UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE

Projet en Recherche d'Information

Eden BELOUADAH 201200006477

Feriel ZOUGGARI 201200007235

Master 2 SII Groupe 1 Année universitaire:2016/2017

Introduction:

Le développement technologique a conduit à une masse importante de données textuelles et visuelles disponibles sous différents formats et venant de plusieurs sources hétérogènes. Le besoin de la recherche d'information sous des conditions pareils a relevé le défit, ce qui a mené à introduire des notions de l'intelligence artificielle qui permettent de rechercher et récolter les informations pertinentes concernant un domaine précis dans un laps de temps le plus réduit possible.

Le but de ce projet est d'utiliser quelques notions simples de la recherche d'information sur un ensemble de documents et cela pour répondre à des requêtes présentées ou bien par l'utilisateur ou bien par l'environnement de test, puis de comparer l'efficacité des différentes techniques pour permettre une meilleure analyse des outils de recherche.

I. Explication des algorithmes:

Dans ce projet, on utilise la collection CACM qui contient plusieurs documents sauvegardés dans un seul fichier CACM.all. Les contenus des documents sont séparés par des marqueurs. Dans le cadre de ce projet, on ne s'intéresse qu'aux marqueurs suivants:

- .I : indique le début du document.
- .T: indique le titre du document.
- .W: indique le résumé du document.

1. Extraction des documents à partir du corpus cacm.all:

On commence par ouvrir le fichier cacm.all et le mettre dans une liste où chaque élément de cette dernière est une ligne du fichier.

Dans une première étape, on cherche le marqueur du début d'un document et on commence à copier le contenu du corpus jusqu'à ce qu'on trouve le prochain marqueur ".I" et on le met dans une autre liste où chaque élément de cette liste représente un document.

Après avoir mit chaque document dans un élément de cette liste, on passe à l'étape suivante.

La deuxième étape consiste à parcourir la liste des documents et pour chaque document on ne garde que le contenu des marqueurs ".T" et ".W", à la fin de cette étape on n'aura que les parties pertinentes de chaque document.

2. Nettoyage des documents:

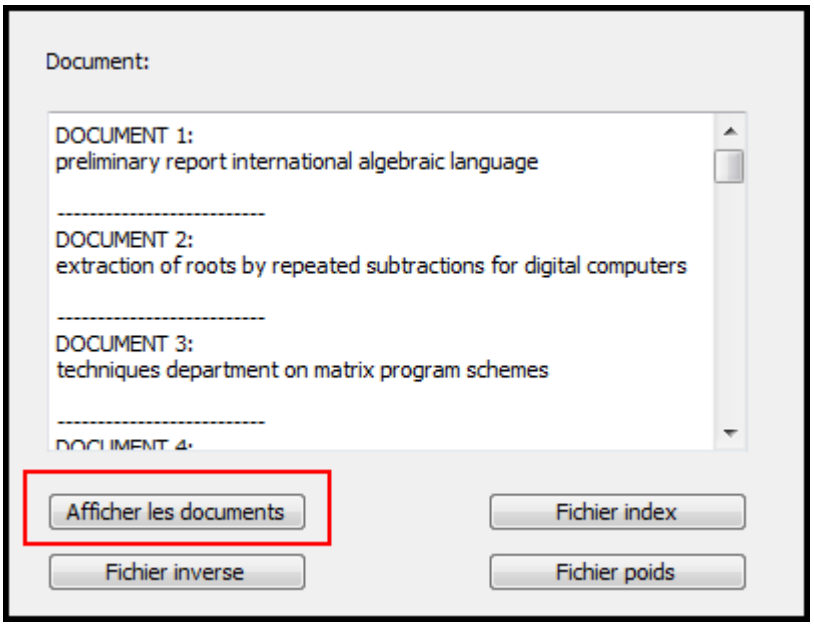
Dans cette partie, on élimine:

- Les séparateurs doublés.
- Les "\n" ajoutés automatiquement pendant la lecture du corpus.
- Tout caractère qui appartient à ListCar, qui est une liste contenant tous les signes de ponctuation et les caractères spéciaux. Par exemple: (, ; ? : !)
- Tous les mots qui appartiennent stopList, qui est une liste contenant tous les mots non significatifs, par exemple: a, an, the, when, for...

3. Tokenisation:

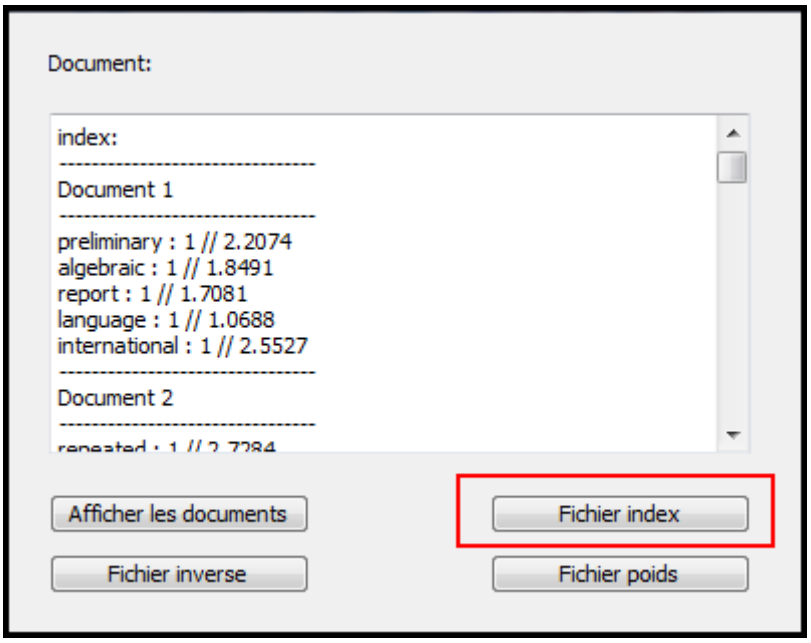
Ce traitement consiste à transformer le document en une liste de mots pour qu'on puisse traiter chaque mot à part.

