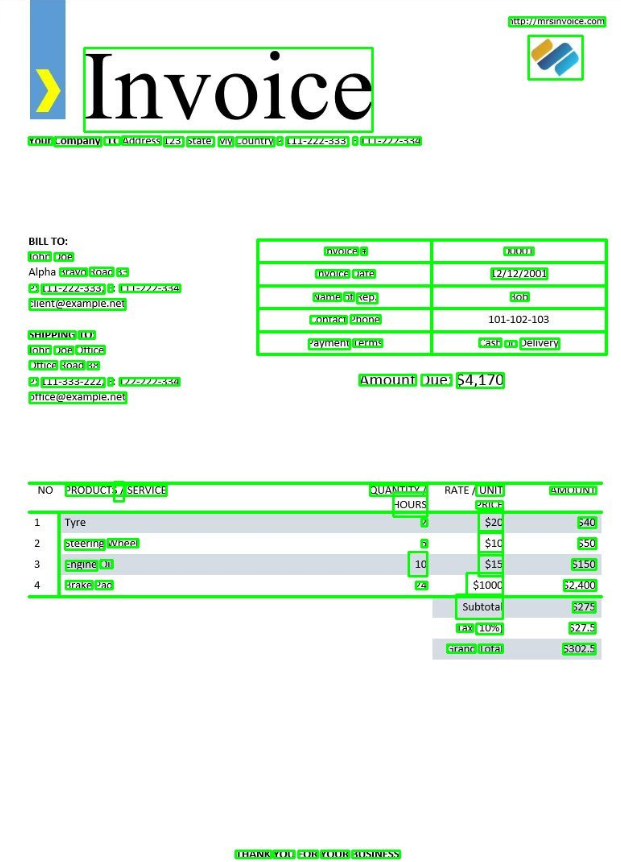
Ce qui retourne un dictionnaire de chaque mot détecté, les informations de son cadre de délimitation, le texte qu’il contient et les scores de confiance pour chacun.



Il est possible de tracer des cases autour de chaque mot sur l’image avec ce code :



Ce qui nous donne :



Pour trouver une date sur une facture il est possible en utilisant un regex :



Et le résultat est correct comme présenter ci-dessous :

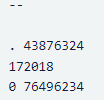


Détecter uniquement des numéros sur une image :





La sortie ressemblerait à sa :



Il est aussi possible de traiter que les lettres de a à z grâce à une white\_list.

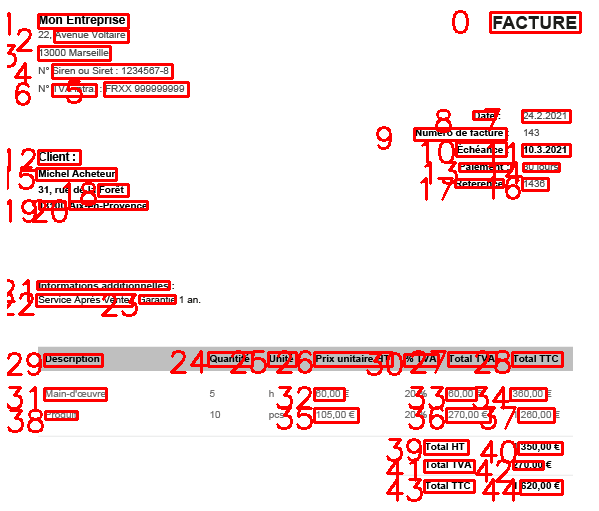
**EasyOCR**

J’ai fait un test sur une facture que j’ai déjà tester grâce a la démo proposer sur le web.

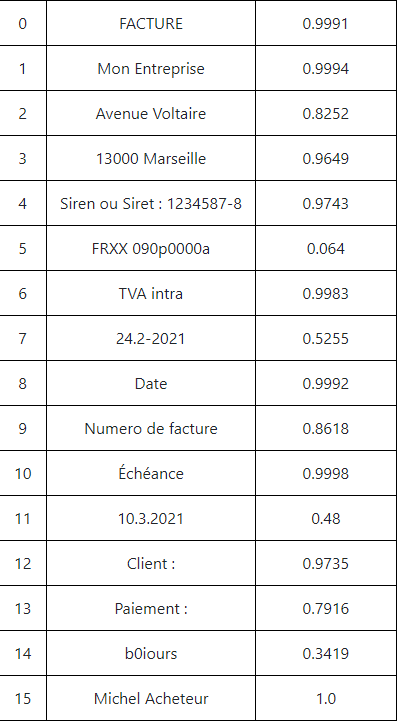
Lien : <https://www.jaided.ai/easyocr/>

Il nous demande de choisir un fichier puis de choisir la langue et de faire un process.

Une fois le process terminer on peut voir sur le résultat ci-dessous que ce dernier a crée différentes box autour de chaque mot ou chiffres trouvé :



Et nous ressort en sortie un tableau avec le No. Le texte qui a été détecté et le score de confiance.



On peut remarquer que sur les éléments détectés certain n’ont pas été détecter comme les différentes quantité (5 et 10).

Les différentes classes de ce modèle sont disponible à ce lien : <https://www.jaided.ai/easyocr/documentation/>