Rapport de Projet : Classification et Extraction de Texte

Mohamed Amine Mrabet

8 septembre 2025

1 Justification de l'approche choisie

Nous avons adopté deux approches complémentaires de classification afin d'améliorer la robustesse du système :

- **Classification basée sur la couleur** : utile pour détecter rapidement et efficacement des classes distinctes en fonction de la distribution des couleurs. Cette approche est simple, interprétable et peu coûteuse en calcul.
- **Classification basée sur l'apprentissage automatique** : permet de capturer des patterns plus complexes et de généraliser au-delà des simples différences de couleurs. Elle offre plus de flexibilité et de précision sur des cas réels variés.

L'utilisation conjointe de ces deux approches permet d'obtenir un système hybride capable de traiter à la fois des cas simples et des scénarios plus complexes.

2 Structure et organisation du code

Le projet est organisé en deux notebooks principaux :

- `Color-Based-classification.ipynb` : contient la partie classification basée sur l'analyse des couleurs. Ce notebook comprend les étapes de prétraitement des images, l'extraction des caractéristiques colorimétriques et la mise en place de règles de décision.
- `problem_statement.ipynb` : intègre la partie classification par apprentissage automatique ainsi que l'extraction de texte via Pytesseract. Ce notebook traite le pipeline complet depuis l'importation des données jusqu'aux résultats finaux.

3 Gestion des cas d'erreur et edge cases

Plusieurs cas particuliers ont été traités :

- Problèmes liés aux dépendances et compatibilités de bibliothèques (notamment EasyOCR).
- Détection de texte dans des zones non pertinentes.

4 Explication des choix techniques

- Pytesseract a été retenu pour sa facilité d'utilisation et sa compatibilité avec de nombreuses langues.
- L'approche par couleur est rapide et interprétable, idéale pour des tests préliminaires.
- L'approche par apprentissage automatique apporte une meilleure précision et adaptabilité.
- Le choix d'adapter les versions des librairies a permis de surmonter les problèmes de compatibilité rencontrés.

5 Analyse des résultats obtenus

— La classification basée sur la couleur fonctionne correctement dans des cas simples et bien délimités.

— La classification par apprentissage automatique s'est révélée plus performante pour des cas variés.

— L'extraction de texte via EasyOCR a permis de récupérer l'intégralité du texte présent dans les documents, bien que le filtrage reste perfectible.

6 Explication de l'approche

6.1 Notebook 1 : Classification basée sur la couleur

1. Chargement et affichage des images.
2. Conversion en espace colorimétrique adapté (HSV/RGB).
3. Segmentation et classification selon des seuils définis.
4. Visualisation des résultats obtenus.

6.2 Notebook 2 : Classification par apprentissage automatique et OCR

1. Prétraitement des données (redimensionnement, normalisation).
2. Extraction des caractéristiques pour l'entraînement.
3. Entraînement d'un modèle de classification supervisée.
4. Application d'EasyOCR pour l'extraction de texte.
5. Définition de la ROI pour isoler les zones pertinentes.
6. Analyse et visualisation des résultats.

7 Instructions pour reproduire les résultats

1. Cloner le projet et installer les dépendances.
2. Ouvrir et exécuter le notebook `Color-Based-classification.ipynb`.
3. Ouvrir et exécuter le notebook `problem_statement.ipynb`.
4. Vérifier que les données d'entrée (images) sont bien présentes dans les dossiers attendus.

8 Analyse des performances et limitations

— L'approche basée sur la couleur est limitée aux cas où les classes sont visuellement distinctes.

— Pytesseract, bien qu'efficace, génère parfois du bruit et nécessite un post-traitement.

— Le manque de temps a limité le raffinement du filtrage du texte extrait.

9 Défis rencontrés

— Problèmes de compatibilité avec Pytesseract, résolus par l'adaptation des versions de bibliothèques.

— Détection de texte hors contexte, corrigée par une définition précise de la ROI.

— Temps d'implémentation conséquent pour atteindre une extraction de texte fiable.

10 Améliorations possibles

— Développer un post-traitement plus robuste pour filtrer et structurer correctement le texte extrait.

— Optimiser la gestion des ROI pour gagner en temps et en précision.

— Avec un jour supplémentaire, il aurait été possible de nettoyer et structurer correctement le texte et les champs extraits.