UCLouvain

École polytechnique de Louvain

LINFO1104 - Concepts des langages de programmation

Projet TwitOZ : un prédicteur de texte

Groupe 15

Auteurs :
Delsart Mathis
Kheirallah Cédric

Enseignant :
Van Roy Peter

Assistants :
Crochet Christophe
Sprockeels Damien
Wirtgen Thomas

2022-2023

1 Concept du projet

Le projet TwitOZ consistait en l'implémentation d'un prédicteur de texte codé dans le langage Oz et devait être capable de prédire le mot suivant l'input de l'utilisateur en se basant sur les N derniers mots de celui-ci.
L'implémentation de l'application a nécessité l'utilisation de multiples savoirs vus au cours LINFO1104 tels que les threads, les fonctions récursives terminales, les structures récursives tels que les arbres, les listes, etc.

2 Choix de l'implémentation

2.1 Structure de l'arbre

Afin de stocker les résultats, nous avons décider d'implémenter un arbre binaire ordonnée ayant comme clés les différents mots N-Gramme et comme valeurs des sous-arbres binaires ordonnées ayant quant à eux comme clés des fréquences et comme valeurs des listes des mots ayant cette fréquence d'apparaître après le mot N-Grammes. La structure récursive de l'arbre binaire est la suivante : obtree = tree(key :Key value :Value tleft :obtree tright :obtree) | leaf. Nous avions choisir de faire des sous-arbres afin de facilement accéder à l'élément de plus grande fréquence (= élément le plus à droite possible) et également afin de faire des recherches de valeurs en 0(log(n)) et non en 0(n).

3 Extensions implémentées

3.1 Généralisation de la formule du N-gramme (∗∗)

Notre généralisation de la formule du N-gramme fonctionne pour tout N >= 1. Pour l'effectuer, nous parcourons tous les éléments d'une liste de mot et pour chaque élément, nous appliquons une concaténation (avec comme délimiteur un espace) des N-1 mots suivants. L'algorithme s'arrête évidemment après Length List - N + 1 itérations puisque il n'y a plus N éléments qui suivent.

3.2 Ajouter des bases de données custom (∗∗)

Nous avons implémenté la possibilité pour l'utilisateur de modifier la base de données sur laquelle se base le programme de prédiction afin que les nouvelles données soient prisent en compte lors de la prochaine prédiction.
Pour offrir plus de flexibilité dans la gestion des fichiers, on peut également interagir avec des fichiers texte déjà présent sur l'ordinateur ou même en créer depuis TwitOZ. En résumé, l'utilisateur peut :

- ✓ sauvegarder le texte écrit en input dans la base de données
- ✓ sauvegarder un fichier texte de son ordinateur dans la base de données
- ✓ sauvegarder le texte écrit en input dans un fichier texte sur l'ordinateur à l'emplacement de son choix
- ✓ charger le contenu d'un fichier présent sur l'ordinateur dans l'input de l'application

3.3 Garder un historique des inputs de l'utilisateur (∗∗)

Dans la même idée que l'extension précédente, l'utilisateur peut sauvegarder l'input actuel pour pouvoir le récupérer lors du prochain lancement du programme et continuer ses prédictions. Nous avons également offert la possibilité de supprimer l'historique directement depuis l'interface utilisateur pour ne plus prendre en compte les anciennes données dans les prochaines prédictions.

3.4 Proposition automatique (∗ ∗ ∗)

Si l'utilisateur choisit d'activer cette extension, la prédiction se fait automatiquement au fur et à mesure de l'écriture de l'input (toutes les 0.5 secondes exactement) et ce sans avoir besoin de cliquer sur un bouton.
L'extension a aussi été améliorer afin de compléter le mot en train d'être écrit par l'utilisateur. Par exemple : entrer "to see the gr" en input proposera "great" afin de compléter le dernier mot.

3.5 Proposer plus d'un N-gramme à l'utilisateur (∗)

L'utilisateur peut manuellement choisir le N-gramme qu'il souhaite utiliser lors du démarrage du programme. Celà doit être au minimum un 1-gramme et n'a pas de maximum puisque nous avons généralisé la formule du N-gramme.

3.6 Proposer la correction de mots dans une phrase déjà existante (∗ ∗ ∗)

Si cette extension est activé alors l'utilisateur peut entrer un mot dans la boîte de correction adjacente et le programme relancera une prédiction sur chaque mot correspondant dans l'input en prenant en compte les N mots précédents ceux-ci (selon le N-gramme choisi par l'utilisateur). Par exemple : "I am going there so I must stop there". Une demande de correction sur "there" avec un Bi-gramme donnera "to" pour le 1er "there" et "the" pour le 2ème "there".

3.7 Améliorer l'interface graphique existante (∗)

Nous avons rendu l'interface graphique plus agréable en y ajoutant des couleurs contrastées dans l'effet bleu/noir ressemblant aux couleurs de ChatGPT et une couleur de fond bleu ciel, couleur officiel de Twitter (vu que les données sur lesquelles nous avont travaillés sont des tweets). Nous y avons également ajouté une image du logo de Twitter avec le mot Oz à l'intérieur afin d'évoquer la coopération entre les tweets de la base de données et le code dans le langage de programmation Mozart.
Nous avons également mit un titre et des boutons lisibles et réactifs à la souris passant dessus. Les actions de ces boutons sont expliqués dans les extensions ci-dessus.

3.8 Modifier le Makefile pour pouvoir spécifier les extensions (∗)

Finalement, nous avons amélioré le Makefile fourni de base afin de laisser à l'utilisateur le choix des extensions qu'il souhaite utiliser pour ses prédictions ainsi qu'une commande d'aide pour avoir un aperçu de toutes les commandes et options disponibles.