L'affichage des documents après le prétraitement donne:



4. Indexation:

Dans cette étape, on parcourt la liste des documents et pour chaque document, on affiche les mots qui le constituent et la fréquence d'apparition de chacun d'eux.

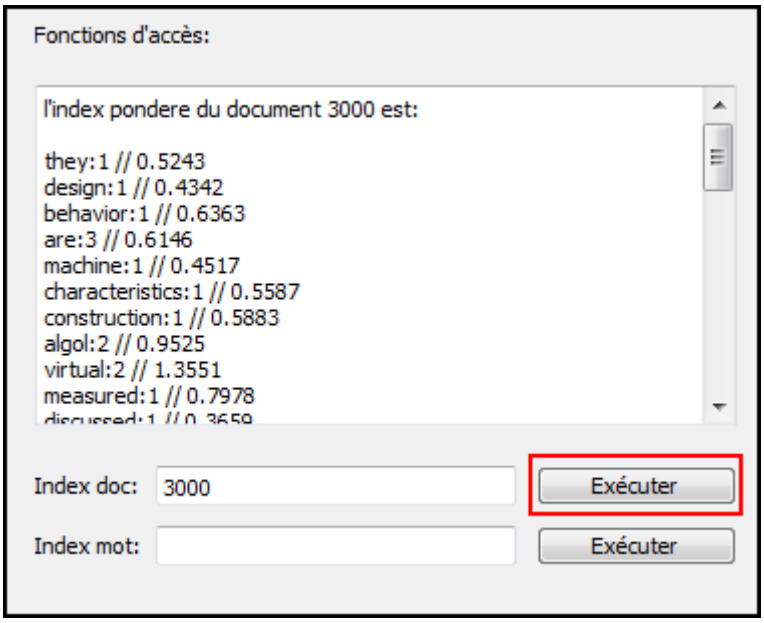


5. Le fichier inverse:

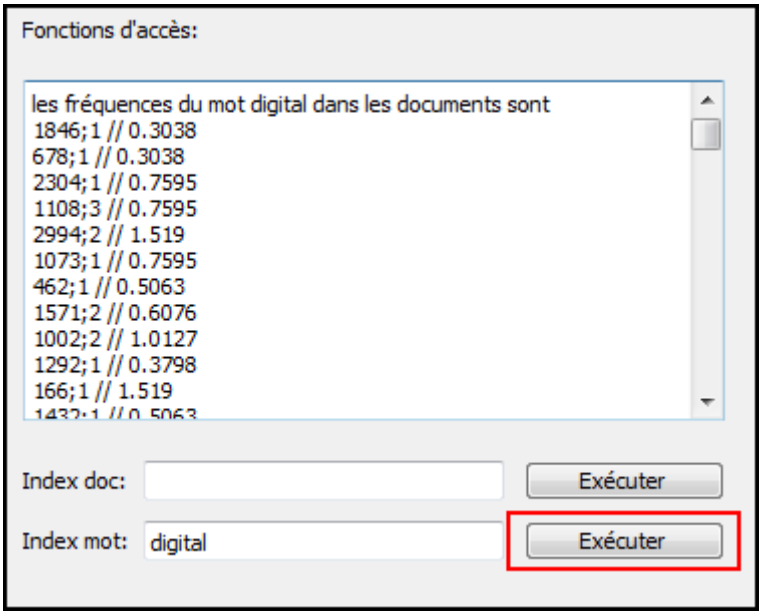
La définition du fichier inverse commence par introduire une matrice où les lignes correspondent aux mots de tout le corpus, et les colonnes représentent les documents, notre algorithme consiste à parcourir les documents et extraire les mots un par un, là où on trouve un nouveau mot, on l'ajoute aux lignes de la matrice et on initialise son nombre de fréquence à 1 avec le document correspondant, sinon (on a déjà rencontré le terme) , on incrémente sa fréquence dans la bonne position dans la matrice.

6. Les fonctions d'accès:

a) *Fonction 1*: Cette fonction reçoit un numéro de document et retourne la liste de mots de ce document avec leurs fréquences. Pour cela on n'a qu'à consulter le tableau d'indexation.



b) *Fonction 2*: Cette fonction reçoit un mot et retourne la liste de documents contenant ce mot avec sa fréquence d'apparition. Pour cela on n'a qu'à consulter le tableau du fichier inverse.



Le premier numéro représente le numéro du document, le deuxième représente la fréquence d'apparition du terme dans le document et le troisième représente le poids du mot dans le document.

7. Modèle booléen:

Dans ce modèle, on représente une requête sous forme d'une expression logique où les termes sont reliés par des connecteurs logiques (AND, OR, NOT). Ce modèle représente un mode d'appariement exact.

Pour se faire, l'utilisateur fait entrer une requête simple ou composée, le rôle de notre programme est de parcourir les documents et pour chaque document, parcourir la requête et remplacer chaque terme par 1 s'il existe dans le document et par 0 sinon. Les opérateurs logiques AND, OR, NOT et les parenthèses ne sont pas concernés par la substitution.

Ensuite, le programme fait passer la nouvelle requête à la fonction **eval** qui évalue l'expression logique. Si **eval** retourne 1 on considère que le document satisfait la requête.

N.B.:

Nous avons traité le cas où les parenthèses dans la requête ne sont pas espacées par rapport aux termes de la requêtes. Par exemple, toutes les requêtes suivantes sont reconnues par notre programme:

- (black and digital) or(report)and white;
- (eden and report) or ((digital or the));
- (there and here) ;
- (stop or (start and not computer)and feriel);
- etc...

Dans le cas où l'utilisateur introduit une erreur syntaxique dans la requête, on la capte et on lui affiche un message d'erreur : Erreur Syntaxique dans la requête.

