Compte Rendu - OCR Project

Afflelou*

30 octobre 2024

*Louis C. - Puis A. - Malgouyres A.

Table des matières

1 Introduction

1.1 Résumé

Ce rapport vise à présenter l'état d'avancement de notre projet de recherche de mots cachés. Nous débuterons par la description des différentes fonctionnalités développées jusqu'à présent, telles que le chargement d'images, la détection de la grille et la résolution de mots cachés. Nous expliquerons également le fonctionnement de notre réseau de neurones, illustrant son apprentissage avec une fonction logique simple. Enfin, nous conclurons par une démonstration pratique des éléments réalisés, en mettant en avant l'interaction entre les différentes parties du programme et les résultats obtenus.

1.2 Présentation du groupe

1.2.1 Corentin

Je m'appelle Corentin, et l'informatique a toujours été pour moi une passion motivée par la curiosité de comprendre comment fonctionnent les systèmes complexes. J'ai une affection particulière pour le domaine de la cybersécurité, un secteur qui me passionne pour ses défis constants et son importance dans notre société actuelle.

1.2.2 Arnaud

Je m'appelle Arnaud, j'ai 18 ans et l'informatique est un domaine qui me passionne depuis mes années de lycée. J'éprouve un réel plaisir à résoudre des problèmes algorithmiques et j'ai un intérêt marqué pour le développement de jeux vidéo. L'analyse d'image, avec ses multiples défis techniques, exige la mise en œuvre de solutions créatives et complexes, c'est pourquoi je me suis spécialisé dans les aspects liés à la détection sous toutes ses formes.

1.2.3 Adrien

Je m'appelle Adrien, j'ai 19 ans. Mes motivations pour l'informatique sont d'apprendre de nouvelles choses et d'améliorer mes compétences dans le plus de domaines

liés à l'informatique possible. Je suis très intéressé par l'intelligence artificielle sous toutes ses formes, notamment la reconnaissance optique. C'est pourquoi je m'occuperai principalement du réseau de neurones.

1.3 Aspects humains

Nous avons décidé de travailler ensemble parce que nous nous connaissions avant le projet ; nous sommes tous les trois amis. Cette amitié est un réel atout dans la réalisation d'un projet de groupe, car il n'y a aucun risque de tension. Nous avons également choisi de ne travailler qu'à trois pour des raisons de facilitation de la répartition des tâches principalement.

2 Répartition des charges

	Attribution	Aide complémentaire
Prétraitement	Corentin	
Rotation manuelle	Corentin	
Détection grille/mots	Arnaud	
Détection et découpage des lettres	Arnaud	
Word Solver	Adrien	Corentin
Réseau de neuronnes	Adrien	
Interface GUI	Corentin	

2.1 Corentin

Au sein de ce projet, je me suis chargé du prétraitement, de la rotation manuelle, et de la création de l'interface GUI. Bien que ces tâches ne soient pas directement liées à mes motivations dans l'informatique, elles m'ont permis de perfectionner des compétences, comme la gestion précise des données et la création d'interfaces.

2.2 Arnaud

Pour ma part, j'ai pris en charge la détection de la grille et du quadrillage afin de pouvoir, par la suite, segmenter celle-ci en différentes lettres. Ce domaine, qui était totalement nouveau pour moi, m'a offert l'opportunité d'apprendre énormément sur l'analyse d'images en explorant divers algorithmes. Je m'occupe donc de l'implentation de nombreux algorithmes afin de pouvoir detécter tout ce qui est nécessaire à la résolution de la grille sur l'image.

2.3 Adrien

Je m'occupe du réseau de neuronne. L'IA m'intéresse tout particulièrement et j'étais très heureux à l'idée de pouvoir créer mon réseau de neuronne. L'équipe et moi avons donc décider de m'attribuer la charge du réseau de neuronne. J'ai également participé à la création du solver, fini très rapidement. Je suis en charge des rapports.

3 État d'avancement

L'avancement actuel du projet est satisfaisant. Nous avons réussi à achever la plupart des deadlines imposées et la plupart des deadlines que nous nous étions imposées. Nous avons pris du retard sur l'interface graphique (Interface GUI). Nous avons voulu prioriser le backend avant le frontend et avons donc consacré plus de temps aux réseaux de neurones et au traitement d'image.
Pour le solver, c'est la première étape que nous avons terminée. Il est fonctionnel. Cependant, nous aimerions l'améliorer si nous avons plus de temps. En effet, nous voulions rajouter un système avec des graphes pour reconnaître les mots commençant de la même manière et donc gagner en efficacité.

