* 1. Introduction

La probabilité de pertinence à priori de documents est parmi les méthodes utilisée pour améliorer les résultats de la recherche d’information. Notre travail consiste à implémenter et tester une nouvelle méthode d’estimation de la probabilité à priori des documents basée sur des caractéristiques du document. Ce score de pertinence à priori est ensuite combiné avec le score de comparaison document/requête.

Ce chapitre est organisé comme suit, nous allons tout d’abord définir notre approche de calcul de probabilité de pertinence à priori. Par la suite nous allons introduire l’environnement de développement de notre approche en précisant les outils et le langage utilisés pour sa mise en œuvre. Enfin nous présentons les résultats obtenus sur la collection de test TREC AP88.

* 1. Description de l’approche

Notre approche consiste tout d’abord à calculer la probabilité de pertinence à priori de document afin de la combiner avec le score initial (correspondance document/requête). Pour effectuer ce calcul nous nous sommes basés sur les caractéristiques du document qui seront présentées dans la partie suivante. Une fois ces caractéristiques identifiées nous effectuons un apprentissage automatique en utilisant la régression linéaire avec la fonction logistique (sigmoïdale) afin d’obtenir la probabilité de pertinence à priori de documents. Une fois cette probabilité est obtenue nous la combinons avec le score initial.

* + 1. Les caractéristiques utilisées pour le calcul de la probabilité de pertinence à priori de document :

Pour effectuer le calcul de pertinence à priori des documents, nous devons tout d’abord spécifier les caractéristiques à prendre en compte. Dans notre approche nous avons utilisé les caractéristiques suivantes :

* + - 1. La longueur de document :

La longueur du document est le nombre de termes que contient le document, Elle peut être un très bon exemple de jugement de pertinence à priori de documents. Elle peut être utile en cas de requête vague de l’utilisateur, le système retournera donc les documents classés par ordre de longueur au lieu d’un classement aléatoire.

Elle est calculée comme suit :

(D) = |D| (III.1)

* + - 1. Nombre de termes uniques :

L’intuition derrière cette caractéristique est qu’un document qui n’a pas de répétition de mots (le moins de bruit) est mieux classé qu’un document qui contient des mots qui se répètent plusieurs fois. Le nombre de termes uniques signifie le nombre de termes qui apparaissent dans le document en ignorant la redondance.

Elle est calculée comme suit :

(D)=|| (III.2)

Où est le nombre de termes dans le document D.

* + - 1. Moyenne IDF :

L’intuition derrière cette caractéristique est de classer les documents par rapport à l’importance des termes qu’il contient, un document qui contient des termes rares est mieux classé qu’un document contenant des termes courants.

La moyenne IDF est calculée comme suit :

= (III.3)

Tel que :

IDF(t)=) (III.4)

Où nbDoc est le nombre de documents dans la collection et nbDoc(t) est le nombre de documents contenant un terme donné.

* + - 1. Écart type IDF :

L’intuition derrière cette caractéristique est que plus est grand, la probabilité de pertinence à priori de document est minime.

Il est calculé de la manière suivante :

= (III.5)

Tel que :

Moyenne écart type IDF(t) = (III.6)

Où IDF(t) est la fréquence inversée d’un terme dans le document D.

* + - 1. Écart type TF :

L’intuition derrière cette caractéristique est que plus est grand, la probabilité de pertinence à priori de document est minime.

Il est calculé de la manière suivante :

= (III.7)

Tel que :

Moyenne écart type TF(t) = (III.8)

Où TF(t) est la fréquence d’un terme dans le document D.

* + - 1. Rapport TFmin/TFmax

Le rapport TFmin/TFmax permet de calculer l’homogénéité des fréquences des termes du document. Plus ce rapport est minime plus la probabilité de pertinence du document augmente.

Il est calculé comme suit :

= (III.9)

Où TFmin(D)et TFmax(D) sont respectivement la fréquence minimale et maximale d’un terme dans un document D.

* + - 1. Entropie

L’intuition derrière cette caractéristique est d’estimer la cohérence de documents, les documents avec une valeur de l’entropie assez importante ont tendance à être plus cohérents et donc une forte probabilité de pertinence à priori du document.

