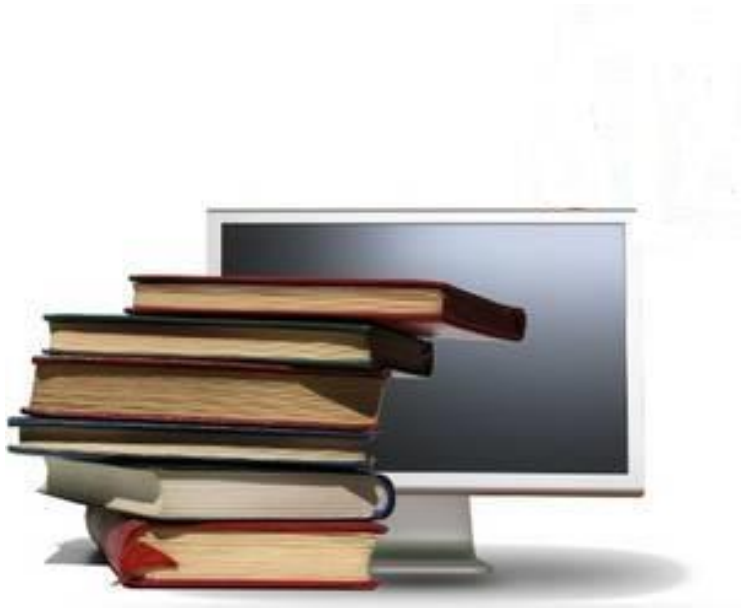
Enseignant : Bernard LEVRAT

Recherche documentaire

-

Mickaël FARDILHA – Raphaël PILLIE



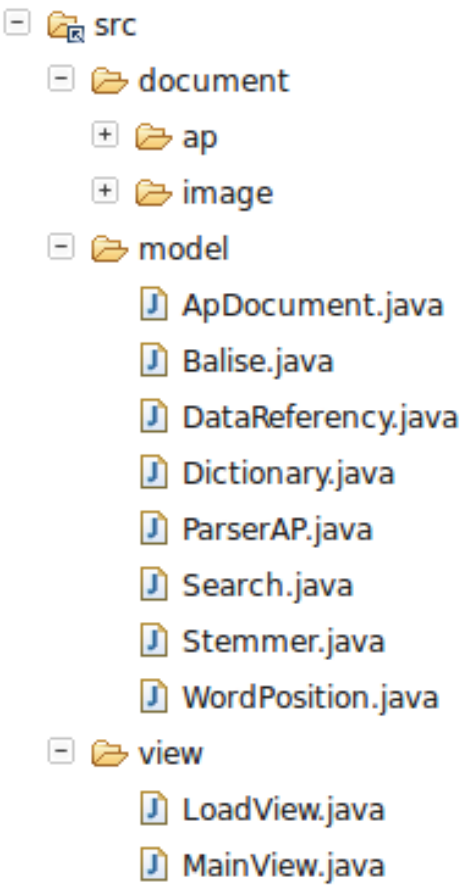
Sommaire

Introduction

Le master 1 informatique d'Angers propose deux options à choisir parmi quatre propositions. Pour cette seconde option, nous avons choisi la recherche documentaire.
Ainsi, suite au cours qui nous a été donné, nous avons mis en place une application permettant d'effectuer des recherches plus ou moins complexes sur un corpus documentaire relativement important.

I – Structure du projet

Afin de répondre le mieux possible au sujet, nous avons effectué une analyse afin de définir une structure propre pour ce projet. En effet, nous avons découpé l'ensemble du code pour séparer la partie algorithme et les conteneurs de la partie documents ou encore de la partie interface graphique de l'application.