	Avancement actuel	Avancement requis
Prétraitement	100%	100%
Rotation manuelle	100%	100%
Détection grille/mots	60%	100%
Détection et découpage des lettres	80%	100%
Word Solver	90%	100%
Réseau de neuronnes	33%	33%
Interface GUI	50%	75%

Le traitement d'image a été une étape prenant plus de temps que prévu. En effet, nous avons particulièrement eu du mal sur le découpage, la détection pour récupérer chaque lettre individuellement. Nous n'avons donc pas encore de fonction permettant de récupérer la liste de mots. Pour rattraper ce retard, nous allons continuer à prioriser le reste du backend (réseaux de neurones et traitement d'image) et ralentir le frontend à savoir l'interface graphique.

4 Aspects techniques

4.1 Prétraitement

Technologies et bibliothèques utilisées Plusieurs bibliothèques externes ont été utilisées :

- **SDL :** SDL.h et SDL_image.h
- **GTK :** gtk.h

Ainsi que des bibliothèques plus courantes telles que : string.h, math.h, stdlib.h et stdio.h.

Aspects techniques : Fonctionnement du prétraitement Le prétraitement est une étape essentielle, garantissant que l'image est dans un format optimal avant de procéder aux étapes de détection de la grille et de la reconnaissance de lettres. Voici les techniques de prétraitement que nous avons utilisées :

Conversion en niveaux de gris Le processus commence par la conversion de l'image en niveaux de gris. Cela simplifie le traitement en réduisant la complexité des informations de couleur, tout en conservant l'essentiel des contours et des formes. Chaque pixel est transformé en une valeur unique de luminosité, calculée à partir de ses composantes rouges, vertes et bleues.

Formule de conversion : gris = 0.299 * rouge + 0.587 * vert + 0.114 * bleu

Amélioration du contraste

- **Contraste :** L'amélioration du contraste est essentielle pour distinguer les détails significatifs dans une image. En utilisant un facteur de contraste calculé, nous ajustons la luminosité des pixels pour accentuer les différences entre les zones claires et sombres. Cela permet de rendre les contours et les formes plus nets et plus perceptibles, surtout lorsque l'image d'origine a des variations de luminosité faibles.
- **Binarisation :** La binarisation est un processus de traitement d'image qui convertit une image en niveaux de gris en une image composée uniquement de deux couleurs : le noir et le blanc. Cette technique est particulièrement utile dans le cadre de l'analyse d'images, car elle permet de simplifier les données tout en conservant les éléments d'intérêt.

Application d'un filtre gaussien L'application d'un filtre gaussien est une étape cruciale dans le processus de prétraitement des images. Ce filtre est utilisé pour réduire le bruit et les variations abruptes de luminosité, ce qui améliore la qualité globale de l'image avant d'appliquer d'autres techniques d'analyse.

— **Principe du filtre gaussien :** Le filtre gaussien repose sur une fonction gaussienne qui attribue des poids aux pixels voisins lors du calcul de la valeur d'un pixel central. Les pixels les plus proches du pixel central ont un poids plus élevé que ceux qui sont plus éloignés. Cela permet d'obtenir un effet de flou qui adoucit les transitions de couleur tout en préservant les contours des objets.

4.2 Rotation manuelle de l'image

Technologies et bibliothèques utilisées Plusieurs bibliothèques externes ont été utilisées :

— **SDL :** SDL.h et SDL_image.h

— **GTK :** gtk.h

Ainsi que des bibliothèques plus courantes telles que : string.h, math.h, stdlib.h et stdio.h.

Aspects techniques : Fonctionnement de la rotation La rotation manuelle d'une image est une fonctionnalité qui permet aux utilisateurs de faire pivoter une image à un angle donné, facilitant ainsi l'ajustement et la correction de l'orientation de l'image. Cette fonctionnalité est essentielle dans les applications de traitement d'image, car elle aide à améliorer la visualisation et la lisibilité des contenus.

Processus de rotation Le processus de rotation de l'image s'articule autour des étapes suivantes :

— **Chargement de l'image :** L'image est chargée à l'aide de SDL_image. Une fois chargée, elle est convertie en texture pour permettre son rendu dans une fenêtre SDL.