Elle est calculée de la manière suivante :

=- (III.10)

Où est la fréquence d’un terme dans le document D.

* + 1. L’apprentissage de la probabilité de pertinence à priori de documents :

Une fois les caractéristiques des documents sont obtenus, nous entamons la phase d’apprentissage nous permettra de prédire la probabilité de pertinence des documents en utilisant la régression linéaire.

L’apprentissage de la probabilité de pertinence à priori des documents consiste à calculer les caractéristiques des documents pour un échantillon donné de la collection , ce calcul nous permettra d’effectuer une régression linéaire en utilisant la fonction logistique présentée ci- dessous :

Y = (III.11)

Tel que :

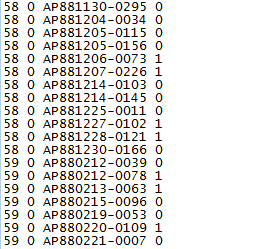
Y : La probabilité de pertinence à priori de document.

w : Les coefficients de corrélation entre les variables X et la pertinence Y.

X : Les variables explicatives.

Cette fonction vise à expliquer une variable d’intérêt binaire (c’est-à-dire de type « oui / non », « vrai / faux »ou « 0 / 1»). Les variables explicatives qui seront introduites dans le modèle peuvent être quantitatives ou qualitatives.

Nous avons utilisé les Qrels de requêtes numérotées de 51 à 100. Ci-dessous un extrait du fichier Qrels :



Pertinence

Identifiant du document

Numéro de la requête

Figure III-1 : Extrait du fichier des Qrels

Dans notre cas les variables explicatives sont pour chaque document : sa longueur, nombre de termes uniques, écart type TF, écart type IDF, moyenne IDF, rapport TFmin/TFmax et l’entropie nous aiderons à expliquer la pertinence des documents.

Une fois la fonction logistique est apprise sur le jeu de donnés, nous l’utilisons pour estimer la pertinence à priori d’un document « D »  noté comme suit : Prior (D).

* + 1. La combinaison des scores :

Afin de combiner les scores à priori des documents et les scores initiaux nous avons utilisé la somme logarithmique citée dans le chapitre précédent, elle est exprimée comme suit:

Nous avons varié la valeur de λ de 0.5 à 5 avec un pas de 0.5 et la valeur qui a donné le meilleur résultat est λ=3.

* + 1. Les outils et langages utilisés :

Pour la mise en œuvre de notre approche nous avons utilisé des outils et des langages de programmation que nous présentons ci-dessous :

* + - 1. Le langage de programmation Java :

La technologie **Java** définit à la fois un langage de programmation et une plateforme informatique. Crée par l’entreprise Sun Microsystems, et reprise depuis par la société Oracle, la technologie Java est indissociable du domaine de l’informatique et du Web.

La particularité et l’objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d’exploitations tels que Unix, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications.

Le langage Java reprend en grande partie la syntaxe du langage C++ , Néanmoins Java à été épuré des concepts les plus subtils du C++ et à la fois les plus déroutants , tels que les pointeurs et références , ou l’héritage multiple contourné par l’implémentation des interfaces.

Java a donné naissance à un système d’exploitation (JavaOS) , à des environnements de développement (eclipse/JDK) , des machines virtuelles (MSJVM (en),JRE) applicatives multiplateforme (JVM) , une déclinaison pour les périphériques mobiles/embarqués (J2ME),une bibliothèque de conception d’interfaces graphiques (AWT/Swing) , des applications lourdes (Jude,Oracle,SQL Worksheet ,etc.), des techniques web (servlets, applets) et une déclinaison pour l’entreprise (J2EE).

* + - 1. NetBeans :

NetBeans est un environnement de développement intégré, placé en open source par Sun en juin 2000 sous licence CDDI. (Common Devloppement and Distribution License) et GPLv2. En plus de Java, Netbeans permet la prise en charge native de divers langage tels le C, C++, JavaScript, le XML, le PHP et le HTML, où d’autres (dont Python et Ruby) par l’ajout de greffons. Il offre toutes les facilités d’un IDE moderne (éditeur avec coloration syntaxique, projets multi-langages, refactoring, éditeur graphique d’interfaces et de pages Web).