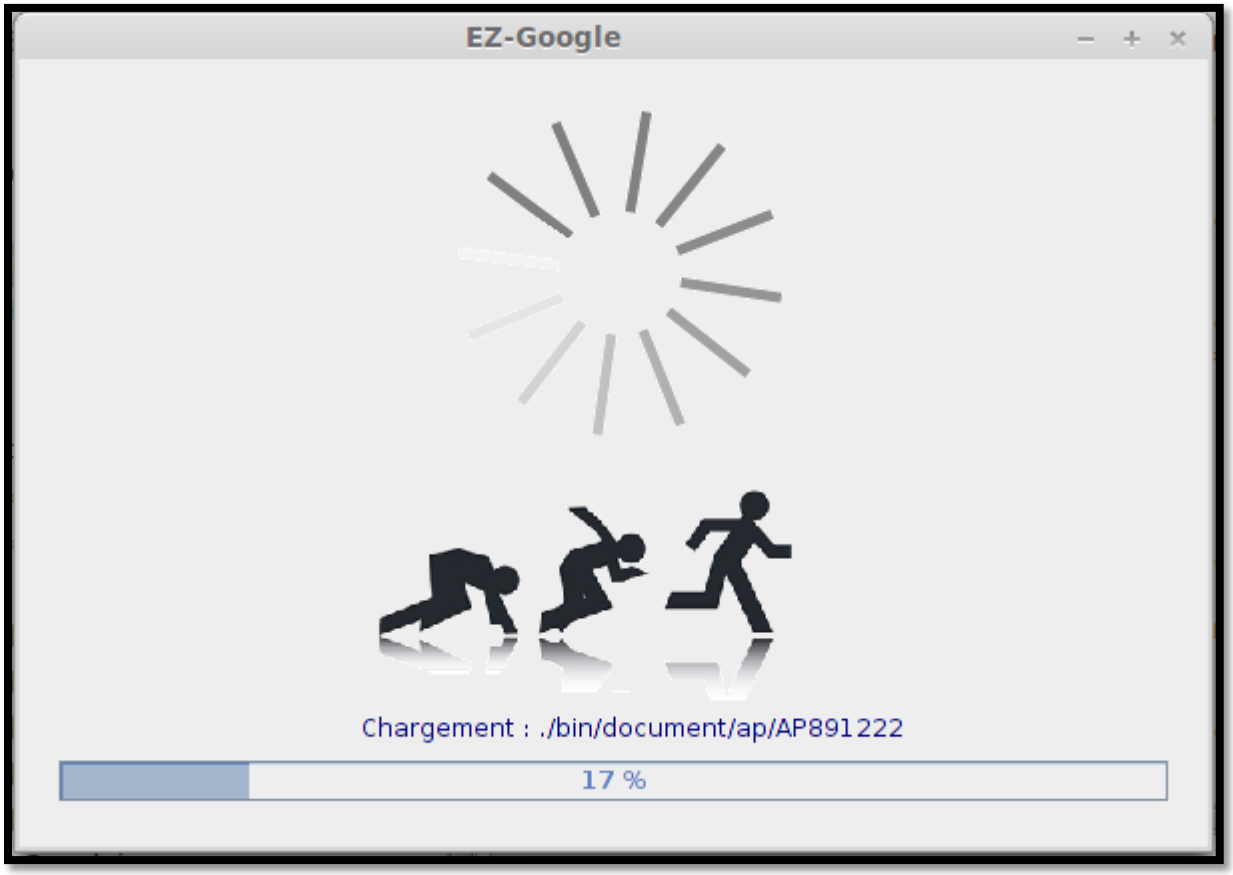
La classe « ApDocument » permet de représenter sous forme d'objet un document entier. Celui-ci est décomposé en plusieurs sous objets. L'objet « Balise » est une énumération permettant de référencer l'ensemble des balises présentes dans l'ensemble des documents,

en affectant un poids à ces dernières. La classe « DataReferency » permet quant à elle de représenter les données d'un mot en incluant la classe « WordPosition ». Celle-ci contient la balise qui contient le mot ainsi que sa position dans le texte.
Enfin, la classe « Dictionary » contient l'ensemble des mots du corpus documentaire ainsi que les informations relatives à ces derniers, grâce aux objets défini ci-dessus : chaque mot est décrit de façon à connaitre les noms des documents qui le contiennent, et ces noms de document font à leur tour référence à la représentation des données sur un mot (on utilise ici la classe « DataReférency »). Ansi, nous utilisons un conteneur java de type « HashMap<String,HashMap<String,DataReferency>> » qui nous permet de stocker efficacement en mémoire et de représenter le dictionnaire afin d'obtenir un accès aux données relativement efficace.

