Ce projet de TP vise à vous familiariser avec les procédures (modules) de base en recherche d’information (RI) à savoir : (1) l’indexation des documents, (2) la pondération des termes et (3) l’appariement requête-document. Dans ce TP, on traite la collection CACM, le modèle Booléen, le modèle Vectoriel et l’évaluation des formules du modèle Vectoriel, avec le langage Python.

1. **Phase indexation :**

Il existe 7 index on total, 2 index général pour tous les titres avec leurs textes, 3 index de d’un fichier inversé par fréquences et 2 autres index pondérés a bases des 2 derniers FIF.

* All titles (AT = {Num de document -> titre, …})

Fichier : All\_titles.pickle

Taille : 978 KO

Nombre d’entrée : 3204

Type de structure : dictionnaire de dictionnaire sur python.

Durée d’indexation de la dernière case 3.3800000000007435e-06ms

Utilité : Faciliter l’affichage sur l’application.

* All textes (AR = {Num de document -> texte, …})

Fichier : All\_texts.pickle

Taille : 978 KO

Nombre d’entrée : 3204

Type de structure : dictionnaire de dictionnaire sur python.

Durée d’indexation de la dernière case : 4.529999999999812e-06 seconde.

Utilité : Faciliter l’affichage sur l’application.

* Full Dict (FD = {Num de document -> {(Mot, fréquence), ...}, …}) :

Fichier : Full\_Dict.pickle

Taille : 882 KO

Nombre d’entrée : 3204

Type de structure : dictionnaire de dictionnaire sur python.

Durée d’indexation de la dernière case 4.7599999999980994e-06 seconde

Utilité : Recherche la fréquence des termes par l’ID d’un document.

* Dinv1 (Dinv1 = {(Mot, Num de document) -> fréquence, …}):

Fichier : Dict\_Inv1.pickle

Taille : 1316 KO

Nombre d’entrée : 67529

Type de structure : dictionnaire d’entier sur python.

Durée d’indexation de la dernière case 4.7599999999980994e-06 seconde.

Utilité : Faciliter l’affichage sur l’application.

* Dinv2 (Dinv2 = {Mot -> [(Num de document, fréquence), ...]}) :

Fichier : Dict\_Inv2.pickle

Taille : 706 KO

Nombre d’entrée : 8275

Type de structure : dictionnaire de liste sur python.

Durée d’indexation de la dernière case 4.7599999999980994e-06 seconde.

Utilité : Recherche la fréquences d’un terme à travers tous les documents.

* FDpond (FDpond = {Num de document -> {(Mot, pond), ...}, …}) :

Fichier : Dict\_Inv1Pond.pickle

Taille : 1116 KO

Nombre d’entrée : 3204

Type de structure : dictionnaire de liste sur python.

Durée d’indexation de la dernière case 6.560000000002675e-06 seconde.

Utilité : Recherche la pondération des termes par l’ID du document.

* Dinv2pond (Dinv2pond = {Mot -> {(Num de document, pond), ...}) :

Fichier : Dict\_Inv2Pond.pickle

Taille : 1168 KO

Nombre d’entrée : 8275

Type de structure : dictionnaire de liste sur python.

Durée d’indexation de la dernière case 2.150000000003538e-06 seconde.

Utilité : Recherche la pondération d’un terme à travers tous les documents.

Nous constatons aucun index est parfait, c’est pour cela exactement nous avons utilisés autant d’index que nécessaire pour gagner en temps de réponse.

La difficulté se poserai lors de la mise à jour des ces index sur tous sur l’échelé massive dans le cas d’un vrai moteur de recherche avec des millions de documents.

1. **Modèle booléen :**

* Évaluation de l’expression booléen :

Tous d’abord, nos operateurs on les mit en majuscule et majuscule seulement, le reste de l’expression peut être quoi que soit vu qu’on va les transformer en minuscule (nos index sont en minuscules).

La transformation d’une expression sans parenthèses se fait très normale en remplaçant les opérateurs AND par « \* », le OU avec « + » et le NOT avec « - ».

Dans le cas où il existe les parenthèses on les garde telle qu’elle est.

Le reste de l’expression est remplacé par leurs valeur d’existence, 1 si le mot a au moins une fréquence dans un document, 0 sinon.

L’évaluation se fait grâce a une fonction de python appelée « eval(expression, globals=None, locals=None) ».

Temps de réponse pour requête : 'NOT optimization AND system AND performance' est

0.03847418000000001 seconde.

1. **Modèle Vectoriel :**

Pour ce modèle nous générons deux vecteur de la requête l'un avec les pondérations de chaque mot et l'autre les fréquences de chaque de ces mots et nous avons utilisé les 4 mesure de similarité :

1. -Produit interne.
2. -Coefficient de dice.
3. -Cosinus.
4. -Jaccard.

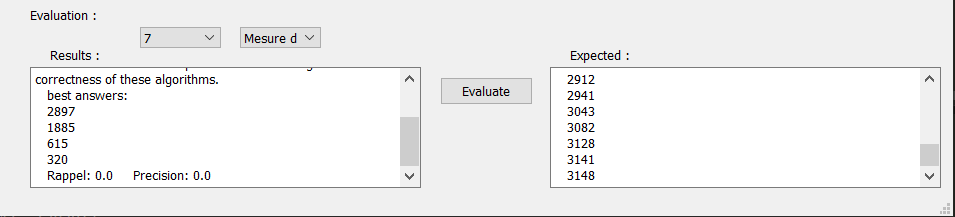
Pour l'évaluation de notre modèle nous avons utilisé les deux importantes performances :

* La précision : qui détermine la value de précision des sélectionné les documents pertinents sans inclure les non pertinents (uniquement les document pertinents).

L’équation de la précision est :

* Le rappel : c'est la valeur de la capacité de déterminer tous les documents pertinents.

L’équation du rappel est :

tel que :(expliquation des variables des l'équations)