NetBeans constitue par ailleurs une plateforme qui permet le développement d’applications spécifiques (bibliothèque Swing(Java)). L’IDE NetBeans s’appuie sur cette plateforme.

Il supporte principalement les langages suivants :

Java, Javadoc, JavaScript12, C.

Les plates-formes supportées sont :

Microsoft Windows, Linux, Mac OS ; Solaris10.

Nous vous présentons ci-dessous l’interface qu’on a générée avec plate-forme NetBeans :

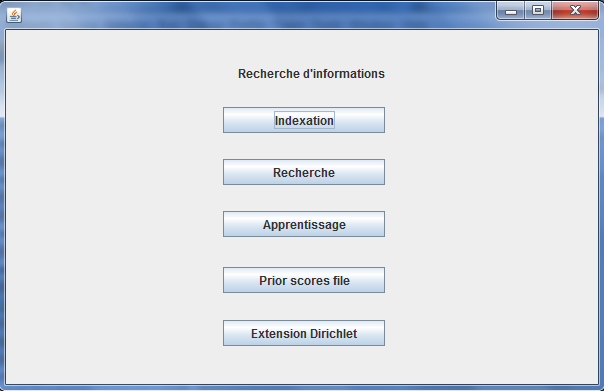


Figure III-2 : Interface Netbeans de notre approche

Tel que :

**Bouton 1** : Permet d’effectuer l’indexation de la collection AP88.

**Bouton 2** : Permet d’effectuer la recherche avec le modèle de langue Dirichlet.

**Bouton 3** : Permet le calcul des caractéristiques pour un échantillon de test Qrels AP88 afin d’effectuer l’apprentissage.

**Bouton 4** : Permet de calculer le score à priori des documents, et les stocker dans un fichier.

**Bouton 5** : Permet d’effectuer la recherche avec l’extension du modèle Dirichlet en se basant sur la formule(III.12).

* + - 1. Terrier :

Terrier est une plate-forme dédiée à la recherche d’information. Elle implémente les différents modules intervenant dans le processus de RI classique et offre en plus un cadre pour l’évaluation des résultats de recherche pour différentes applications. Terrier a été largement éprouvé**.** Le choix de cette plate-forme est dû aussi à sa capacité à traiter de grandes collections de documents telles que les collections TREC.

L’architecture de la plate-forme Terrier distingue les deux phases classiques : l’indexation et la recherche. Un corpus documentaire est fourni en entrée au module d’indexation. Les documents de la collection passent par un ensemble de prétraitements tels que la tokenisation. Les tokens sont ensuite injectés dans une chaîne de traitement TermPipLines, à savoir le StopWords PipeLine pour l’élimination des mots vides de sens, ou encore les Stemming pipeline et qui dépendent de la langue en question. La phase d’indexation conduit à la construction de l’index (Data structures).

La phase de recherche comprend le Manager, un module qui interagit avec l’application, réalise la mise en correspondance à travers les calculs des pondérations (selon le schéma de pondération (Weighting Model) choisi : PL2, BM25, Dirichlet LM etc.) ainsi que les scores des documents. Le résultat renvoyé à l’utilisateur, est la liste des documents jugés pertinents et leurs scores respectifs, dans notre cas on a choisi d’étendre la classe Dirichlet LM.

* + - 1. RStudio :

R est un langage de traitement et d’analyse de données de plus en plus répandu, notamment grâce à sa puissance et au fait qu’il est libre, gratuit et multiplateforme.

RStudio est un outil apparu récemment et qui vient combler un manque dans la collection des outils associés à R : il s’agit d’un environnement de développement intégré (IDE en anglais) fonctionnel, libre, gratuit et multiplateforme.

Un IDE n’est pas une interface graphique au sens de SPSS ou Modalisa, qui permettrait d’utiliser le logiciel à travers des menus et des boîtes de dialogue : il s’agit d’un environnement facilitant la saisie, l’exécution de code, la visualisation des résultats, etc.