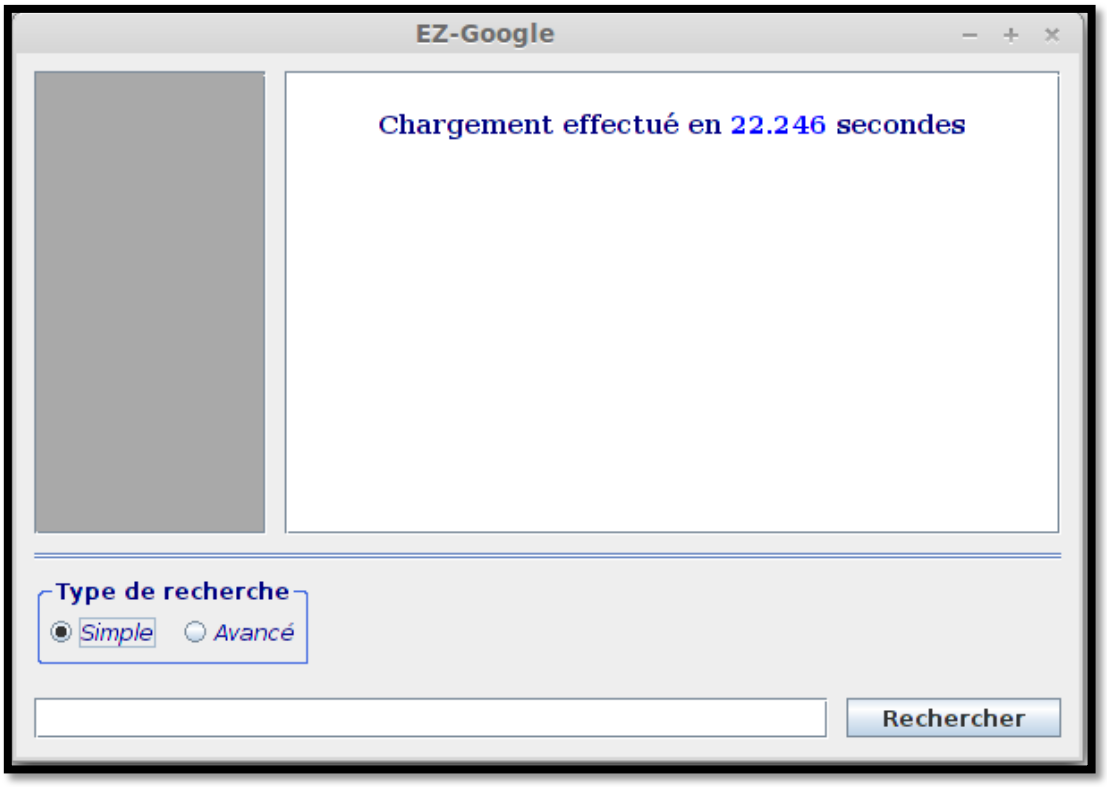
II – Présentation de l'application

L'application que nous avons mise en place possède une interface graphique, permettant de lancer la recherche souhaité sur le corpus documentaire que nous avons sélectionné.

