Assistant au Scrabble

Matthieu Annequin - Arthur Guedon - Valentin Odde

Abstract

Ce projet de Vision par Ordinateur consiste en la mise en place d'un assistant au Scrabble. En donnant en entrée, une grille de Scrabble et le tirage de lettre effectué, on renvoie le mot qui donne le score le plus important.

Keywords: ComputerVision, Scrabble, IA for fun

Contents

1. Introduction

Notre objectif lors de ce projet est de construire un outil prenant en entrée une photographie d'un plateau de Scrabble et une série de lettre et de trouver le meilleur mot à poser sur le plateau afin de maximiser son nombre de point.

Ce projet comprend ainsi une grande partie de computer vision afin d'analyser le plateau à partir d'une photo et d'en extraire les données sous-jacentes.

On peut dès maintenant esquisser les différentes sous parties du projet au sein de la partie vision.

Il faut tout d'abord traiter l'image reçue afin de la recadrer pour qu'elle ne contienne que la partie du plateau qui nous intéresse, c'est à dire celle où se situe le carré de 15 cases sur 15 cases où peuvent être placées les lettres. Nous reconstruirons la position des cases afin d'isoler une image pour chacune des 15*15 cases. Nous classifierons ensuite les cases entres celles qui possèdent une lettre et celles qui sont vides. Enfin pour les cases non vides, qui contiennent donc une lettre, nous devrons identifier la lettre dont il s'agit.

Ainsi en suivant ce processus, notre objectif est de reconstruire la grille de Scrabble de manière algorithmique afin d'en obtenir un objet python sur lequel nous pourrons analyser les mots possibles à placer en fonction des lettres dont nous disposons.

L'image utilisée pour notre projet provient d'un projet similaire (3). Nous nous sommes inspirés de la pipeline générale de son projet.

2. Fonctionnement du github

2.1. Organisation du github

Le github est composé d'un dossier data qui contient quelques images que nous avons utilisées pour faire notre projet. Notamment notre image de référence 'board1.jpeg', ainsi que les images 'I.jpg' et 'I2.jpg' qui sont nécessaire pour fine tuner le modèle. Les deux autres fichiers servent à la réalisation du Solver.

Le github est également composé du dossier Vision dans lequel on trouve les fichiers relatifs à la vision par ordinateur de la grille :

- *rescale.py* : ce fichier contient toutes les fonctions qui permettent de recréer une image centrée sur la grille
- *dataset_creation.py* : ce fichier permet de créer un dataset qui servira à fine tuner le modèle
- *ViT_finetuning.py* : ce fichier contient le code pour fine tuner le modèle
- *letter_detection.py* : ce fichier contient le code permettant de prédire les lettres présentes sur la grille
- *get_grid.py* : ce fichier permet de réaliser le préprocessing nécessaire au traitement de la grille

Le dossier Solver contient les fichiers permettant de gérer la suggestion de mots sur la grille:

- *dawg.py* : ce fichier contient toutes les fonctions permettant de créer la structure de graphe orienté acyclique de mots (DAWG) à partir du dictionnaire du Scrabble
- *board.py* : ce fichier permet de gérer le fonctionnement de la grille de jeu et la recherche de mots sur celle-ci
- *solver.py* : ce fichier contient les fonctions permettant de suggérer des mots à partir d'une grille

Enfin, le fichier *main.py* sert à faire tourner l'ensemble de notre programme.

2.2. Fonctionnement du programme

2.2.1. Fine-tuning du modèle

Pour détecter correctement les lettres sur le Scrabble nous avons besoin de fine-tuner le modèle. En effet, celui-ci a du mal à distinguer les "I".

On lui donne ainsi en entrée les images I et I2.

Puis on fine-tune le modèle avec la commande suivante : *python − mVision.ViT_finetuning*

Une fois que l'entraînement a fini de tourner votre modèle s'enregistre dans le dossier modèle.

2.2.2. Obtenir le meilleur mot

Pour être assisté au Scrabble, il vous faut ensuite exécuter la commande python *main.py*. On vous demande alors votre image. Indiquez par exemple "board1.jpeg".

L'algorithme détecte alors la grille et vous en donne un aperçu.

On vous demande ensuite les lettres que vous avez piochées. Entrez vos 7 lettres (par exemple : AEBNERT).

Le programme vous renvoie alors le mot avec sa position ainsi que la nouvelle grille générée en jouant ce mot.