— **Résultats et sauvegarde :** Une fois la rotation effectuée, l'image peut être sauvegardée en cliquant sur le bouton de sauvegarde. Cette fonctionnalité permet de conserver les modifications et d'utiliser l'image tournée pour des traitements ultérieurs.

4.3 Détection de la grille et de la liste de mots

Detection de lignes Bien que la détection complète de la liste de mots ne soit pas encore implémentée, notre projet intègre déjà un algorithme performant permettant de localiser la position de la grille. Cette détection repose sur deux procédés fondamentaux : d'abord, l'application du filtre de Canny pour identifier les contours dans une image, et ensuite, l'utilisation de la transformée de Hough pour détecter les lignes. L'implentation de ces derniers se fait à l'aide de la bibliothèque SDL.

Explication Le filtre de Canny est un algorithme de détection de contours qui met en évidence les variations brusques d'intensité dans une image, ce qui permet de repérer les bords des objets. Il faut d'abord récuperer les valeures de luminosité de chaque pixel pour ensuite appliquer un kernel / matrice de convolution sur ces valeures, dans ce cas le filtre de canny, afin de récuperer les positions dans l'image ou l'intensité varie fortement. Cette étape préliminaire est essentielle pour simplifier l'image et isoler les éléments importants.

Une fois les contours identifiés, la transformée de Hough est utilisée pour détecter les lignes présentes dans l'image. Cet algorithme permet de transformer l'espace des pixels en un espace de paramètres où chaque ligne potentielle est représentée par un ensemble de points. C'est-à-dire que chaque point dans l'espace cartésien représente une ligne dans l'esapce de Hough. Ainsi, lorsqu'un grand nombre de lignes convergent dans un point de cet espace, cela indique la présence d'une ligne dans l'image d'origine. Ce point d'intersection est aux coordonnées (theta, rho) dans l'esapce de Hough ce qui représente les coordonnées polaire d'une ligne dans l'espace cartésien.

Ainsi grace à la transformée de Hough, nous pouvons repérer la grille en prenant les quatres lignes plus exterieurs qui forment un rectangle.

4.4 Détection et découpage des lettres

Procédé En s'appuyant sur les mêmes processus évoqués précédemment, nous sommes en mesure de découper les grilles qui possèdent déjà un quadrillage afin de déterminer la position précise de chaque lettre. Cette étape repose sur l'analyse des intersections des lignes détectées grâce à la transformée de Hough. En calculant ces points d'intersection, il devient possible de diviser la grille en sections correspondant à chaque case contenant une lettre.

Une fois que la position des lettres est identifiée, ces segments de la grille peuvent être extraits individuellement. Ces parties découpées sont ensuite fournies à un réseau de neurones pour identifier les caractères.