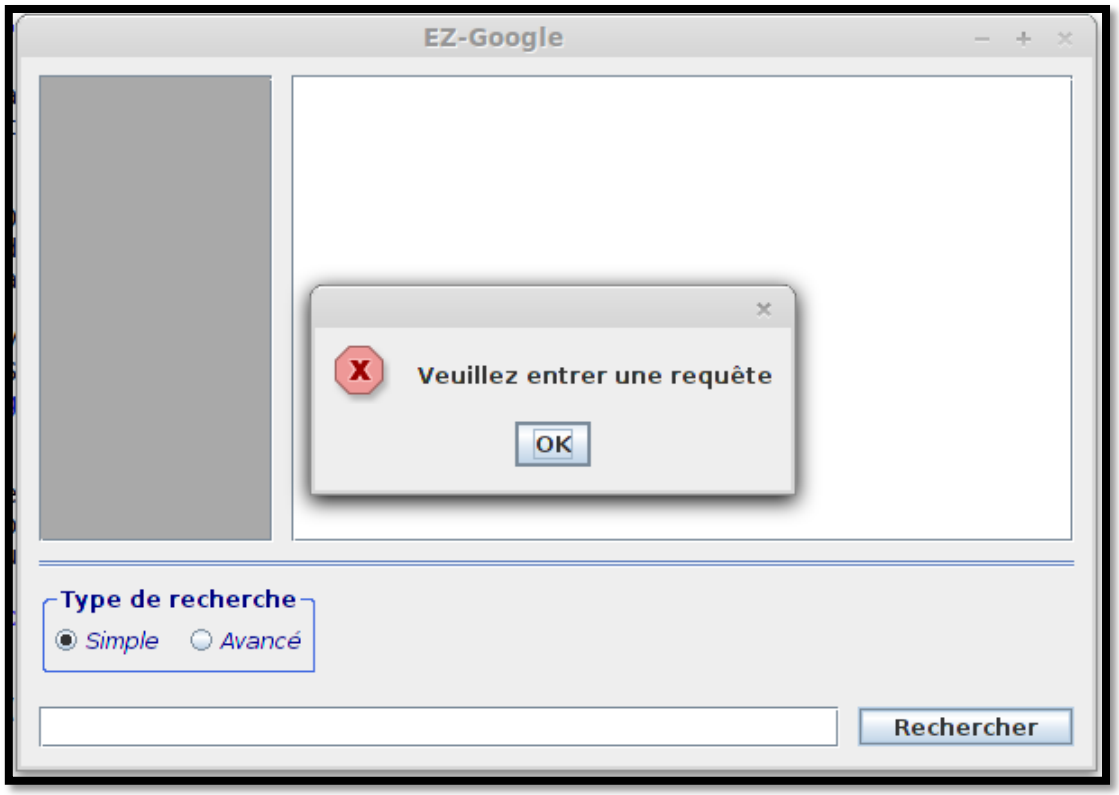
Au lancement de celle-ci, une première fenêtre apparait et permet à l'utilisateur de se rendre compte de l'état d'avancement du chargement des différents éléments :



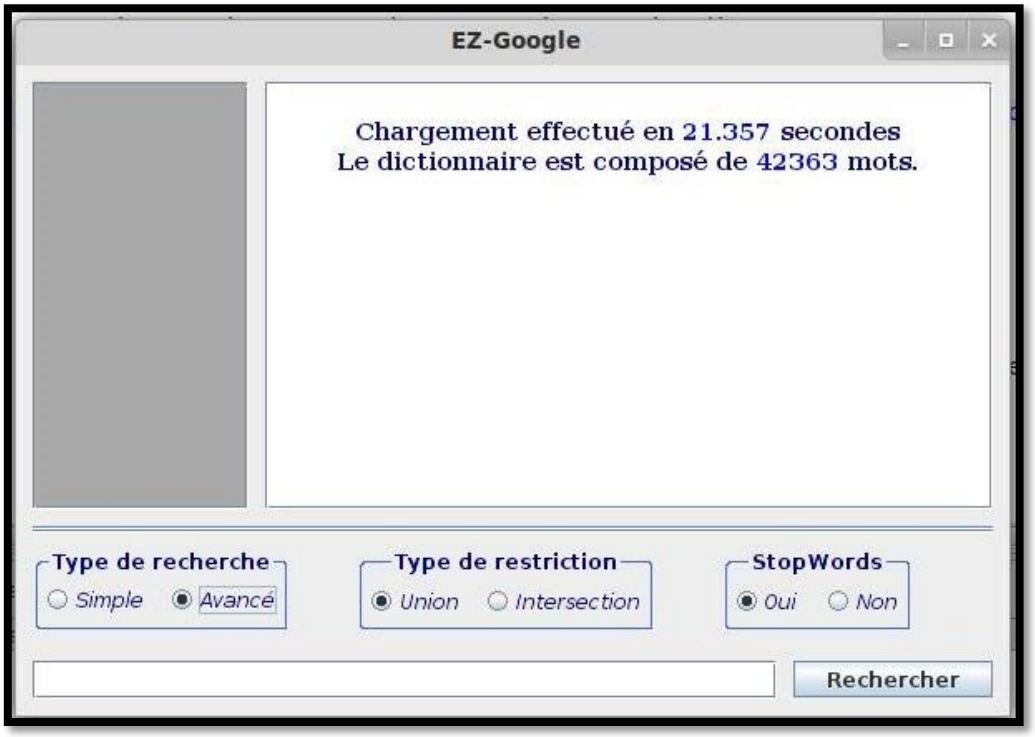
Une fois le dictionnaire chargé, la fenêtre permettant la recherche apparait à son tour :



