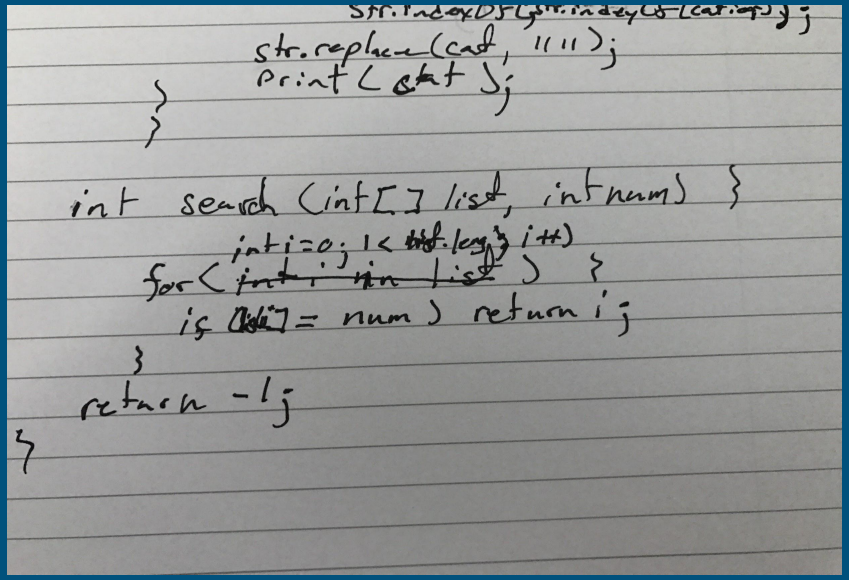
Reconnaissance de code manuscrit

L’intérêt du projet ?

En TD, on écrit des algorithmes sur feuilles sans pouvoir les tester...



L’intérêt du projet ?

- Pas besoin de recopier son code de tester
- Compilation et exécution directement et facilement
- Logiciel évolutif !

Gain de temps !

Fonctionnalités

- Lire une image d’un code manuscrit
- Possibilité de modifier le résultat
- Ajouter le résultat à la base de données pour mieux comprendre l’utilisateur
- Compilation et exécution du code lu

Architecture du projet

Pattern Modèle-vue-contrôleur(MVC):

- Interface: Javafx
- Controleur: Java
- Model: Python

Repartition des modules

5 Modules indépendants:

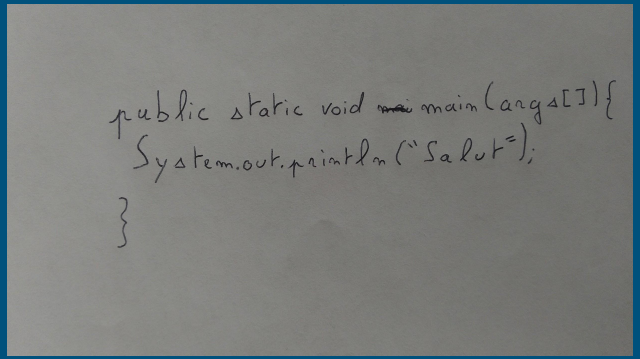
- Segmentation et unification des lettres manuscrites
- Base de données (Machine learning)
- Comparaison
- Interface
- Compilation et exécution

Difficultés rencontrées

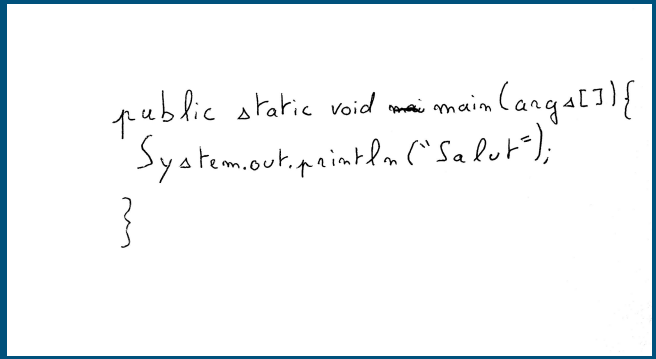
- Élaboration des algorithmes
- Documentation sur le machine learning
- Utilisation de certaines API

Segmentation de lignes et de lettres

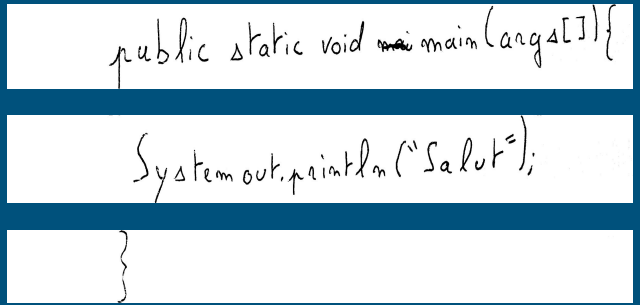
1)



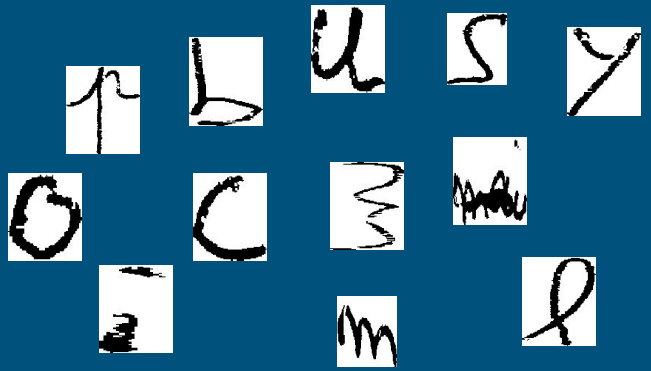
2)



3)



4)



Conclusion

- Projet qui a demandé beaucoup de recherche de par sa complexité
- Des améliorations sont possibles