8. Indexation par pondération TF*IDF:

L'indexation par pondération consiste à affecter un poids à chaque terme du document, pour caractériser leurs importances dans le document.

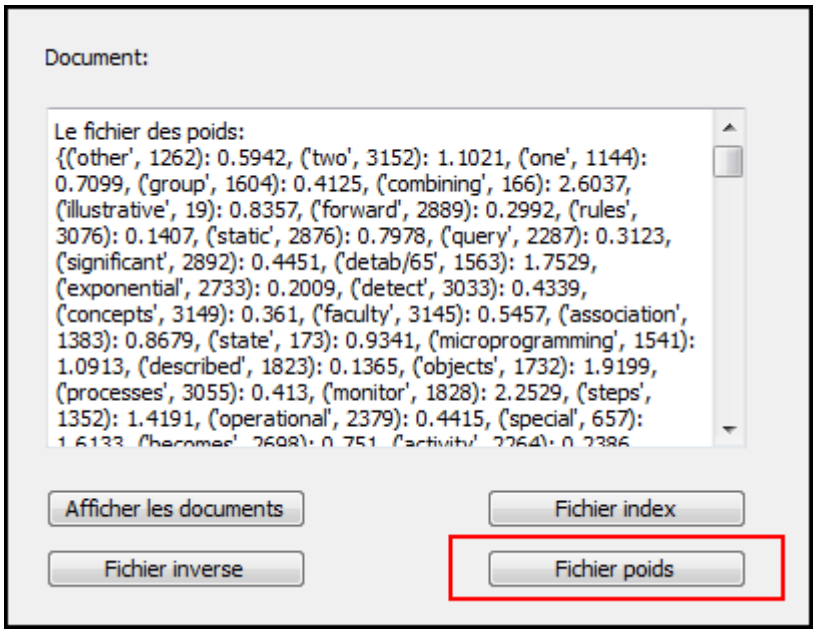
Dans notre programme, on fait le même traitement que celui du fichier inverse sauf qu'au lieu d'afficher la fréquence d'un mot, on affiche son poids, et cela en utilisant la formule suivante:

$$Poid(t_i,d_j)=[Tf(t_i,d_j)/Max(Tf(t_k,d_j))]*\log(N/n_i+1)$$

Où:

- $Tf(t_i,d_j)$: La fréquence du terme t_i dans le document d_j .
- $Max(Tf(t_k,d_j))$: La fréquence du terme le plus fréquent dans le document d_j .

- N: Le nombre total des documents .
- n_i: Le nombre de documents contenant le terme t_i .



9. Modèle Vectoriel:

Dans cette étape, l'utilisateur fait entrer le numéro d'une requête parmi celles qui existent dans le fichier query.text, le rôle de notre programme est d'extraire la requête à partir du fichier et exécuter une des quatre formules du modèle vectoriel que l'utilisateur a choisi puis faire une évaluation de la requête en calculant le rappel et la précision. Les formules associées au modèle vectoriel sont :

- **Formule du produit interne:**

$$SIM(d_i,q_k)=\sum p_{ij} * w_{ik}$$

- **Formule du Coefficient de Dice:**

$$SIM(d_i,q_k)=(\sum p_{ij} * w_{ik} / \sqrt{(\sum p_{ij}^2 * \sum w_{ik}^2)})$$

- **Formule du cosinus:**

$$SIM(d_i,q_k)=2*\sum (p_{ij} * w_{ik})/ (\sum p_{ij}^2 + \sum w_{ik}^2)$$

- **Formule de Jaccard:**

$$SIM(d_i,q_k)=(\sum (p_{ij} * w_{ik})/ (\sum p_{ij}^2 + \sum w_{ik}^2 - \sum(p_{ij}^2 * w_{ik}^2)))$$

Où:

- P_{ij}: Le poids du terme i dans le document j.
- W_{ik}: Le poids du document i dans la requête k.

10. Evaluation du Modèle Vectoriel:

Pour pouvoir évaluer la formule choisie du modèle vectoriel, l'utilisateur est obligé de faire entrer une requête à partir de l'environnement de test.

Notre programme consulte le fichier qrels.text pour extraire les documents pertinents à cette requête et exploite aussi les documents considérés pertinents par le système vis à vis la requête puis évalue le modèle vectoriel en utilisant le rappel et la précision qui ce calculent comme suit:

- Rappel:

$$Rappel = \frac{\text{Nombre de documents pertinents trouvés par le système}}{\text{Nombre total de documents pertinents dans l'environnement}}$$

- Précision:

$$Précision = \frac{\text{Nombre de documents pertinents trouvés par le système}}{\text{Nombre total de documents trouvés par le système}}$$

II. Format des fichiers d'indexation:

Les structures de données que nous avons utilisé dans notre implémentation sont comme suit:

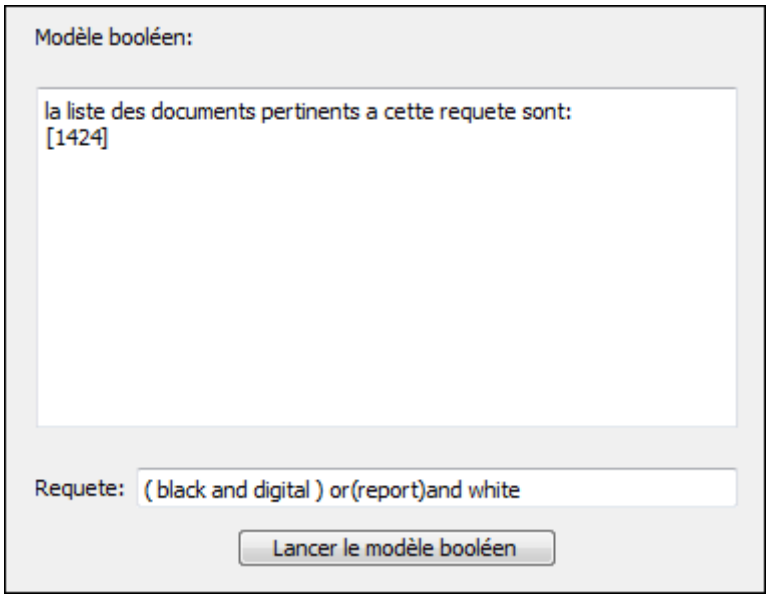
- *Documents*: Les documents, après prétraitement, sont sauvegardés dans une liste à accès direct.
- *Fichier inverse:* Le fichier inverse de la collection est représenté sous forme d'une matrice où les lignes sont les termes de tous le corpus et les colonnes sont les documents de tous le corpus, un élément M[i,j] de la matrice contient la fréquence d'apparition du terme numéro i dans le document numéro j.