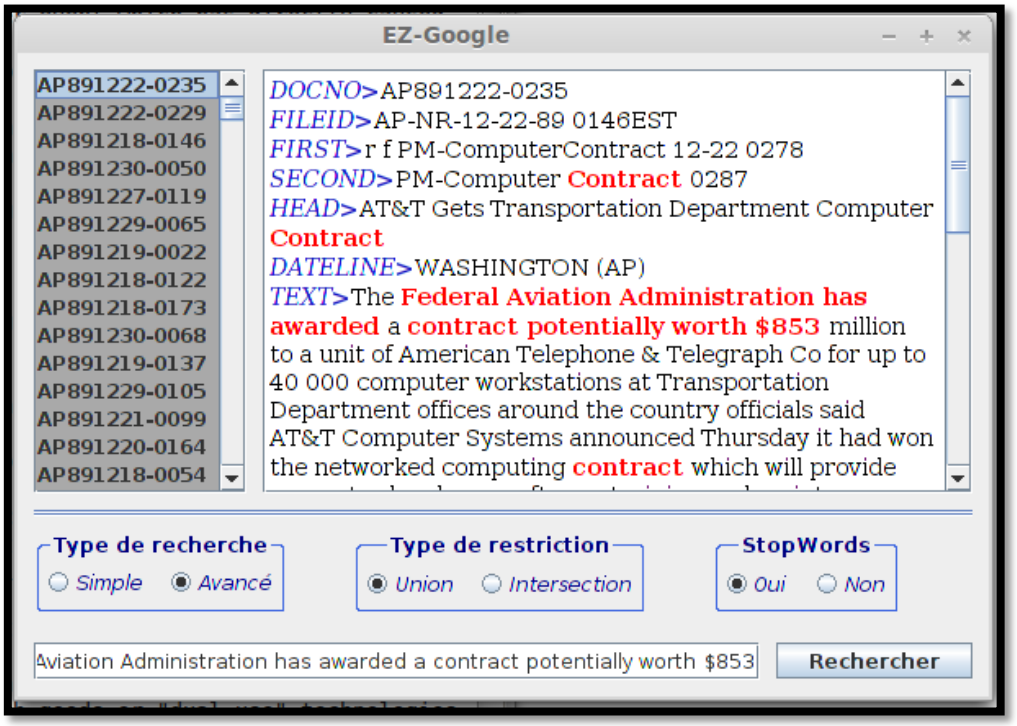
Le fait de lancer la recherche via le bouton sans renseigner de requête affichera le message suivant :



Avant de lancer la recherche il est possible de passer en mode avancé et de choisir si, lors de la recherche, nous souhaitons choisir un critère d'union et d'intersection, ou encore prendre en compte les stopwords :



Depuis cette fenêtre, il est donc possible de lancer une requête et d'en afficher le résultat sous la forme suivante :



Chaque documents dans le quel apparait la recherche est référencé dans la liste située sur la partie gauche de l'écran. En cliquant sur l'un des documents, il est possible de visualiser celui-ci et plus particulièrement les zones de textes concernant la recherche : ces zones apparaissent en gras et en rouge.

