LINFO1103: Analyse de sentiment

5 Avril 2023

Auteur : PREDA Patrick

1 Introduction

En ayant eu 100% au projet de l'année passée, j'ai decider d'ameillorer mon code en utilisant le plus de one-liners possibles mes code sont disponible ici : Repo Github

2 Explications

1. La vérification du type de `forme_parenthesee` (dans creer_arbre()) : a été simplifiée en utilisant `isinstance(forme_parenthesee, str)` au lieu de
   `type(forme_parenthesee) == str`
2. Dans la classe `ArbreBinaire`, la déclaration des attributs `left` et `right` a été simplifiée en les initialisant directement à `None`.
3. Les méthodes `poids()` et `score()` ont été optimisées pour un calcul plus efficace et propre. Par exemple, dans la méthode `poids()`, l'utilisation d'une liste comprehension permet de calculer la somme directement.
4. La fonction `re_arrange()` a été modifiée pour utiliser des regular expressions et ainsi "simplifier" le code.
5. La fonction `wheretocut()` a été modifiée pour être plus claire et efficace en utilisant des variables pour conserver les compteurs et l'index maximal.
6. La fonction `forme_correcte()` a été simplifiée en one-liner pour vérifier si la chaîne commence par une parenthèse ouvrante et se termine par une parenthèse fermante.
7. La fonction `main()` a été modifiée pour être plus concise et claire en utilisant des variables pour stocker les sous-arbres gauche et droit. De plus, elle utilise `allist` (un Dictionnaire 💀) pour stocker les arbres créés. `allist` n'est plus une variable globale dans le nouveau code.

3 UnitTests

voir Github Les tests sont assez explicites. on a une un arbre binnaire `self.arbreDefault` sous forme de parentheses a transformer :
on verifie simplement si la fonction `Score()` appliquée sur l'arbre vaut 17 (la bonne reponse precalculée). Nous faisont de meme pour la fonction `poids()` et `forme_correcte()`