- *Fichier Poids:* Le fichier des poids a le même principe du fichier inverse sauf qu'on parle des poids au lieu des fréquences, donc la représentation en terme de structure de données reste la même.
- *Requête* : une requête est représentée, soit pour le modèle booléen ou pour le modèle vectoriel, par une liste de mots (les mots de la requête), et cela pour permettre d'accéder aux mots de la requête un par un et vérifier leurs appartenance ou non à l'ensemble des documents.

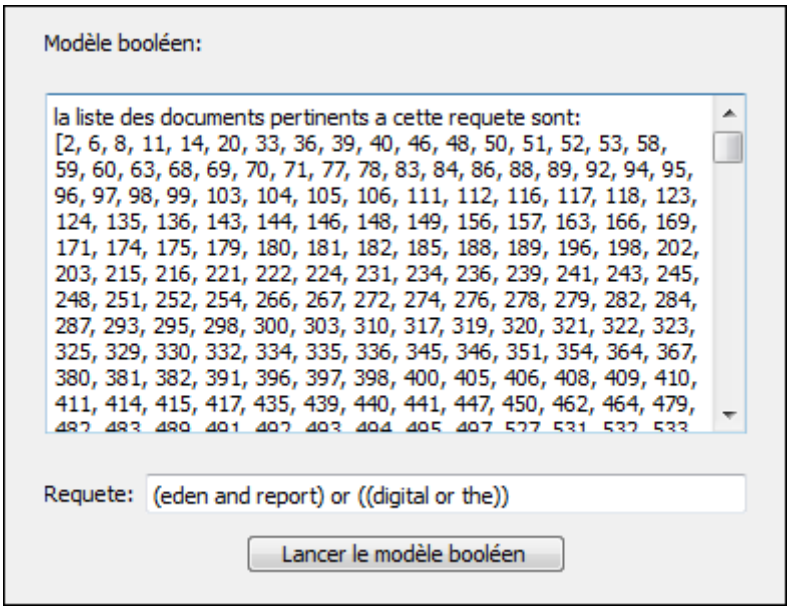
III. Résultats des requêtes

1. Modèle booléen :

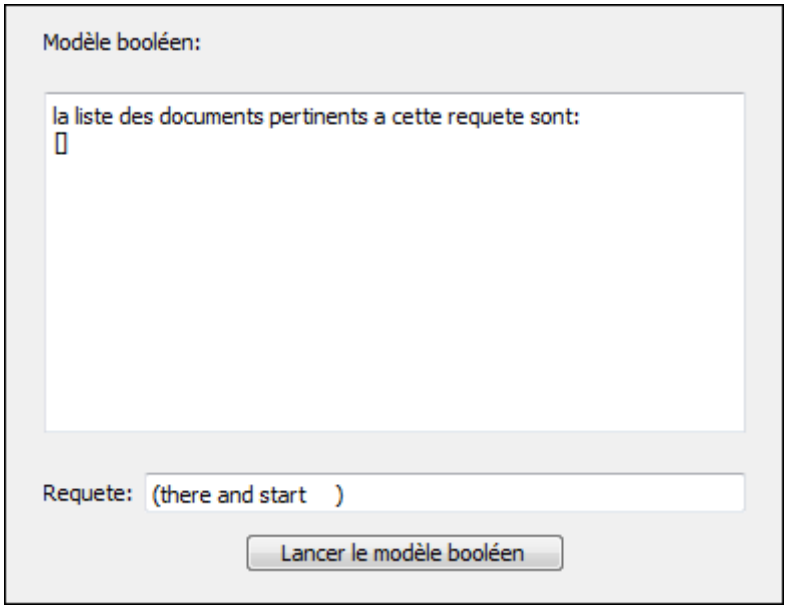
- (black and digital) or(report)and white



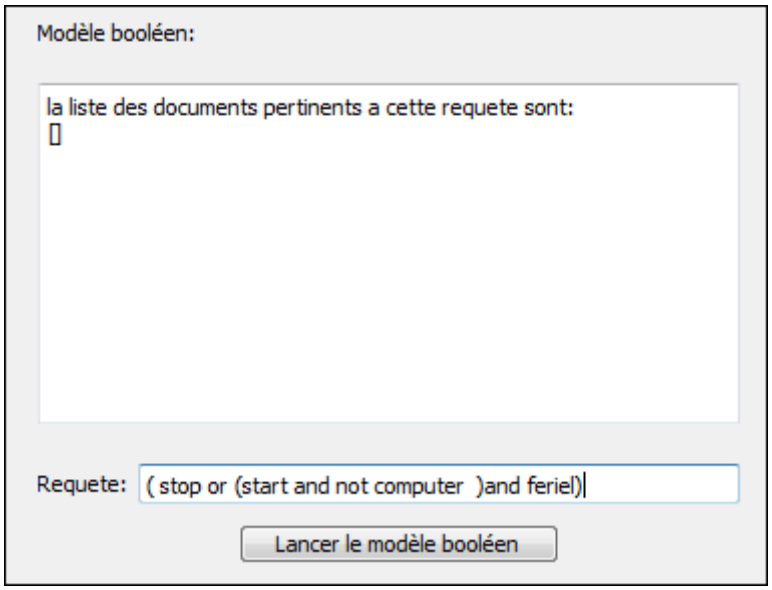
- (eden and report) or ((digital or the))



