LINGI1104 - Twit'Oz

Groupe AY : Christopher Castel 22421900, Martin D'Hoedt 47751500

1 Choix d'implémentation

1.1 Main

1.1.1 Paramétrisation

— L'utilisateur peut choisir le nombre de fichiers à traiter. Les fichiers doivent respecter le format `part_N.txt` où N est un entier positif.
— Le nombre maximum de threads est également personalisable en fonction des spécifications de la machine utilisateur.

1.1.2 Démarrage

Une threadpool est initialisée avec le nombre de threads spécifié par l'utilisateur. Une liste de jobs, procédures comprenant un thread pour le parsing et un thread pour la lecture d'un fichier, est ensuite créée. Les jobs et la threadpool sont finalement fournis en paramètres à la fonction "barrière" qui exécute la lecture et le parsing des fichiers dans des threads séparés.

1.1.3 Threadpool

L'implémentation a été réalisée via une liste dont le dernier élément est "unbound", ce qui permet de rendre la liste circulaire afin de recycler les threads.

Les threads de la pool sont dits "dédiés", car ils ne sont créés qu'à l'initialisation de cette dernière. Chaque thread est implémenté via un port object qui accepte le message `start(Job Unit)`. Il aurait été envisageable d'ajouter un message `stop` afin de mettre fin à l'appel récursif et ainsi tuer le thread. Le message `start` contient une procédure (Job) et un callback (Unit). À la réception du message, la procédure est exécutée puis le callback est finalement lié à 'unit' afin de prévenir le "client" de la fin de l'exécution du job.

1.1.4 Barrière

Lors de l'exécution au sein de la barrière les threads sont consommés un par un jusqu'à ce que la liste soit épuisée. Un job est "send" à un thread disponible qui exécute alors ce dernier. La barrière "wait" sur le callback afin de pouvoir réinsérer le thread dans la threadpool. Chaque thread est ainsi créé une seule fois, puis utilisé et recyclé au fur et à mesure de l'exécution du programme. Lorsqu'il n'y a plus de job à exécuter, la barrière "wait" sur l'ensemble des jobs toujours en cours d'exécution, puis rend finalement la main au thread principal qui démarre le GUI.

1.2 Reader

Le reader comporte une seule fonction permettant la lecture d'un fichier. La fonction construit au fur et à mesure une liste des lignes lues.

1.3 Parser

1.3.1 Paramétrisation

— `ToReplace`, une liste de tuples composés d'un mot à remplacer et du mot qui le remplace.
— `BreakList`, une liste de caractères qui marquent la fin d'une phrase.
— `ToRemove` (pas utilisé)

1.3.2 Exécution

Le contenu d'un fichier est parsé ligne par ligne. Chaque ligne est "sanitizée" en :
— ignorant les caractères qui ne seraient pas définis dans le charset ISO8859.
— restreignant les caractères valides à un subset du charset ISO8859, à savoir ceux compris entre [32, 126].
— "trimmant" les espaces pour ne garder qu'un espace maximum entre deux mots.
— convertissant les caractères en minuscule.

Toutes les lignes sont ensuite décomposées pour construire une unique liste de mots. Ces mots sont éventuellement remplacés par d'autres (e.g. "&" qui est remplacé par "&"). Ils sont rassemblés en phrases afin de séparer sémantiquement les mots terminant et commençant une phrase. Ils sont finalement envoyés au dictionnaire de prédiction.

1.4 Prediction dictionnary

Le dictionnaire, servant de base de données de mots, est implémenté via un port object afin d'être thread-safe. La structure de données utilisée est un dictionnary de dictionnaries : `Dictionnary(mot(s), Dictionnary(prédiction, fréquence))`. La clé `mot(s)` du premier dictionnary correspond à un mot ou deux mots séparés par un espace, ce qui permet facilement de construire le 1-gramme et le 2-gramme. En valeur du premier dictionnary se trouve un second dictionnary qui contient les `prédictions` en clé et leur `fréquence` en valeur. Le choix du second dictionnary est motivé par la rapidité d'update une fréquence d'une `prédiction` durant le parsing. Grâce à la `fréquence`, le dictionnaire peut renvoyer les N mots les plus fréquents pour un/des `mot(s)` donné(s).

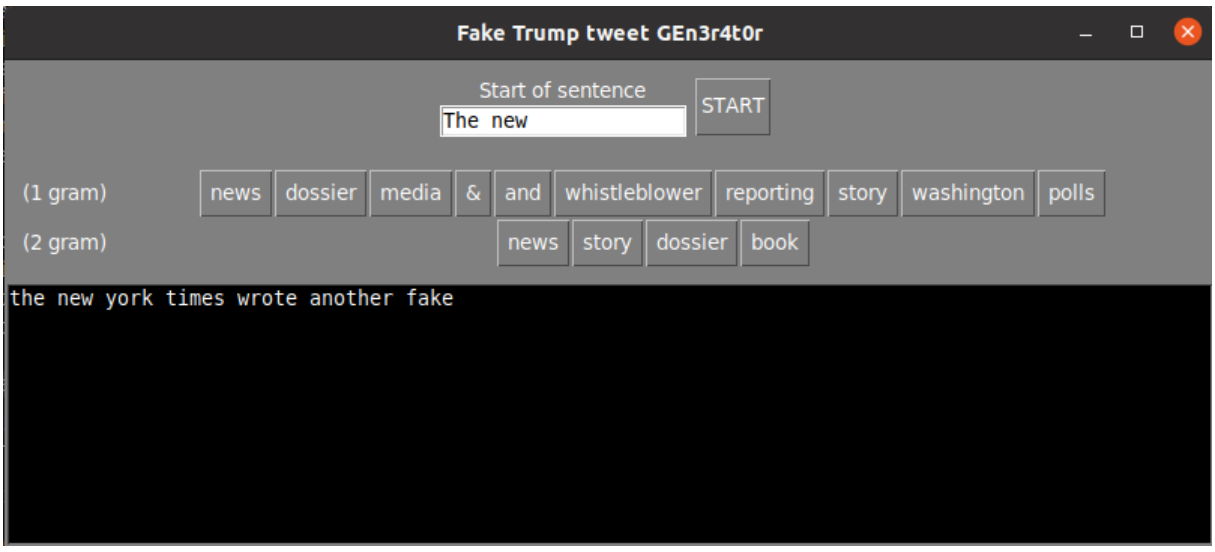
1.5 GUI

1.5.1 Paramétrisation

— `PredictionRange`, permet de définir le nombre de prédictions à générer.

1.6 Fonctionnement

Le début de la phrase est introduite dans un widget "text" et est soumise au dictionnaire en cliquant sur le bouton "START". Un message d'erreur s'affiche si l'utilisateur initialise la prédiction avec 0 ou plus de 2 mots. Les prédictions suivantes sont effectuées via les boutons propres aux prédictions "1-gramme" et "2-gramme". Un widget "placeholder" est utilisé afin de (re)générer les boutons à la volée. Lorsque l'utilisateur sélectionne une prédiction, elle est ajoutée au tweet en construction.



1.7 Solution alternative

Une seconde solution pour la gestion du threading a été développée sur une autre branche du projet. Dans cette version, le main crée des batches de fichiers. Pour chaque batch, un thread s'occupe du parsing et un thread s'occupe de la lecture. Le reader y est également modifié : plutôt que de lire un seul fichier, il en lit plusieurs.