3. Vision par ordinateur du plateau

3.1. Identification des contours de la grille

Pour isoler la partie de l'image qui nous intéresse, nous allons tout d'abord identifier les contours de la grille.

Nous suivons donc les étapes suivantes à l'aide des fonctionnalités offertes par OpenCV :

- On convertit l'image en noir et blanc
- On applique un filtre gaussien sur notre image
- On applique ensuite un filtre de Canny afin d'extraire les "arêtes" de notre image
- On peut ensuite extraire les contours grâce à la fonction getContours de la bibliothèque OpenCV
- On garde enfin le contour le plus long et on l'identifie au contour de notre grille.

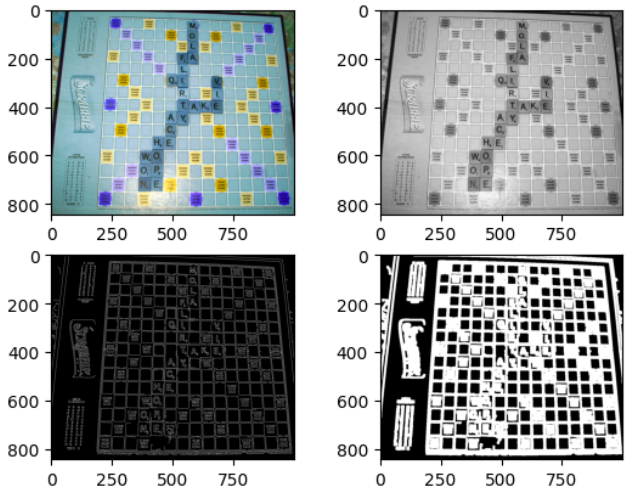

Figure 1: Étapes de détection des contours

Figure 2: Détection des contours de la grille

3.2. Transformation géométrique

Dans un second temps, lorsque nous avons identifié les contours de la grille nous allons créer une nouvelle image carrée à partir de notre grille en la recadrant. Pour cela, nous utilisons un outil de transformation géométrique, présent dans la bibliothèque opencv, celle-ci est notamment utile pour aplatir des objets qui sont déformés par la perspective. De manière plus concrète, nous identifions les angles de notre contour qui est plus ou moins carré selon la position de l'appareil lors de la prise de vue. Ensuite, une fois les quatre angles identifiés, nous les projetons sur les 4 angles de notre image carrée. Ainsi, nous obtenons une image carrée et recadrée de la grille, comme si l'image avait été prise parfaitement au dessus du plateau. Cette image va ensuite nous servir pour identifier les cases.

Figure 3: Grille recadrée par transformation géométrique

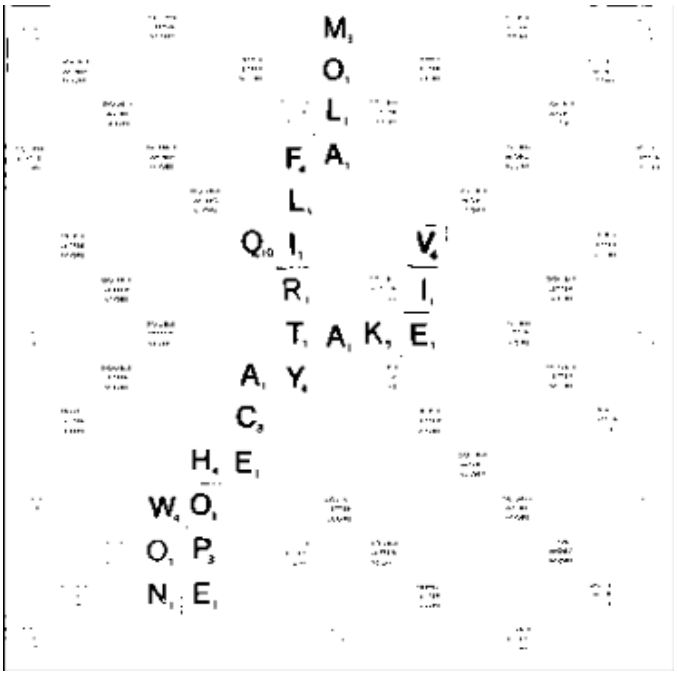

Figure 4: Image de la grille après traitement et seuillage HSV

3.3. Identification des cases

Notre objectif dans cette partie est de découper notre image carrée du plateau afin d'en extraire les différentes cases. Nous savons que les plateaux de Scrabble sont carrés et composés de 15 sur 15 cases. Ces cases sont toutes identiques et elles mêmes carrées. Ainsi le tout forme un quadrillage régulier. A partir de notre image carrée nous la découpons donc selon ce quadrillage afin d'en extraire nos différentes cases.

