Shape

Description automatically generated with medium confidence

**LOG3430**

**Méthodes de test et de validation du logiciel**

**TP3**

- Hiver 2022 -

Groupe 1

Hugo Demeyere - 1939239

Alexander Ciaciek - 1957252

Antoine Déry - 2049399

Soumis à :

Daphné Lafleur

Hanane Ikhelef

13 mars 2022

1. **Quelle est la taille de la matrice MCA pour la puissance 2, 3, 4 ?**

Pour obtenir les matrices MCA, nous avons d’abord ajouté les 4 paramètres dans ACTS. Par la suite, nous avons généré les matrices pour les puissances 2, 3 et 4 avec l’algorithme IPOG, tel que recommandé par IPOG. La taille des matrices est présentée ci-dessous :

**Tableau 1 : Taille des matrices MCA selon la puissance**

|  |  |
| --- | --- |
| **Puissance** | **Taille de la MCA**  (Rangées X colonnes) |
| 2 | 8x4 |
| 3 | 16x4 |
| 4 | 32x4 |

1. **Quel est l’*accuracy* du système pour chaque cas de test ?**

Une fois l’implémentation des différentes fonctions effectuées, nous avons exécuté le système RENEGE pour obtenir le résultat des différents paramètres d’analyse. Les résultats d’*accuracy* de chaque cas de test sont présentés dans le Tableau 2.

**Tableau 2 : *Accuracy* du système selon le cas de test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **log\_calculate** | **log\_combine** | **Frequency** | **Stemming\_disable** | **Accuracy** |
| Vrai | Vrai | 1 | Vrai | 0.61 |
| Vrai | Faux | 1 | Faux | 0.09 |
| Faux | Vrai | 1 | Faux | 0.62 |
| Faux | Faux | 1 | Vrai | 0.64 |
| Vrai | Vrai | 2 | Faux | 0.61 |
| Vrai | Faux | 2 | Vrai | 0.12 |
| Faux | Vrai | 2 | Vrai | 0.62 |
| Faux | Faux | 2 | Faux | 0.47 |
| Vrai | Vrai | 3 | Vrai | 0.61 |
| Vrai | Faux | 3 | Faux | 0.14 |
| Faux | Vrai | 3 | Faux | 0.62 |
| Faux | Faux | 3 | Vrai | 0.36 |
| Vrai | Vrai | 4 | Vrai | 0.00 |
| Vrai | Faux | 4 | Faux | 0.20 |
| Faux | Vrai | 4 | Faux | 0.62 |
| Faux | Faux | 4 | Vrai | 0.36 |

1. **Quelle configuration donne les meilleurs résultats ?**

Pour effectuer l’analyse, nous avons exporté les résultats obtenus suite à l’exécution du programme. Autrement dit, ce qui se retrouvait à être affiché dans le terminal à été sauvegardé dans un fichier texte nommé *results.txt*. Ce fichier est remis avec ce rapport et le code.

Pour supporter nous analyse, nous avons utilisé les définitions suivantes :

Précision =

Rappel =

Où *i* est la classe *spam* ou *ham.*

Ainsi, plus ces deux valeurs sont élevées, plus le système est fiable. Nous avons donc recherché le cas de test ayant la meilleure *accuracy*, précision et rappel. La configuration ayant le meilleur résultat est « False, False, 1, True ». Donc, un système d’utilisant pas les formules avec les logarithmes, limitant la fréquence des mots à une seule occurrence et qui ne fait pas le nettoyage des mots serait le plus fiable.

1. **Quels sont les défauts trouvés dans le système ?**

Nous avons identifié quelques défauts dans le code déjà existant. D’abord, quelques fonctions sont implémentées, mais ne sont pas utilisées. Il en va de même pour certains paramètres de fonctions qui ne sont jamais utilisés. Nous avons corrigé ces points.

De plus, nous avons eu une certaine difficulté à gérer les logarithmes puisque Python interprète les nombres flottants (*float*) et les nombres décimaux (valeur retournée par la fonction log10()) de manière différente. Il a donc fallu s’adapter.

Finalement, le cas de test « True, True, 4, True » a posé problème puisqu’une division par zéro était effectuée dans le calcul des paramètres de performance. Nous avons donc adapté le code en conséquence.