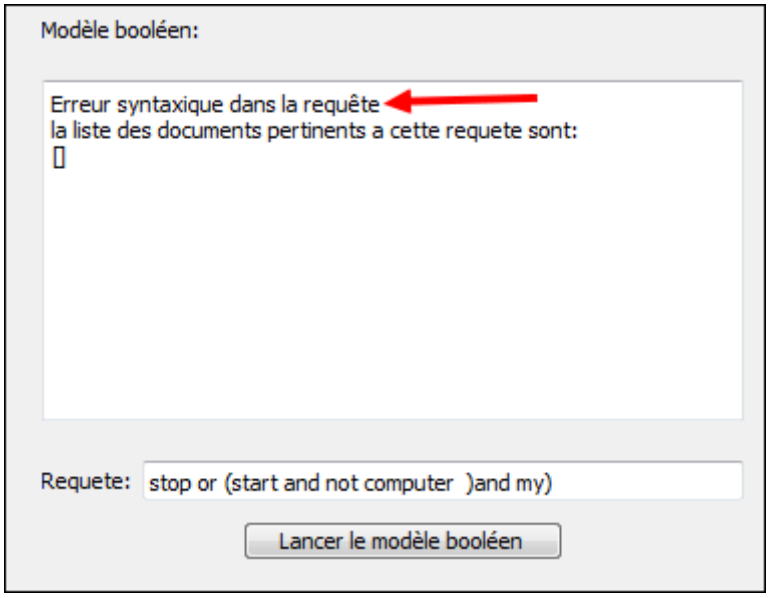
- (there and start) ;



- (stop or (start and not computer)and feriel)

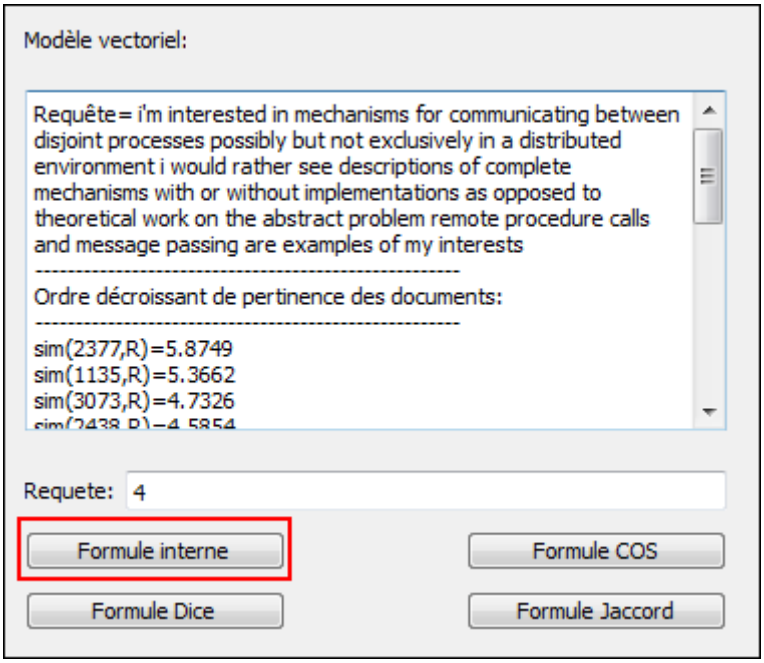


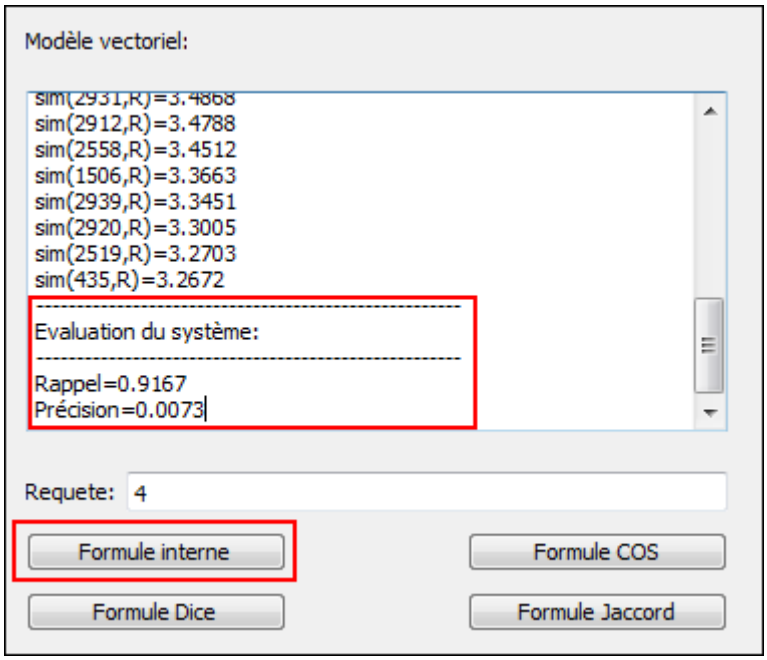
- stop or (start and not computer)and my) // Erreur Syntaxique : Parenthèse ouvrante manquante