Ce mécanisme fonctionne très bien et a le mérite d'être très simple à mettre en place. Cela dit, il est très dépendant de la qualité et de la précision de notre transformation de l'image initiale. En effet, si la grille est décalée dans l'image ou bien présente une marge trop importante sur un côté, alors cela va entièrement décaler nos cases et ainsi tout ce qui s'ensuit.

3.4. Identification des cases vides

Cette partie est sans doute la plus technique. Nous devons identifier les cases ne présentant pas de lettres. Cela n'est pas si évident car certaines cases vides contiennent des inscriptions (mots compte double...) qui peuvent facilement s'apparenter à une lettre dans notre programme.

Ainsi voici comment nous procédons.

Tout d'abord nous transformons l'image RGB de notre grille en un encodage HSV.

Nous appliquons ensuite une opération de seuillage gaussien sur la composante V de notre image afin d'identifier les lettres.

Nous obtenons donc ensuite une image binaire. A cette étape, les lettres sont bien visibles mais nous détectons toujours les cases vides contenant du texte.

A ce moment nous découpons donc notre grille en cases comme nous l'avons expliqué précédemment.

Ensuite nous avons une image binaire pour chacune des cases. Nous procédons ensuite à un clustering sur ces images afin d'identifier les différents amas de pixels. Nous gardons ensuite seulement le plus gros des clusters et nous éliminons les autres.

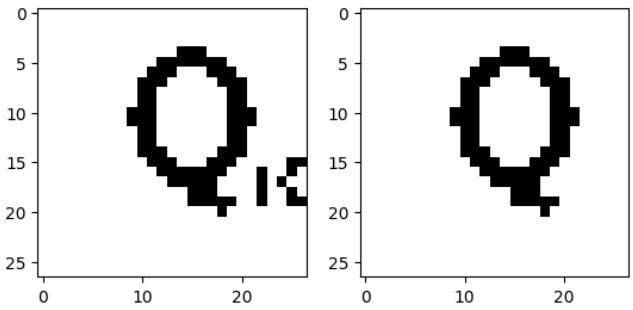

Figure 5: Traitement d'une image de lettre par sélection du plus gros cluster de pixel

Ensuite, si la taille du cluster restant dépasse un certain seuil, nous considérons qu'il s'agit d'une lettre et dans le cas contraire, nous estimons que la case est vide.

4. Classification des lettres

4.1. Modèle de classification VIT

Pour identifier les lettres à partir des images dont nous disposons, nous avons utilisé un modèle VIT préentrainé sur un jeu de données de lettres manuscrites (2)

Pour rappel, le modèle VIT (Vision Transformer) est un réseau de neurones convolutif basé sur la transformer architecture, qui est traditionnellement utilisé pour le traitement

de langage naturel. Contrairement aux CNNs traditionnels, qui utilisent des opérations de convolution pour extraire des caractéristiques visuelles, le modèle VIT utilise une séquence de blocs transformers pour capturer les motifs spatiaux dans une image. Il est capable d'atteindre des performances comparables, voire supérieures, aux meilleurs modèles CNNs sur plusieurs jeux de données de référence pour la vision par ordinateur.

Ce modèle fonctionne très efficacement sur notre jeu de données. Il a cela dit du mal à identifier les "I", et les interprète comme des "L". Cela vient du jeu de données d'entraînement qui regroupait des lettre manuscrite, à l'inverse de nos données qui sont plutôt formalisé comme des lettres d'ordinateur.

4.2. Fine tuning pour différencier les I

Afin de régler ce problème, nous avons légèrement fine tuné le modèle en lui donnant en entrée des images labelisées de "I" provenant de nos images de Scrabble. En utilisant un faible learning rate et un jeu de données de 60 images de "I", nous obtenons de bien meilleurs résultats et nous arrivons à détecter les I bien plus efficacement.

5. Outil de suggestion de coups

L'objectif de cette partie est de créer un outil qui permette, à partir de la grille de Scrabble et des lettres dont dispose le joueur, de suggérer les lettres à placer de façons optimales pour maximiser le nombre de points à chaque tour.

