Rapport projet LINFO1104

Avril 2023

1 Introduction

Dans le cadre du cours LINFO1104, nous avons créé un programme capable de prédire le mot suivant dans une phrase donnée. Pour cela, nous possédons une base de donnée de 208 fichiers textes composés de tweets sur lesquels nous nous basons pour effectuer nos prédictions.

2 Notre programme

Notre programme répond à 4 commandes:

- **make**, qui est la première commande que l'utilisateur doit lancer et permet de compiler le projet.
- **make run**, qui lance le programme et ouvre l'interface graphique qui permet d'essayer notre programme. Lorsque l'analyse des fichiers est en cours, la phrase "Loading... Please wait." est affichée, une fois que celle-ci disparait, l'utilisateur peut utiliser le programme pour effectuer des prédictions.
- **make clean**, qui supprime les fichiers compilés.
- **make tests**, qui lance les tests de nos fonctions, en affichant true pour chaque test réussi et false si un test échoue. Nous avons utilisé la technique dite "Test driven development" pour ce travail, ce qui signifie que nous écrivions d'abord les tests des fonctions avant de les coder. Cette commande a donc été créée au début du projet, et nous a servi tout au long du travail.

3 Implémentation

Nous expliquons ici l'implémentation de l'arbre de données qui permet de savoir quelles sont les suites de mots qui ont été croisées et à quelle fréquence.

L'arbre principale contient des noeuds appelés "Tree", qui possèdent chacun 4 attributs:
-string : Le mot auquel il correspond
-right: Un autre noeud de même type dont le string se trouve plus bas dans l'alphabet
-left: Un autre noeud de même type dont le string se trouve plus haut dans l'alphabet
-subtree: Un noeud appelé "Leaf", qui commence un sous-arbre.

Les sous-arbres contenant des noeuds "Leaf" sont des arbres qui contiennent les mots croisés à la suite du string du noeud principal. Chaque noeud possède également 4 attributs:
-string: Le mot auquel il correspond
-right: Un autre noeud de même type dont le string se trouve plus bas dans l'alphabet
-left: Un autre noeud de même type dont le string se trouve plus haut dans l'alphabet
-value: Un record, dont chaque champ correspond à un mot croisé à la suite du duo de mot auquel correspond le noeud. Les champs portent donc comme noms les mots croisés, et donnent le nombre d'occurence de chaque mot.

4 Extensions

Pour utiliser nos extensions, nous avons ajouté des arguments au programme et inclu ceux-ci dans le Makefile. Ceux-ci se trouvent dans la variable "EXTENSIONS" du Makefile. Chaque argument correspond à une extension et permet de l'activer. Il suffit de modifier l'argument à "true" pour activer les extensions désirées. Les 4 extensions sont nommées "custom_dataset", "history", "automatic" et "more_gramme". Nous avons également un argument en plus, "better_parse", qui permet une analyse plus rigoureuse des fichiers.

4.1 Ajout de bases de données custom

Cette extension permet à l'utilisateur d'ajouter une base de données, ainsi que de choisir les bases de données qu'il veut utiliser. Pour cela, nous avons créé l'onglet "Dataset" qui propose 3 fonctionnalités :

- "Add Dataset", qui permet donc d'entrer le nom d'un dossier contenant des fichiers textes, et ce dossier sera alors utilisé pour les prédictions. Le dossier doit se situer dans même répertoire que le programme pour être accepté. Notre programme vérifie si ce dossier existe bel et bien avant de l'accepter, sinon il prévient l'utilisateur que ce dossier n'existe pas.
- "Select Datasets", c'est ici que l'utilisateur peut sélectionner les bases de données qu'il désire utiliser pour sa prédiction. Par défaut, la base de donnée est sélectionnée sur "Default", qui est donc le dossier nommé "tweets" proposée par le professeur. On peut remarquer qu'il existe au préalable une base de donnée "History", correspondant à l'historique, que nous développerons dans la prochaine extension. Afin de tester cette fonctionnalité, nous avons mis à votre disposition un dossier nommé "smaller_data", qui contient uniquement les deux premiers fichiers textes du dossier Tweets.
- "Reset Datasets". Comme son nom l'indique, elle permet de remettre à zéro toutes les modifications faites sur la base de donnée, et de réinitialiser à la version par défaut.

Il est important de noter que lorsqu'on choisit les bases de données à utiliser, il est nécessaire de relancer le programme grâce au bouton "Reload", qui relancera l'analyse des fichiers avec les bases de données choisies.

En ce qui concerne l'implémentation, pour retenir quel dataset est utilisé, nous stockons ces informations dans un fichier "save/custom_dataset". Dans ce fichier, à chaque ligne se trouve une entrée "Nom dossier true/false" qui nous permet de savoir si la base de donnée est sélectionnée ou pas. Default et History sont 2 entrées spéciales, comme expliqué plus haut. Lorsque l'utilisateur ajoute ou sélectionne des bases de données, ce fichier est modifié afin de sauvegarder les modifications.

4.2 Historique des inputs de l'utilisateur

Sur notre interface est placé un bouton "Save". Ce bouton permet, lorsqu'on l'actionne, de sauver l'input écrit dans le cadre par l'utilisateur. Cela crée donc un fichier texte, qui sera utilisé lors des prochaines prédictions. Pour activer l'utilisation de l'historique, il suffit d'aller dans l'onglet "Dataset", ensuite dans "Select datasets", et d'ensuite cocher la case "History". Une fois cette option activée, l'historique enregistré par le bouton "Save" sera utilisé comme base de donnée pour les prochaines prédictions. Comme pour les bases de données personalisées, il est nécessaire de relancer le programme à l'aide du bouton "Reload".

4.3 Proposition automatique (tier 1)

Cette extension est capable de proposer un mot simplement en appuyant sur la barre espace, et ce en s'imprimant directement dans l'encadré d'écriture de l'utilisateur. Elle diffère donc de la fonctionnalité de base "Predict" qui ne fait que proposer le mot le plus probable dans l'encadré de sortie du programme. Le mot proposé est celui qui a la plus grande probabilité d'être la suite des deux mots précédents, par conséquent, il est nécessaire d'avoir écrit au moins deux mots pour qu'une proposition s'affiche. Si le mot proposé ne convient pas à l'utilisateur, la touche Backspace du clavier permet de supprimer le dernier mot proposé, ou rentré par l'utilisateur.

4.4 Amélioration de l'interface

Afin de rendre l'interface plus agréable et utiliser facilement les extensions, nous avons ajouté des boutons et un menu, comme cela se fait sur de nombreux programmes. Nous avons également ajouté des boites de dialogues qui informent l'utilisateur sur les actions effectuées.

4.5 Proposer plus d'un N-gramme à l'utilisateur

Cet extension activée grâce au paramètre "more_gramme" permet à l'utilisateur d'avoir plusieurs propositions dans la sortie du programme. Celle-ci s'affiche par ordre de probabilité, en allant de la plus haute à la plus faible. Nous proposons par défaut les 5 mots les plus probables, si l'utilisateur en veut plus, il faut ajuster la variable globale NberWord dans le fichier main.oz au nombre désiré.

4.6 Amélioration du parsing

Nous avons voulu proposer une analyse plus précise des fichiers, en formattant les mots plus efficacement. Lorsque l'extension "better_parse" est activée, les hashtags, "@" ainsi que les "-" au début d'un mot sont retirés. Les parenthèses, les crochets, les points d'exclamation et d'interrogation ainsi que les guillemets sont considérés comme des débuts et des fins de phrase. Enfin, les majuscules sont considérées comme identiques aux minuscules.