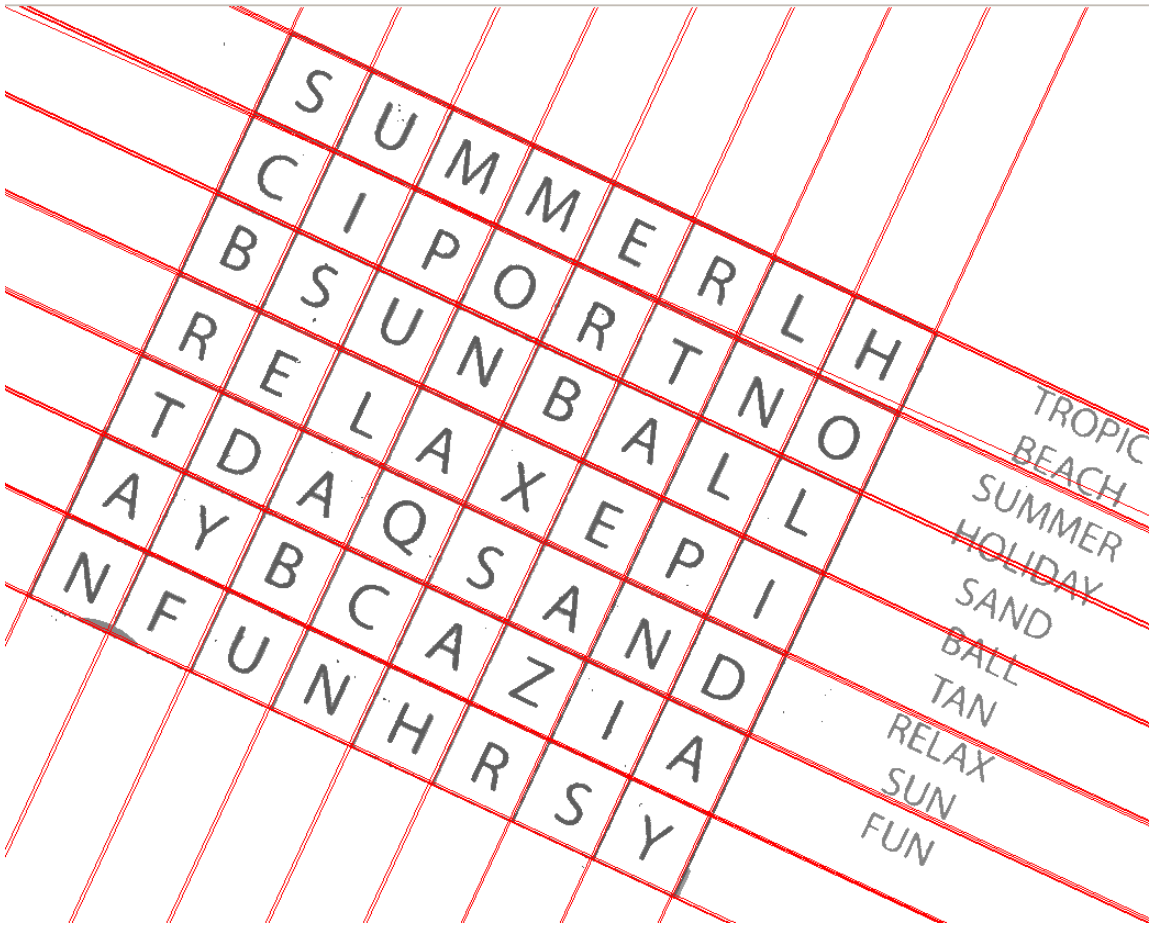

FIGURE 1 – Line detection

Contraintes Cependant, une contrainte majeure subsiste : la détection des lettres lorsque le quadrillage n'est pas déjà tracé sur la grille. Dans ce cas, il est nécessaire de recourir à une autre approche pour repérer la position des lettres. Cela impliquerait de localiser les espaces blancs entre les lettres et de déduire les lignes séparatrices qui segmentent ces dernières.

Cette tâche est plus complexe, car elle exige un traitement supplémentaire pour identifier les zones homogènes sans texte, qui constituent les séparations naturelles entre les lettres. Une fois ces espaces identifiés, des lignes virtuelles peuvent être tracées pour créer un quadrillage approximatif permettant de segmenter la grille. Ces lignes permettront ensuite de découper la grille en zones correspondant à chaque lettre.

4.5 Solver

Technologies et bibliothèques utilisées Aucune bibliothèque externe n'a été employée (hormis string.h, maths.h etc.).

Aspects techniques : Fonctionnement du Solver de Mots Mêlés Pour construire un programme capable de résoudre un mot mêlé, nous avons utilisé une approche qui parcourt la grille dans toutes les directions pour identifier des séquences correspondant à un mot cible. Voici les principaux aspects techniques et choix logiques de cette implémentation.

Gestion des Données d'Entrée et Préparation de la Grille

- **Chargement de la Grille :** Le programme commence par ouvrir le fichier texte spécifié par l'utilisateur, qui contient la grille de mots mêlés. Chaque ligne du fichier est lue et stockée dans une matrice `grid[100][100]`, créant une représentation 2D de la grille. En définissant le nombre de lignes et de colonnes en fonction de la taille du fichier, on garantit une gestion dynamique des grilles de différentes tailles, avec un maximum de 100x100 pour éviter la surcharge mémoire.
- **Pré-traitement du Mot Cible :** Pour permettre une recherche insensible à la casse, la fonction `remove_lowercase` transforme toutes les lettres du mot cible en majuscules. Ce pré-traitement simplifie la comparaison, car la grille est traitée sans tenir compte de la casse des lettres, et permet de rechercher de manière uniforme dans la grille, quelle que soit l'écriture du mot initialement.

