Rapport IN104 - Wordle

par Sébastien Kerbourc'h et Adrien Wallon

I. Coder le jeu wordle

La première partie de ce projet consistait à créer le jeu wordle en console. Dans ce jeu, il faut trouver un mot de 5 lettres en 6 tentatives. À chaque tentative, le programme donne des indices sur les lettres correctes, incorrectement placées ou n'existant pas dans le mot.

Nous avons commencé par choisir un format particulier de dictionnaire : ce sont des fichiers textes, chaque ligne correspond à un mot (de n'importe quelle taille) et les mots sont triés par ordre alphabétique. Chaque langue à son propre fichier. Pour simplifier la gestion de la lecture et des entrées utilisateurs, nous avons fait le choix de ne prendre des mots constitués que de lettres minuscules de a à z. En soi, la majeure partie du programme pourrait être rendue compatible avec d'autres caractères (comme les accents, ou même d'autres alphabets) sans trop de modifications. Toutefois, ce n'est pas sur cela que nous nous sommes concentrés.

Ce formatage de dictionnaire rend facile le chargement de dictionnaire. On procède de la manière suivante : on demande à l'utilisateur de choisir une langue (ce qui correspond à un choix de fichier), on demande le nombre de lettres avec lequel il veut jouer, on peut alors pour chaque ligne vérifier la taille du mot et si elle correspond, on ajoute le mot à un tableau. Le tableau de mot correspond alors au dictionnaire avec lequel le joueur va jouer. Ce dictionnaire est aussi trié par ordre alphabétique, l'avantage principal étant qu'il est alors possible de faire une recherche dichotomique pour vérifier l'existence d'un mot (ce qui est nécessaire lorsque le joueur fait une tentative). La recherche dichotomique est intéressante au vu du nombre de mots qui peuvent être chargés.

Une fois le dictionnaire chargé, un mot est choisi aléatoirement : ce sera le mot à deviner. Ceci a un inconvénient important : il est courant de tomber sur des mots très peu connus. Pour régler ce problème, nous avons pensé à deux solutions (bien que nous n'ayons pas eu le temps de les implémenter) : il faudrait soit utiliser un dictionnaire de mot plus usuels (mais donc avec moins de possibilités, car plus petit), soit tirer le mot dans une sous-liste de mot, tout en conservant le dictionnaire complet pour jouer.

Enfin, lorsque le mot a été choisi par le programme, le joueur peut commencer à jouer. Il reste ainsi à pouvoir comparer le mot à trouver et le mot donné par l'utilisateur. Ceci se passe en deux temps, on commence par repérer toutes les lettres bien placées. Puis, on cherche si certaines lettres sont présentes dans le mot donné par l'utilisateur, mais mal positionnées. Durant ces deux étapes, on garde en mémoire (pour chaque lettre des deux

mots) si elle n'a pas déjà été assignée une validité (à savoir : bien placée ou existante, mais mal placée). Cela permet d'éviter des erreurs dans les indices affichés. Par exemple, si le mot à trouver est arbre et que le mot de l'utilisateur est acaba cela permet d'éviter que le 2d et 3e "a" soient en jaune (puisque seul le 1er a doit être en vert).

II. Les différentes intelligences artificielles

La seconde partie du projet avait pour but de nous faire créer une intelligence artificielle capable de jouer au jeu (et si possible de gagner).

La 1re chose faite est un algorithme capable, à partir d'un indice donné en jouant un mot, de réduire le dictionnaire aux seuls mots possibles. Ceci a été fait avec l'aide de la fonction de comparaison entre le mot à trouver et le mot donné par l'utilisateur. En fait, pour chaque mot du dictionnaire, on regarde quel indice on aurait obtenu si c'était ce mot qui devait être trouvé. Ensuite, on compare cet indice avec celui réellement obtenu : s'ils sont identiques, alors ce mot est encore possible sinon il ne l'est pas.

A. Par fréquence des lettres

La première idée a été de créer une intelligence artificielle qui se base sur la fréquence des lettres en français. Chaque lettre à un score attribué en fonction de sa facture. Ensuite, il y a deux versions à ce robot : on peut sommer les scores ou les multiplier. Le meilleur mot est celui qui a le plus grand score. La version avec la somme a tendance à favoriser les mots avec beaucoup de e ou de a (les lettres très fréquentes) tant dis que les lettres peu fréquentes ont peu d'impact sur le score. Le calcul est assimilable à une moyenne des scores individuels des lettres du mot.. Pour le produit, l'analyse est plus subtile : les lettres fréquences vont beaucoup favoriser un mot alors que les lettres peu fréquentes vont activement le défavoriser. Un avantage très clair de cet algorithme est qu'il est très rapide à calculer.

B. Entropie

Notre 2d automate utilise une (petite) formule liée à la théorie de l'information. L'idée sous-jacente est qu'un événement très probable a beaucoup de chances d'arriver, mais qu'il n'importera que très peu d'informations. En effet, prenons un cas extrême : on considère que l'on va tester un certain mot et que l'on va obtenir un indice qui nous permettra d'être sûr du mot à trouver. Ce cas a une probabilité d'arriver de l'ordre de 1/nombre de mots (puisque les mots sont choisis aléatoirement). Donc bien qu'il apporte énormément d'information, il est peu probable qu'en jouant ce certain mot, on tombe sur cet indice : on tombera vraisemblablement sur un autre indice. Tout le but de l'entropie est de quantifier ce rapport entre information gagnée et probabilité d'arriver.

Lors de l'implémentation de l'entropie , nous avons d'abord eu un problème dans la fonction qui testait la compatibilité des mots avec les mots et indices déjà obtenus. Cela était dû au fait que la fonction de test de compatibilité n'exploitait pas toute l'information obtenue grâce aux indices. L'utilisation de celle déjà réalisée pour la vérification des entrées utilisateurs a permis de corriger le problème.

III. Interface graphique

L’interface terminal -bien que lacunaire- se base sur celle du wordle traditionnnel : les lettres bien placées sont en vert, les mals placées en jaune et le reste en rouge. Des messages accompagnent le joueur tout du long.

L’objectif futur serait de rendre l’interface plus jolie avec termbox2 ou ncurses par exemple.