LINGI1131- Project 2020
Twitt'Oz
Groupe BB

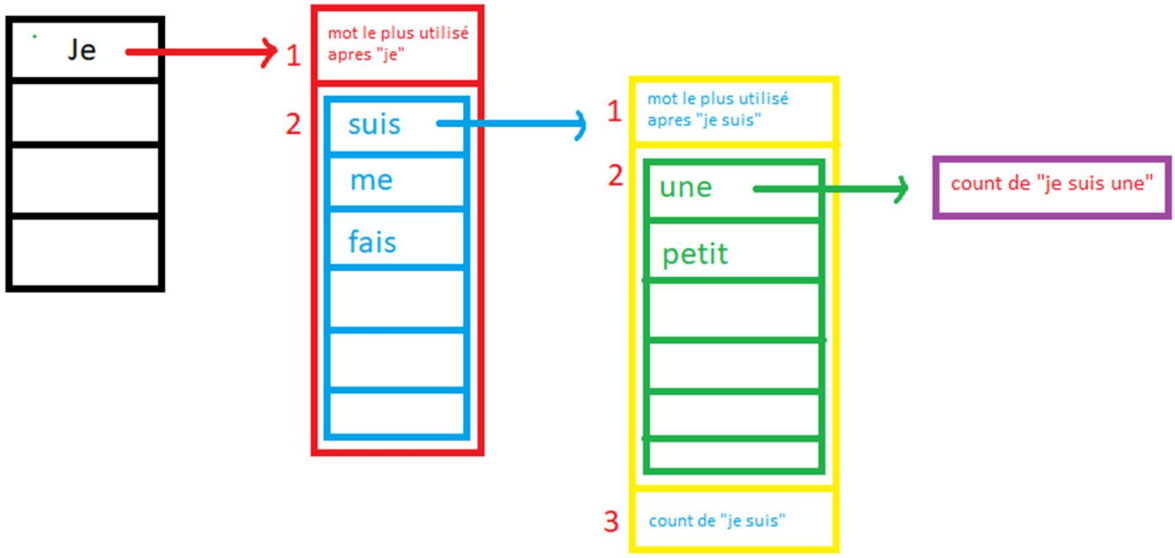
LADERRIERE Loïc (NOMA 4426-19-00)

PUCHE Fabian (NOMA 4442-19-00)

Choix d'implémentation

Dico

Nous avons utilisé des dictionnaires, pour stocker chaque mot, car ils permettent d'accéder à un élément avec une complexité en O(1).
Pour ce qui est de notre **dictionnaire**, nous avons choisi de combiner le 1-gramme et le 2-gramme.
Pour bien comprendre le fonctionnement de notre **dictionnaire** nous allons prendre l'exemple de l'expression « je suis une » qui contient le 1-gramme « je suis » ainsi que le 2-gramme « je suis une ».
Les dictionnaires fonctionnant avec des associations clé/valeur. Le « je » est une clé du **dictionnaire principale** qui a pour valeur un **tuple** qui possède le mot le plus utilisé après « je » ainsi qu'un **autre dictionnaire** contenant tous les mots possibles après « je ». Ce **dictionnaire** a donc lui-même une liste de clé/valeur. La clé « suis » a pour valeur un **tuple** contenant le mot le plus utilisé après « je suis », le nombre de fois que l'expression « je suis » apparait, mais aussi un **troisième dictionnaire** contenant les mots se trouvant après « je suis ». Encore une fois, ce dictionnaire est de la forme clé/valeur, la clé « une » a pour **valeur**, non pas un tuple, mais juste le **nombre de fois** que l'expression « je suis une » apparait.
En faisant cela, il possible pour chaque mot, de chaque o-grammes, de savoir tous les mots possibles ainsi que le nombre fois qu'ils apparaissent et quel est le mot plus souvent employé.
Il se présente sous la forme suivante :



Thread

En ce qui concerne gestion des threads de notre application, il fonctionne de la manière suivante qui est la même qui a vue en cours. Le "main va créer 1 thread saver, 3 threads de reader (Un thread reader se voit attribuer 1 seul fichier). À leur création, ceux-ci vont créer 1 thread parser à qui ils pourront envoyer sur leur port, les phrases se trouvant dans le fichier. Quand le reader aura fini de lire le fichier, il préviendra son reader. Le reader va créer un dictionnaire sur base des phrases envoyer par le reader. Quand il recevra un message de fin de fichier de son reader, il finira de parser les dernières phrases avant de les envoyer sur le port du Main, un message signifiant qu'il arrive en fin de vie. Ainsi le main pourra créer dynamiquement un nouveau thread.
Il peut avoir au maximum 8 threads qui fonctionnent simultanément.