Algorithme de Recherche et de Comparaison de Mot Le cœur de l'algorithme de recherche réside dans le parcours de chaque case de la grille en vérifiant toutes les directions possibles (horizontale, verticale, et diagonales). Cette méthode repose sur deux concepts :

- **Parcours Multi-Directionnel :** Huit directions sont définies pour explorer chaque point de départ possible dans la grille. Pour chaque case initiale, un ensemble de valeurs de `dir_x` et `dir_y` sont associées à une direction :
  - `dir_x` et `dir_y` permettent de définir les déplacements sur la grille (par exemple, 1,1 pour se déplacer en diagonale vers le bas à droite).
  - La boucle `for` itère de 0 à 7, où chaque entier représente l'une des directions.
- **Recherche Incrémentale de Séquences :** La fonction `starts_with` permet de vérifier si le début d'une chaîne donnée correspond au mot cible. Cela est

effectué en parcourant la grille à partir de chaque case de départ possible, dans la direction actuelle. La comparaison commence avec le premier caractère et se poursuit en ajoutant un caractère supplémentaire à chaque itération de la boucle.
  - Si `starts_with` renvoie 2, cela signifie que la séquence en cours correspond exactement au mot cible : le programme affiche alors les coordonnées de départ et d'arrivée.
  - En revanche, si `starts_with` retourne 0, l'algorithme passe à une autre direction ou à un autre point de départ.
- **Gestion des Limites de la Grille :** Lors de l'extension de la séquence dans une direction spécifique, des vérifications s'assurent que la recherche reste dans les limites de la grille. Cela empêche les débordements en dehors de la matrice (i.e., éviter de lire des données invalides).

4.6 Réseau de neurones

Technologies et bibliothèques utilisées Aucune bibliothèque externe n'a été employée (hormis string.h, maths.h etc.)afin de mieux comprendre et contrôler chaque étape du réseau de neurones, de la construction des couches à l'entraînement.

Structure du réseau de neurones Le réseau de neurones comporte trois couches :
- **Couche d'entrée :** 2 neurones représentant les deux entrées binaires (ex. [0, 1]).
- **Couche cachée :** 2 neurones, permettant de capter des non-linéarités.
- **Couche de sortie :** 1 neurone, générant la sortie binaire (1 ou 0)

Les neurones sont dotés d'un biais (poids ajustable), d'une fonction d'activation (ReLU pour la couche cachée et Sigmoïde pour la sortie) et de poids vers la couche suivante.

Chaque neurone est défini dans une structure `neuron_t` contenant ses attributs principaux :
- **actv :** activation de sortie du neurone.
- **out_weights :** liste des poids vers les neurones de la couche suivante.
- **bias :** biais du neurone, initialisé aléatoirement.

Chaque couche (`layer_t`) stocke un tableau de `neuron_t`, correspondant au nombre de neurones spécifié par `NEURONS_NUMBER`.

Initialisation et création du réseau La fonction `create_network` initialise chaque couche en respectant l'architecture prédéfinie (`NEURONS_NUMBER`), et elle attribue les

poids et biais aléatoirement pour briser la symétrie et favoriser l'apprentissage.

```
layer create_layer(int neuron_number) {
    neuron *neurons = calloc(neuron_number, sizeof(neuron));
    layer lay;
    lay.neuron_number = neuron_number;
    lay.neurons = neurons;
    return lay;
}
```

Propagation avant (Forward Propagation) La propagation avant (fonction `forward_prop`) passe les valeurs d'activation d'une couche à l'autre :

- **Calcul de la somme pondérée :** Pour chaque neurone de la couche courante, une valeur z est calculée en fonction des activations de la couche précédente.
- **Application de la fonction d'activation :**
  - **ReLU pour les neurones cachés :** activations z négatives sont fixées à zéro.
  - **Sigmoïde pour le neurone de sortie :** normalise la sortie entre 0 et 1, utile pour un problème binaire.