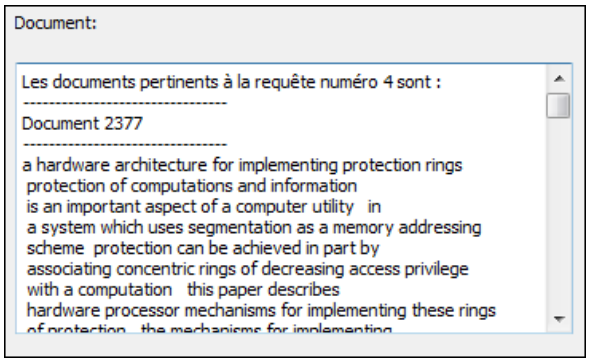


2. Modèle Vectoriel:

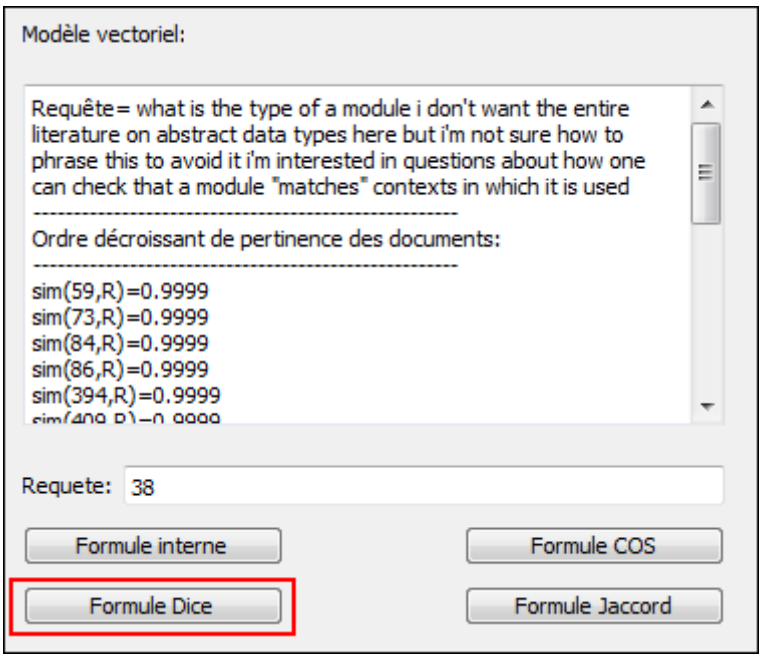
- **Formule du produit interne:**

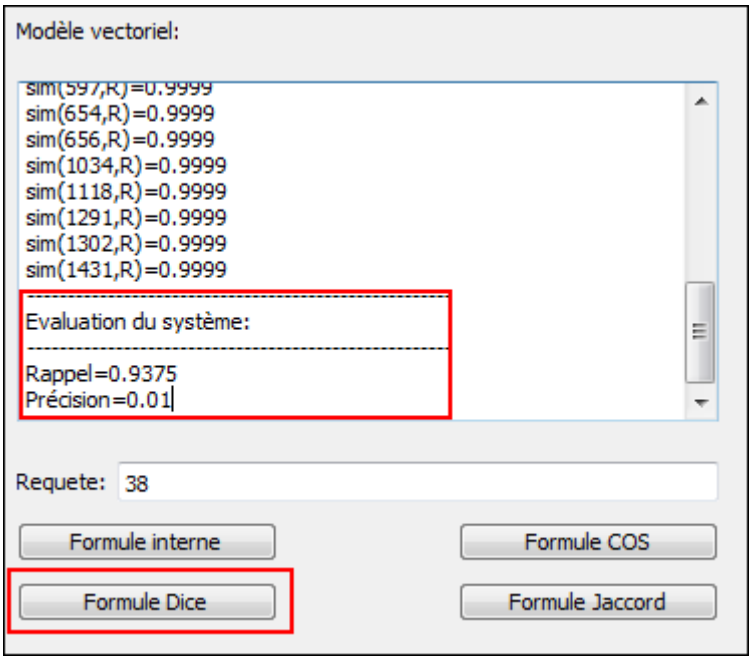


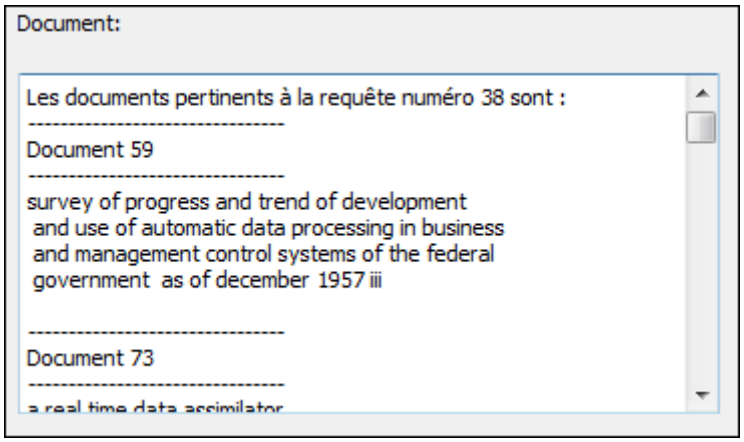




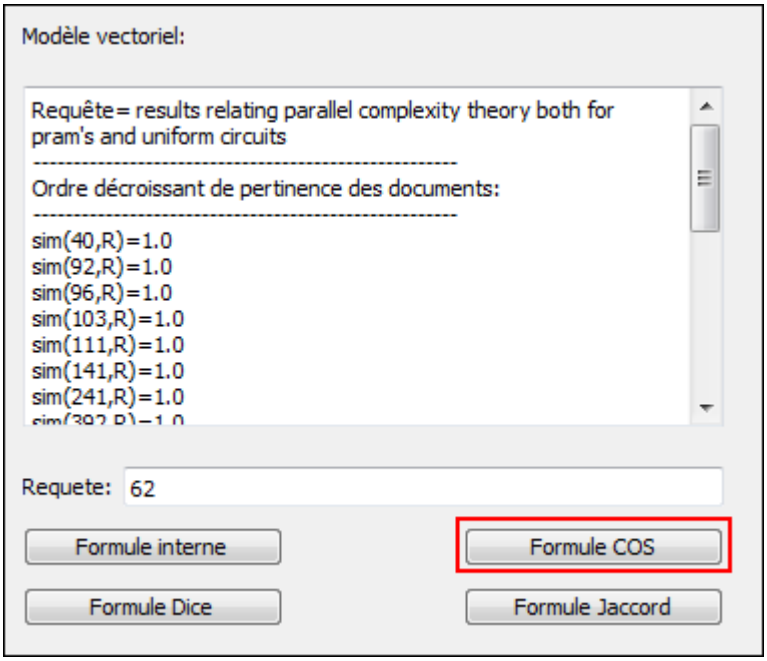
- **Formule du Coefficient de Dice:**

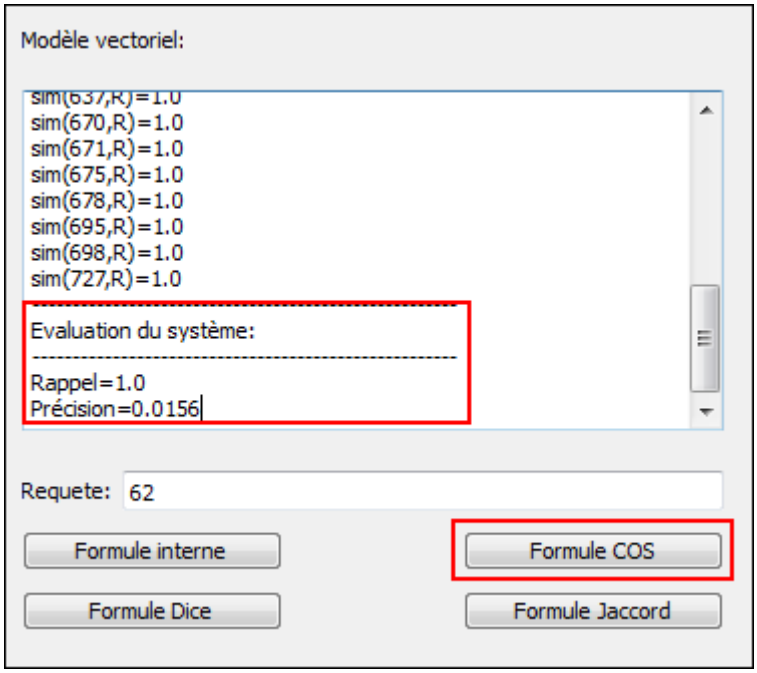


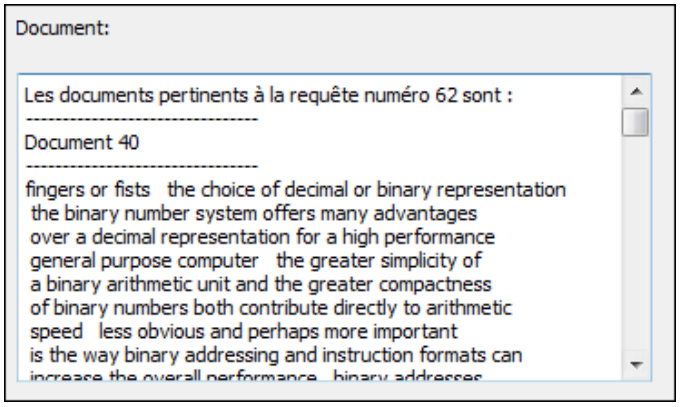

Modèle vectoriel:
sim(654,R)=0.9999
sim(656,R)=0.9999
sim(1034,R)=0.9999
sim(1118,R)=0.9999
sim(1291,R)=0.9999
sim(1302,R)=0.9999
sim(1431,R)=0.9999
Evaluation du système:
Rappel=0.9375
Précision=0.01
Requete: 38
Formule interne
Formule COS
Formule Dice
Formule Jaccord


Document:
Les documents pertinents à la requête numéro 38 sont :
Document 59
survey of progress and trend of development
and use of automatic data processing in business
and management control systems of the federal
government as of december 1957 iii
Document 73

