RIW

Analyse de la performance

# Traitement

## Rapidité de l’indexation

J’ai choisi de faire un index plutôt lourd mais unique avec l’ensemble des informations nécessaire aux 3 modèles implémentés. Ce choix est guidé par le fait que la création de l’index est une tâche unique sur des collections fixées comme proposé dans l’article. Egalement par cette méthode j’arrive à utiliser des résultats de certains modèles pour calculer des coefficients utiles à d’autres modèles et donc à réduire sur le total l’ensemble des calculs effectués.

Au niveau de la structure de données de l’index j’ai choisi un dictionnaire de type récursif, il s’agit d’une astuce que j’ai trouvée sur internet. Il y a probablement une petite perte de performance avec l’utilisation de ce type de dictionnaire par rapport à un dictionnaire classique mais cela permet de gérer plus facilement l’algorithme. Cela permet en effet de définir un type par défaut lors de l’appel à une clef du dictionnaire non préalablement définie.

Mon algorithme génère donc un fichier Json qui est sauvegardé sur le disque et j’ai également créé une fonction permettant de reprendre ce fichier en mémoire sans devoir le régénérer.

Cela donne donc un temps d’indexation sur la base CACM de :

Mon index prend une place de :

## Rapidité de la recherche

Pour tester la rapidité de la recherche j’ai utilisé des requêtes test et j’ai lancé 10 fois la même recherche afin d’obtenir un temps moyen de recherche.

### Modèle booléen

Requête :

Temps moyen de réponse :

Requête :

Temps moyen de réponse :

Requête :

Temps moyen de réponse :

### Modèle vectoriel

Requête :

Temps moyen de réponse :

Requête :

Temps moyen de réponse :

Requête :

Temps moyen de réponse :

### Modèle probabiliste

Requête :

Temps moyen de réponse :

Requête :

Temps moyen de réponse :

Requête :

Temps moyen de réponse :

# Recherche

## Evaluation par banc d’essai

Sur la collection CACM, les requêtes proposées par défaut dans la collection sont plutôt de l’ordre du besoin d’information et non pas de la requête. Pour pallier ce problème j’ai reformulé quelqu’une de ces requêtes pour essayer d’avoir des résultats plus pertinents vu la faiblesse du traitement linguistique que nous appliquons à la requête lors de notre recherche.

# Système