RStudio est multiplateforme, vous pouvez donc le télécharger et le faire fonctionner aussi bien sous Windows, Mac OS X ou Linux :

Son interface se présente sous la forme d’une unique fenêtre découpée en quatre zones que l’on peut redimensionner, masquer ou maximiser selon ses préférences.

Dans notre cas nous avons utilisé ce logiciel afin d’apprendre la probabilité de scores à priori pour un échantillon de 8080 documents (extrait présenté dans la Figure III.1) et récupérer les coefficients de corrélation des variables explicatives citées dans (III.2.1).

* 1. Architecture de notre approche :

Nous présentons ci-dessous l’architecture de notre approche :

Collection de documents

Indexation

Calcul des caractéristiques de pertinence à priori de document

Jugements de pertinence

Score initial de la requête

Evaluation

Modèle de recherche

Apprentissage de score à priori de document

Score à priori de document

Figure III-3 : Architecture générale de notre approche

* + 1. Indexation des documents

La première étape à faire avant la recherche est l’indexation de la collection en utilisant la classe TRECIndexing de la plateforme Terrier.

* + 1. Calcul des caractéristiques utilisées pour le calcul de la probabilité de pertinence à priori des documents

Afin de calculer la probabilité à priori de pertinence de documents nous avons tout d’abord, pris un échantillon de Qrels (jugement de pertinence) afin de calculer les caractéristiques pour chaque document de cet échantillon de 8080 documents pour enfin les classer dans un fichier Excel qui sera chargé dans le logiciel RStudio afin d’effectuer une régression linéaire (figure ci-dessous ) :

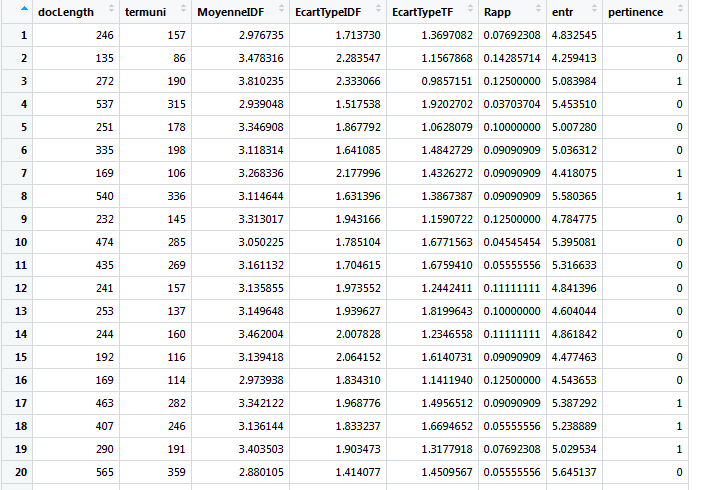


Figure III-4 : Fichier Excel contenant les valeurs des caractéristiques utilisées pour le calcul du score à priori de pertinence de documents

* + 1. Apprentissage et calcul de scores à priori de documents :

Une fois le fichier chargé nous allons effectuer une régression linéaire en utilisant la fonction logistique expliquée précédemment en utilisant les commandes de la figure suivante :

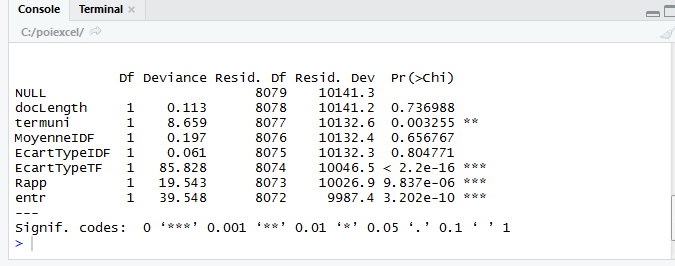
logistique.PNG

Figure III-5 : Commandes utilisées pour la mise en œuvre de la fonction logistique

Cette régression nous permettra de récupérer les coefficients de corrélation qui vont être utile pour le calcul du score à priori de document. Pour savoir quelles caractéristiques prendre en compte dans notre approche, on exécute la commande suivante :

anova (model, test="Chisq")

On aura donc le résultat présenté dans la Figure III.6, A partir de ce résultat il est désormais possible de distinguer les caractéristiques qui ont une influence considérable sur la probabilité de pertinence à priori de document, tel que plus la valeur de  **Pr** est basse, la corrélation entre la variable explicative et la pertinence est plus grande.