Rétropropagation (Backpropagation) La rétropropagation (fonction `back_prop`) calcule l'erreur et ajuste les poids en fonction de la dérivée de l'erreur (descente de gradient) :

- **Erreur en sortie :** calculée par rapport à la sortie souhaitée (`DESIRED_OUTPUTS`).
- **Propagation de l'erreur dans chaque couche,** ajustant les poids et biais pour minimiser l'erreur globale.
- **Mise à jour des poids et biais en fonction de `LEARNING_RATE`.**

Entraînement et sauvegarde des poids Pour entraîner le réseau, une boucle d'entraînement exécute `EPOCHS` itérations de propagation avant et arrière sur chaque entrée de l'opération XOR. L'erreur quadratique moyenne est affichée périodiquement, et le réseau est réinitialisé si cette erreur reste élevée après de nombreuses itérations.

Les poids et biais finaux sont sauvegardés dans un fichier texte avec la fonction `save_weights` pour éviter de réentraîner le réseau à chaque exécution.

Cette structure permet une reconstruction rapide du réseau et une amélioration de la compréhension des concepts clés comme l'initialisation des poids, la propagation avant/arrière et l'optimisation des paramètres du modèle.

On peut voir sur ci-dessous que le script charge le fichier txt weights avec les poids sauvegardés pour exécuter la "forward propagation" sans entrainner de nouveau le réseau :

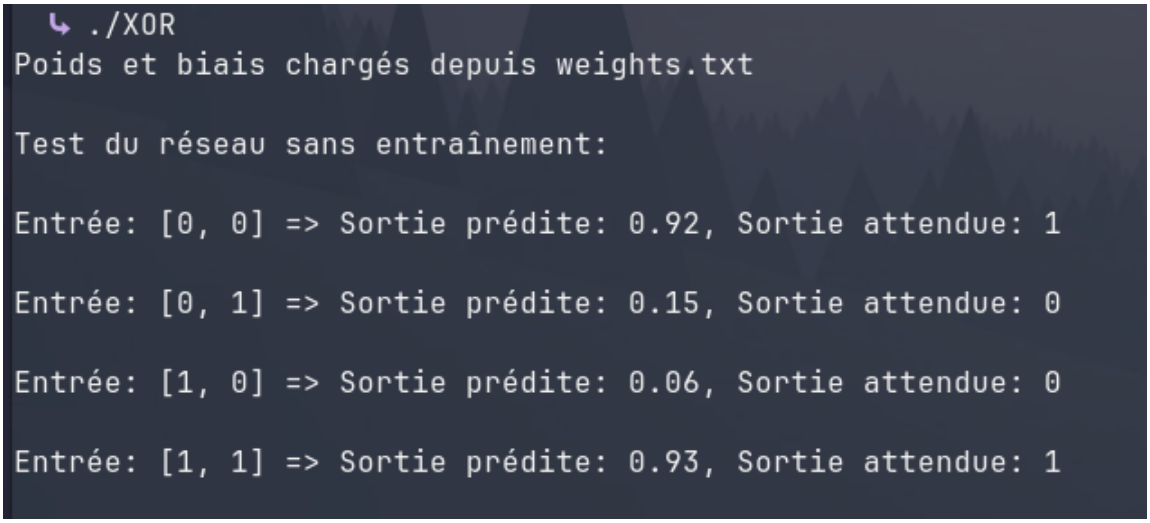

FIGURE 2 – Neural Network

5 Conclusion

5.1 État actuel

À ce jour, nous pensons être en mesure de terminer le projet dans les délais impartis. Le léger retard accumulé sur certaines tâches pourra être rattrapé sans difficulté. À l'heure actuelle, aucun problème majeur de développement ou d'organisation n'a été rencontré.

5.2 Étapes à venir

Dans les prochaines étapes, nous allons implémenter une rotation automatique des images. Cette automatisation facilitera le traitement d'image et améliorera la fluidité de l'expérience utilisateur.
Nous allons également adapter notre réseau de neurones, qui est pour l'instant uniquement capable de résoudre des opérations logiques. Il sera modifié pour pouvoir reconnaître de petites images de lettres de 28x28 pixels. Nous entraînerons ce réseau avec le jeu de données EMNIST (Extended MNIST), qui contient plusieurs images de lettres au format 28x28 pixels.
Une autre étape consistera à créer une interface graphique plus agréable et complète. Nous ajouterons diverses options de paramétrage, notamment pour le traitement de l'image, comme les seuils de détection, le contraste, etc.
Enfin, si le temps le permet, nous aimerions également améliorer notre solveur pour qu'il soit compatible avec des graphes reconnaissant les mots ayant des préfixes similaires. Cela rendra le solveur bien plus performant.