- 3 threads pour le reader
- 3 threads pour le parser
- 1 thread pour le saver
- 1 thread pour le main

Reader

Les threads Reader vont lire chaque lignes de leur fichier et découper en phrases qu'ils enverront sur le port de leur Parser.

Parser

Celui-ci va lire en continu un stream sur lequel sont envoyés les messages de son reader.

Les threads Parser, ayant reçu les lignes des Reader, vont créer un dictionnaire de la forme présentée en début de pdf puis enverront ce dictionnaire sur le port du Saver

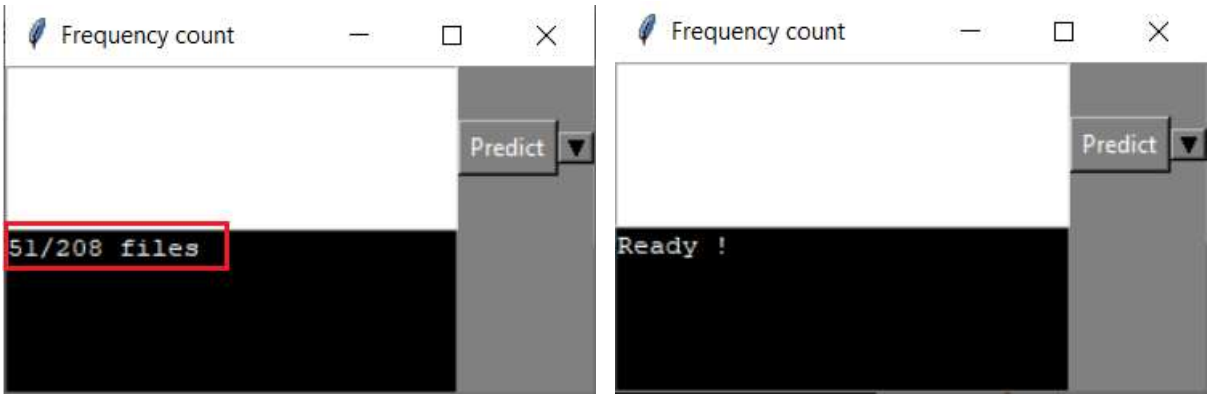
Saver

Celui-ci va lire en continu un stream sur lequel sont envoyés les dictionnaires des Readers.

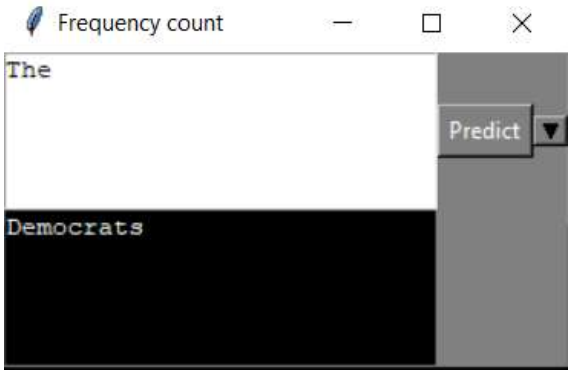
Le rôle du Saver est juste de « merge » chaque dictionnaire reçu par les Saver dans un seul dictionnaire qui sera utilisé pour la prédiction de texte.

GUI

Lors du lancement de l'application, un compteur de fichiers affiche la progression de l'analyse.



En fonction du nombre de mot taper (1 ou 2) et après avoir appuyé sur le bouton « Predict », nous afficheront le mot le plus utiliser (pour le 1-gramme ou 2-gramme).



A droite du bouton « Predict » se trouve une droplist qui propose une liste de mot possible pour le mot entré par l'utilisateur. Celui-ci, peux cliquer sur le mot pour l'ajouter dans l'entrée de texte.