Caractéristiques significatives pour la pertinence

Figure III-6: Résultats de la corrélation entre les variables explicatives et la pertinence

Après avoir distingué les caractéristiques les plus ou moins influentes sur la pertinence on pourra donc choisir les coefficients à retenir pour le calcul du score à priori de document.

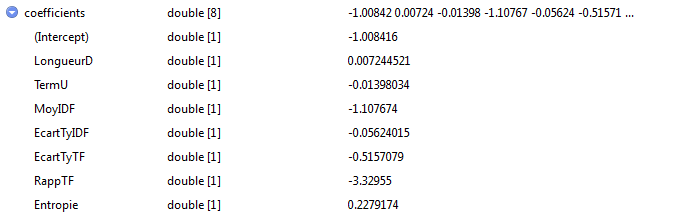


Figure III-7: Coefficients de corrélation entre les caractéristiques et la Pertinence

Après la récupération des coefficients de corrélation, nous calculons le score à priori de document en utilisant la formule suivante :

Prior Score (D) = coef1\*car1+coef2\*car2+…. +\*. (III.13)

A partir de la figure III.6 on remarque que le nombre de termes uniques, l’écart type TF, le rapport TFmin/TFmax et l’entropie sont des variables très significatives pour Y qui est la pertinence.

Le calcul se fera donc avec la formule suivante :

Prior score(D) = coef1× TermU+coef2×écart type TF + coef3×RappTF + coef4×Entropie (III.14)

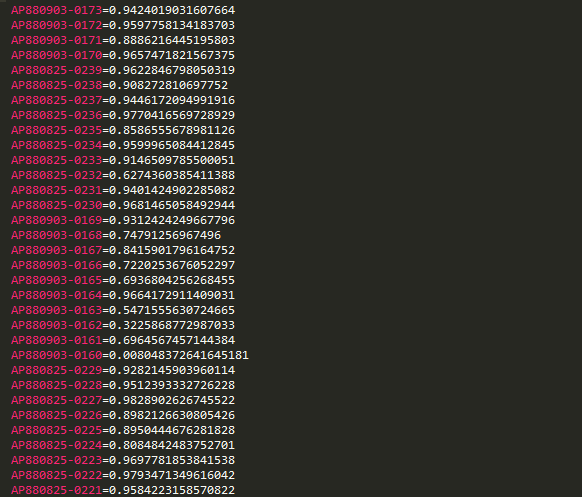
Ces scores seront chargés dans un fichier comme le montre la figure ci-dessous :

Figure III‑ : Le fichier contenant les scores à priori de documents

* + 1. Extension du modèle de recherche avec la score à priori de document :

Une fois la probabilité de pertinence à priori de document calculée on procédera à l’extension du modèle de recherche Dirichlet avec µ=2500 (par défaut) en utilisant la combinaison précédemment citée dans la formule (III.12).

* 1. Evaluation et résultats

Dans cette section nous présentons dans un premier temps la collection et les requêtes utilisées ainsi que les mesures d’évaluation adoptées. Ensuite nous présentons les résultats obtenus par notre approche et celle effectuée avec une recherche simple (Modèle de Dirichlet). Pour avoir une idée précise nous avons analysé aussi les résultats requête par requête.

* + 1. La collection de test et les requêtes utilisés :

Différentes collections de tests sont utilisées en recherche d’information. La collection que nous avons utilisée dans notre étude est : La collection **TREC AP88** (**A**ssociated **P**ress Newswire, 19**88**). Elle contient 79 919 documents.

Pour la recherche nous avons utilisé 50 requêtes issues de topics numérotées « 101-150» de la collection TREC.