III – Algorithme

Dans un premier temps, nous créons le dictionnaire de la façon suivante : après avoir parsé tous les mots de toutes les balises, pour chaque mot, nous utilisons le « Stemmer de Porteur » puis nous stockons dans notre structure d'objet le mot avec ses références aux documents, sa position, etc…
Puis, nous parcourons le dictionnaire créé afin de calculer les poids de chacun des mots dans les documents :

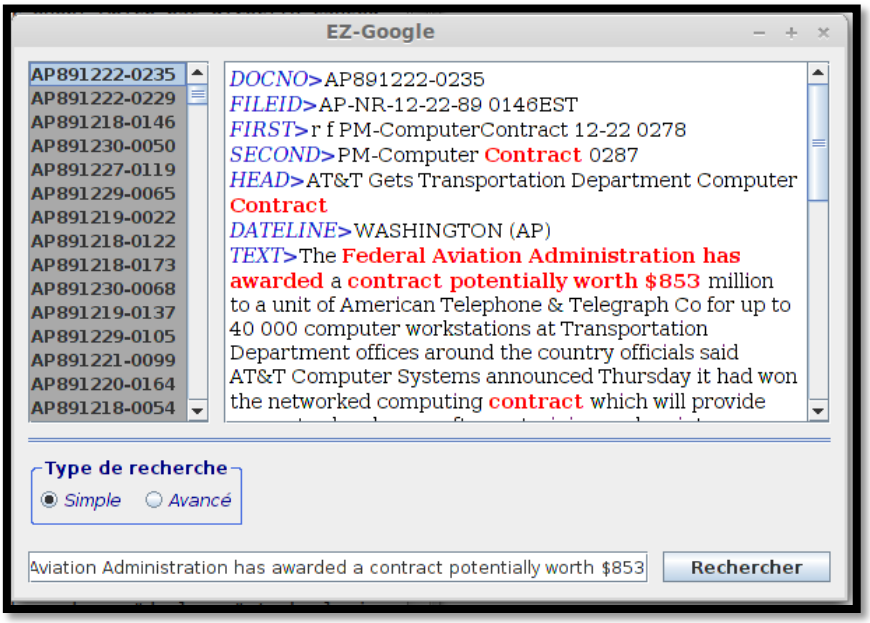
$$\frac{\text{nombre d'ocurence du mot pondéré par valeur de sa balise}}{\text{nombre d'occurrence du mot le plus présent}}$$

Une fois le chargement terminé, nous pouvons exécuter l'algorithme, qui se déroule sous la forme suivante : nous appliquons le « Stemmer de Porteur » sur la requête (nous prenons en compte les stopwords ou non selon les choix de l'utilisateur) puis nous recherchons dans le dictionnaire les références des mots de la requête aux documents.
Enfin, nous affections au document le poids du mot. S'il y a plus d'un mot dans la requête, nous utilisons une recherche positionnelle.
Cette recherche consiste à faire la somme des écarts des mots de la requête dans le document, et de rechercher l'écart minimum. Puis, nous rajoutons au poids du document déjà calculé, l'inverse de l'écart multiplié par le nombre de mot comprit de document au carré et divisé par le nombre de mots totales de la requête puis multiplié par un coefficient arbitraire (50) afin de rendre la recherche positionnel plus importante :