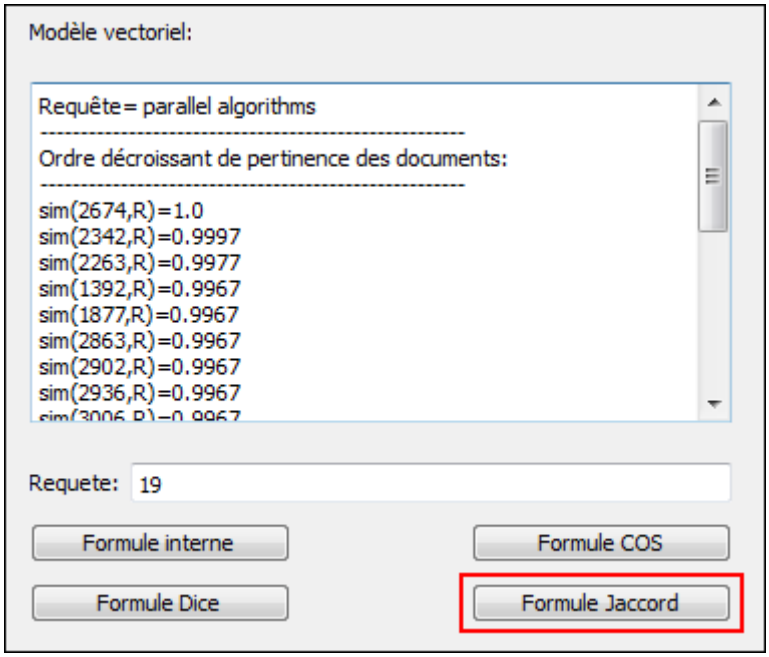
- **Formule du cosinus:**

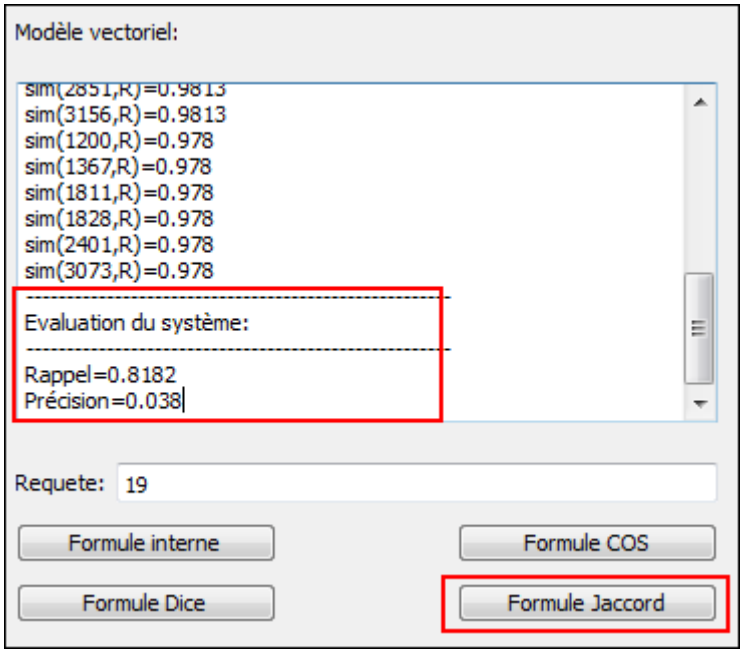


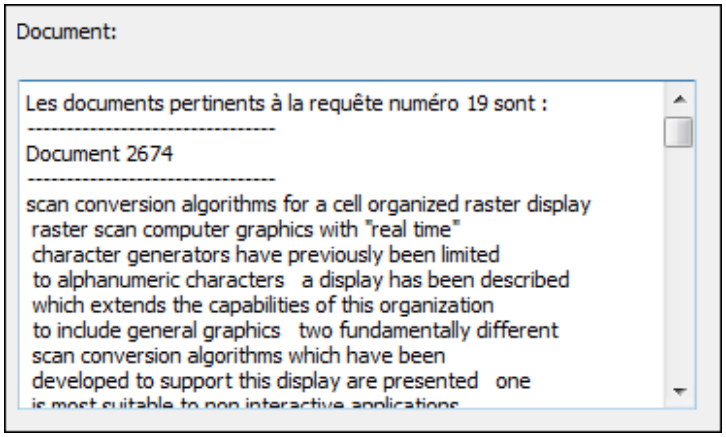




- **Formule de Jaccard:**







IV. Analyse et discussion de l'implémentation

L'utilisation des matrices comme des structures de données permet un accès directe aux éléments concernés par notre recherche, aussi, elle permet un simplicité d'implémentation et une meilleure lisibilité du code. L'espace mémoire occupé par les matrices est réduit en le comparant avec des structures plus complexes comme les listes et les dictionnaires.

V. Comparaison des résultats

1. *Modèle booléen VS. modèle vectoriel*

Modèle booléen:*

Le modèle booléen est un modèle qui est basée sur une logique booléenne très simple et facile à implémenter, aussi il permet à l'utilisateur de bien présenter ses besoins des informations. Cependant, il présente certains problèmes tels que:

- L'importance d'un terme dans un document n'est pas exploitée.
- La similarité d'un document avec une requête est soit 1 soit 0 ce qui fait qu'il est impossible de trier les documents par ordre décroissant de pertinence.
- Problème de conjonction: Il suffit d'avoir la similarité d'un terme qui est à 0 pour annuler toutes la conjonction d'un ensemble de termes dans la requête.
- Problème de disjonction: Il suffit d'avoir la similarité d'un terme qui est à 1 pour retourner une similarité de document égale à 1 même si tous les autres termes de la requête ont une similarité égale à 0.

Modèle Vectoriel:*

La requête dans le modèle vectoriel est représentée sous forme de vecteur. Ce modèle est venu pour régler certains problèmes du modèle booléen. Par exemple, le modèle vectoriel utilise les pondérations des termes ce qui permet d'améliorer les résultats des recherches. Aussi, la mesure de similarité permet de trier les documents par ordre décroissant de pertinence vis à vis la requête. Cependant, il présente aussi certains inconvénients tels que:

- La représentation vectorielle ne donne pas importance aux relations qui existent entre les termes.

2. *Les quatre formules du modèles vectoriel*

Quelque soit la requête, nous avons constaté qu'on a le même rappel et la même précision pour les quatre formules du modèle vectoriel malgré que les documents pertinents pour la requête changent d'un modèle à un autre.

* : [Informations basées sur des remarques tirées du support de cours de Mr. Kechid]

N.B.: Tous les prétraitements que nous avons fait sur les documents sont aussi faits sur les requêtes afin d'obtenir une cohérence lors du calcul de la similarité entre les documents et les requêtes.

N.B.: Tous les traitements de notre programme ont été fait sur les 3000 documents de la collection et non pas sur une partie seulement. Par contre les affichages ne portent que sur 20 documents et cela pour une meilleure lisibilité des résultats.

VI. Analyse et discussion des résultats

Notre application permet un temps de réponse relativement réduits car nous avons essayer d'établir un code avec une complexité polynomiale étudiée. De ce fait, le temps d'indexation des 3000 documents se fait en moins de 5 secondes.

Concernant le rappel et la précision de notre système, ils étaient réduits car l'environnement prend en compte la sémantique dans le calcul des similarité des requêtes avec les documents, et cela parce que nous avons trouvé qu'il existe des documents jugés pertinents par l'environnement pour une requête alors que celle ci ne contient aucun terme de ces documents. Et puisque les 4 formules du modèle vectoriel se basent sur la syntaxe seulement, les résultats étaient un peu moins concrets vis à vis l'environnement de test.

Conclusion

Le domaine de recherche d'information est devenu un domaine indispensable dans les axes de l'intelligence artificielle et cela parce qu'il se base sur des techniques de recherche dans un très grand nombre de fichiers et de sources d'information. Ce projet était un premier pas dans l'exploration des différentes techniques de recherche d'information et ouvre la fenêtre sur un monde vaste d'outils qui accélèrent l'accès aux informations pertinentes.

FIN.