Pour effectuer l’évaluation sous Windows il suffit d’accéder à l’emplacement de votre fichier Terrier, puis accéder au fichier bin et saisir les commandes suivantes :

* trec\_terrier –e pour une évaluation simple.
* trec\_terrier –e –p pour une évaluation requête par requête.

Afin d’évaluer les résultats, nous avons utilisé la mesure MAP, qui est la mesure la plus utilisée en recherche d’information et la précision à 1, 10, 15 et 20 documents.

* + 1. Résultats obtenus avant et après l’extension des modèles de recherche :

Dans cette section nous présentons les résultats globaux et détaillés de notre approche et la comparaison de résultats après l’évaluation.

Le tableau ci-dessous montre les résultats en termes de précision **(MAP)** obtenus avant et après la mise en œuvre de notre approche avec **le modèle de recherche Dirichlet**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Résultats de précision** | | | |
| **Précision** | **Dirichlet\_2500** | **Après l’extension** | **Taux d’amélioration** |
| **MAP** | 0.2437 | **0.2452** | **+0.61%** |
| **P@1** | 0.4694 | **0.4898** | **+4.34%** |
| **P@10** | **0.3347** | 0.3306 | -1.22 |
| **P@15** | 0.3061 | **0.3129** | **+2.22%** |
| **P@20** | 0.2959 | **0.2969** | **+0.33%** |

Tableau III ‑ : Précisions avant et après l'extension du modèle de recherche

A partir du tableau (III.1) nous constatons que notre approche améliore le modèle de base (Dirichlet) notamment avec la mesure P@1.

Pour avoir une idée plus précise de l’impact de notre extension nous avons analysé la précision requête par requête (voir figure III.8 et tableau III.2).

Figure III‑ Analyse requête par requête sur la collection AP88.

Dans le tableau ci-dessous nous avons sélectionné les 22 requêtes améliorées :

|  |  |  |  |
| --- | --- | --- | --- |
| **Requête** | **MAP avant l’extension** | **MAP après l’extension** | **Taux d’amélioration** |
| R101 | 0.1193 | 0.1328 | 11.31% |
| R102 | 0.3557 | 0.3740 | 5.14% |
| R104 | 0.2716 | 0.2755 | 1.43% |
| R106 | 0.3810 | 0.4059 | 6.53% |
| R107 | 0.4026 | 0.4155 | 3.20% |
| R115 | 0.2064 | 0.2288 | 10.85% |
| R116 | 0.0013 | 0.0015 | 1.15% |
| R117 | 0.2247 | 0.2329 | 3.46% |
| R122 | 0.3715 | 0.4421 | 19% |
| R125 | 0.2168 | 0.2349 | 8.34% |
| R128 | 0.0988 | 0.1039 | 5.16% |
| R131 | 0.0422 | 0.0504 | 19.43% |
| R132 | 0.7943 | 0.8217 | 3.44% |
| R134 | 0.6917 | 0.7011 | 1.35% |
| R135 | 0.4034 | 0.4368 | 8.27% |
| R137 | 0.1571 | 0.1731 | 10.18% |
| R139 | 0.0437 | 0.0494 | 13.04% |
| R140 | 0.1194 | 0.2606 | 118.25% |
| R141 | 0.0934 | 0.1024 | 9.64% |
| R144 | 0.0404 | 0.0534 | 32.18% |
| R145 | 0.1429 | 0.1433 | 0.27% |
| R148 | 0.0373 | 0.0396 | 6.17% |

Tableau III‑ : Les requetes améliorées par notre approche avec le modèle de recherche Dirichlet

* 1. Conclusion :

Dans ce chapitre nous avons présenté en premier lieu notre approche de calcul de la pertinence à priori de documents. Cette pertinence est estimée via une méthode d’apprentissage basée sur un ensemble de caractéristiques : longueur de document, nombre de termes uniques dans le document, moyenne IDF, écart type TF, écart type IDF, Rapport TFmin/TFmax et l’entropie.

Nous avons ensuite combiné cette pertinence avec la pertinence classique (Document/Requête).

Les évaluations effectuées sur une collection de test AP88 ont montré des améliorations notamment avec les mesures de haute précision.