5.1. Répresentation du dictionnaire du Scrabble

Pour développer notre outil d'assistance de jeu, nous nous sommes appuyés sur le dictionnaire officiel du Scrabble qui était en 2012 composé de 386 264 mots français. Afin d'effectuer des requêtes efficacement dans cette base de données, nous avons représenté ce dictionnaire sous forme de graphe orienté acyclique de mots (DAWG). (1)
Cette structure a l'avantage de permettre de représenter le dictionnaire de manière réduite et de faciliter la recherche dans une suite de lettres. En effet, le DAWG considère que si deux mots ont des préfixes différents mais un suffixe commun alors ils peuvent partager un noeud en commun pour le suffixe. Cela permet donc de réduire considérablement la représentation du dictionnaire. Pour construire cette structure nous représentons d'abord le lexique sous forme d'arbre avec chaque noeud représentant une lettre et les feuilles formant des mots. Nous transformons ensuite cet arbre en DAWG que nous enregistrons sous format pickle pour pouvoir le réutiliser facilement, à noter que cette structure est 3 fois moins lourde que le dictionnaire original.

5.2. Recherche de coup optimal

Une fois que nous avons réussi à détecter les lettres présentes sur le plateau de jeu et que nous avons représenté de manière efficace le dictionnaire du Scrabble, nous avons développé l'outil permettant de suggérer un coup optimal. Notre outil prend alors en entrée l'état actuel de la grille de jeu, les lettres dont dispose

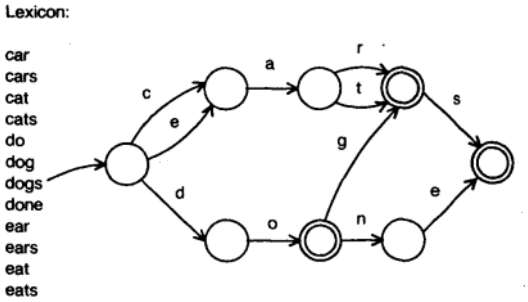

Figure 6: Graphe orienté acyclique de mots (Appel & Jacobson 1988 (1))

le joueur et indique au joueur les lettres à poser rapportant le plus de points ainsi que leur emplacement.

De manière plus détaillée, voici comment fonctionne cet algorithme de recherche de coup optimal:

- Nous parcourons la grille horizontalement et verticalement (pour cela nous transposons la grille et donc on ne raisonne qu'en ligne, avec une lecture de gauche à droite)
- Lorsqu'une lettre est rencontrée, nous recherchons dans le lexique (représenté sous forme de DAWG) tous les préfixes possibles à partir des lettres dont dispose le joueur.
- A partir de ce préfixe (ou simplement de la lettre rencontrée), nous recherchons tous les suffixes possibles. L'exploration se fait case par case et lorsqu'un noeud terminal du DAWG est rencontré, le mot formé est ajouté à la liste des coups possibles.
- Nous calculons le score de chacun des coups possibles, et nous renvoyons celui donnant le plus de points.

Par ailleurs, afin de simplifier la recherche de préfixes et suffixes, chaque case du plateau adjacente à une lettre dispose d'une liste de "cross checks" qui correspond aux lettres jouables sur ces cases. Cette liste est mise à jour à partir du dictionnaire à chaque modification de la grille.

6. Limites et conclusion

L'outil que nous avons développé permet, à partir d'une photo d'une grille de Scrabble, ainsi que des lettres dont dispose un joueur de suggérer les lettres à poser qui lui rapportent le plus de points.

Cependant, notre modèle présente certaines limites. Tout d'abord il est nécessaire d'adapter le traitement de l'image initiale en fonction du type de plateau de Scrabble puisque les seuils que nous utilisons peuvent varier d'une version à une autre du jeu. De plus, notre outil ne permet pas actuellement de traiter les "joker", ces pièces sont confondues avec des cases vides du plateau et ne sont pas gérées par notre algorithme de suggestion de coups. Par ailleurs, pour l'instant lorsque notre modèle de détection de lettre n'est pas assez confiant sur un résultat nous la remplaçons par une case vide. Nous n'avons pas

pu la développer par manque de temps mais il serait possible de proposer une interface à l'utilisateur afin de vérifier le résultat et lui permettre de rentrer manuellement les lettres dans les cases incertaines identifiées. Cela pourrait permettre également de gérer les pièces "joker" en permettant à l'utilisateur de les signaler.

References

[1] A. W. APPEL and G. J. JACOBSON. The world's fastest scrabble program. *Communications of the ACM*, 31(5):578–585, 1988.

[2] H. Face. Vision transformers.

[3] SRDewan. Scrabble-assistant, 2021.