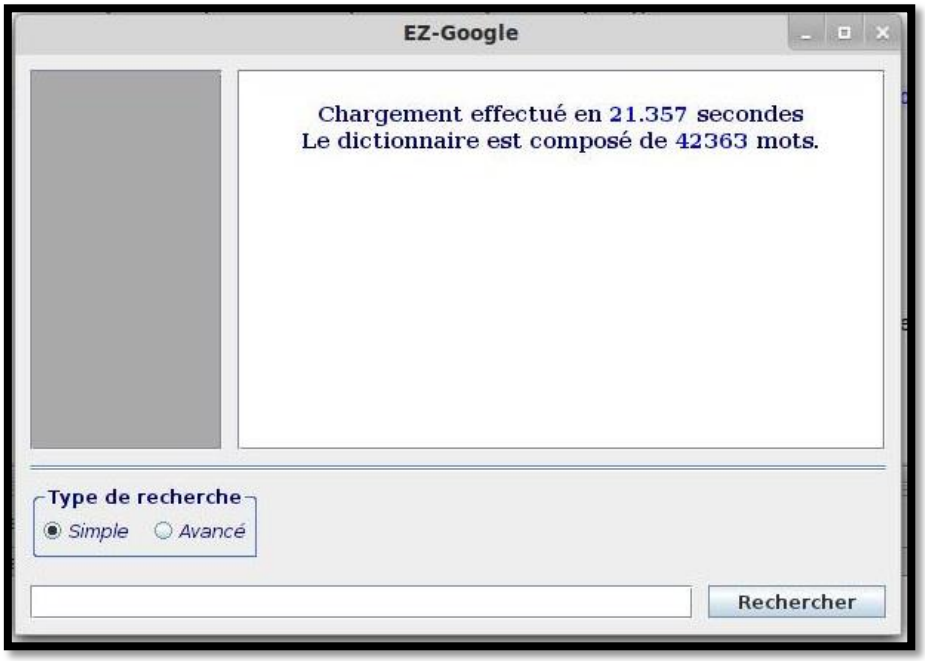
$$\frac{1}{ecart} \times \frac{nombre\ de\ mot\ dans\ le\ document^2}{nombre\ de\ mots\ dans\ la\ requête}$$

IV – Résultats & performances

L'ensemble des résultats obtenu est correcte avec les requêtes proposées. Ces derniers sont observables via l'interface graphique de façon claire : nous avons mis en place un affichage permettant de visualiser, par documents, les résultats obtenu en aillant une liste de documents trié par pertinence :



En termes de performance, nous pouvons observer un chargement de l'application d'environs vingt secondes. Cela est nécessaire pour la mise en place du dictionnaire et de l'ensemble des éléments nous permettant de lancer une recherche, qui est désormais instantanée :



V – Extensions

Outre la réalisation du projet, nous avons mis en place plusieurs extensions qui nous paraissaient indispensable pour un moteur de recherche. Les améliorations sont venues au fur et à mesure du développement pour palier à des besoins de rapidité, de lisibilité ou encore de cohérence.
La première extension que nous avons implémentée fut le classement des résultats. Celle-ci permet d'avoir en première position le document le plus cohérent vis à vis de la requête. Puis, nous avons jugé plus pertinent de mettre en évidence des mots qui ont été décisifs pour la recherche. En somme la mise en sur-gras et en couleurs des mots correspondant à la requête ou encore la lise en italique et en couleur des balises.
Afin d'avoir une méthode de recherche plus spécifique aux documents, nous avons implémenté une recherche positionnel qui permet de valoriser le rapprochement des mots. Ainsi, si l'on copie une phrase d'un document en recherche, on a beaucoup plus de chance de trouver ce document en tête de liste.
Pour finir, nous avons implémenté une type de rechercher « avancé », permettant de choisir de faire une union ou une intersection sur les mots de la requête, et de prendre en compte ou non les « Stopwords » lors de la recherche. La recherche par intersection permet d'obliger un document à avoir tous les mots de la requête (avec ou sans « Stopword »), alors qu'une recherche par union va considérer un document valide à partir du moment où il contient un mot de la requête.

Conclusion

Ce projet nous a permis de mettre en pratique les notions importantes vues en cours afin de nous sensibiliser à l'importance de la recherche documentaire, qui peut vraisemblablement devenir un enjeu majeur dans certains domaines de l'informatique.
Ainsi nous avons pu apercevoir la complexité de ce domaine, et commencer à apercevoir les techniques de recherche utilisés à ce jour.