# TP moteur de recherche

Philéas SAMIR, Gwladys SANCHEZ, Pooran SHAHDI

## 1 Première phase : crawling du site [Pooran SHAHDI]

### Question 1

Le nombre de pages dans la base est : 6424.

### Question 2

Le nombre de pages crawlées pour lesquelles l'URL de requête est le même que l'URL de réponse est : 3197.

### Question 3

De manière générale, certaines pages sont non indexées pour différentes raisons :
- contenu soumis à une authentification (pages privées, services "confidentiels" comme les comptes bancaires, payants comme l'accès à du contenu soumis à abonnement, etc.)
- pages "isolées", vers lesquelles aucune autre page ne renvoie
- pages dont le contenu et/ou formattage ne respecte pas les conditions d'indexation par le moteur de recherche considéré.

Dans notre cas, la page en question n'est pas indexée parce qu'elle correspond au deuxième cas de figure : elle n'est citée par aucune autre.

## 2 Deuxième phase : précalcul de TF et IDF [Philéas SAMIR]

### Question 4

La page qui contient le plus d'occurrences du mot matrice est la page Matrice (mathématiques).

### Question 5

L'IDF de "matrice" est 1.376.

## 3 Calcul du PageRank [Gwladys SANCHEZ]

### Question 6

Les 20 pages qui ont le meilleur PageRank sont les suivantes :

- Lien 1 : Fonction de Legendre
- Lien 2 : .css
- Lien 3 : .css
- Lien 4 : index
- Lien 5 : .css
- Lien 6 : .css
- Lien 7 : .css
- Lien 8 : .css
- Lien 9 : .css
- Lien 10 : .css
- Lien 11 : .css
- Lien 12 : .css
- Lien 13 : .css
- Lien 14 : Mathématiques

— Lien 15 : Mathématicien
— Lien 16 : Nombre complexe
— Lien 17 : Nombre réel
— Lien 18 : Géométrie
— Lien 19 : Polynôme de legendre
— Lien 20 : American Mathematical Society

Les pages qui ont le meilleur score au PageRank sont donc :
— la page obtenue par le lien "aléatoire". En effet ce lien est présent depuis toute les pages, par contre il n'est exploré qu'une fois lors du crawling, donc bien qu'il renvoie chaque fois à une page aléatoire, le système considère que toutes les pages pointent vers ce lien. Ici, il s'agit de la fonction de Legendre.
— les liens utilisés pour la mise en forme des pages (.css). Ceci s'explique par le fait que toutes les pages dépendent du même domaine, ont des "formes" similaires, et utilisent donc un nombre limité de fichiers de formattage, vers lesquels elles pointent nécessairement.
— les pages suivantes dans le classement sont des pages très générales, et donc souvent citées par les autres.

# 4 Requêtage [Philéas SAMIR]

## 4.1 Question 7

Les 10 meilleures pages pour la requête "comment multiplier des matrices" selon TF-IDF sont :

— Lien 1 : Matrice (mathématiques)
— Lien 2 : Matrice de rotation
— Lien 3 : Théorème des facteurs invariants
— Lien 4 : Matrice diagonalisable
— Lien 5 : Décomposition en valeurs singulières
— Lien 6 : Matrice par blocs
— Lien 7 : Logarithme d'une matrice
— Lien 8 : Exponentielle d'une matrice
— Lien 9 : Matrice inversible
— Lien 10 : Théorème de Cayley-Hamilton

## 4.2 Question 8

Les 10 meilleures pages pour la requête "comment multiplier des matrices" selon PageRank × TF-IDF sont :

— Lien 1 : Matrice (mathématiques)
— Lien 2 : Théorie des graphes
— Lien 3 : Mathématiques
— Lien 4 : Nombre complexe
— Lien 5 : Déterminant (mathématiques)
— Lien 6 : Groupe (mathématiques)
— Lien 7 : Albert Einstein
— Lien 8 : Groupe de Lie
— Lien 9 : Matrice inversible
— Lien 10 : Matrice de rotation

# 5 Extensions possibles [Philéas SAMIR, Gwladys SANCHEZ, Pooran SHAHDI]

## 5.1 Stemming plus intelligent et filtrage des *stop-words*

**Stemming NLTK** Dans le but d'améliorer le stemming réalisé au cours de l'indexation ainsi que de la requête, nous avons utilisé le modèle de stemming fourni par NLTK, avec le stemmer "FrenchStemmer".

Ce stemming était légèrement plus long, le script index.py s'exécutait en environ un quart d'heure, contre cinq minutes avec le stemmer simplifié, mais a un impact réel sur

les tables créées. Avec le stemmer initial, la table "IDF" comportait 81777 lignes, contre 67750 avec NLTK (−17%). De même, la table "Inverted_Index" est passée de 1894763 à 1833158 lignes (−3%).

**Filtrage des *stop-words*** Dans un second temps, nous avons essayé d'ajouter une étape supplémentaire au pré-traitement, en retirant les stop-words, dans le but de limiter l'importance de ces mots dans les requêtes, tout en réduisant la taille des index stockés sur la base de données. Nous constatons une réduction très importante de la taille des index : la table "IDF" est réduite à 67684 lugnes en retirant les *stop-words* (−17% par rapport à l'état initial), tandis que la table "Inverted_Index" est passée à 1616851 lignes (−15%).

Ces mots de liaison sont également retirés des requêtes, afin de ne conserver que les mots les plus riches de sens.

## 5.2 Amélioration du ranking

Nous avons utilisé les requêtes suivantes pour l'évaluation :

1. "multiplication matrices"
2. "comment multiplier des matrices"
3. "machine turing"
4. "comment s'appelle cette machine d'un mathématicien anglais"
5. "machine qui fait des calculs et éventuellement est capable de se faire passer pour un humain"
6. "hypothèse de riemann"
7. "la somme des entiers positifs fait elle moins un douzième"

Le but était d'évaluer les métriques de ranking au regard de leur capacité à répondre adéquatement à des requêtes variées qui illustrent différentes stratégies d'interaction avec un moteur de recherche. **On choisira les améliorations dans un but généraliste, i.e. les méthodes qui améliorent les résultats pour tous les types d'utilisateurs.**

**Avant stemming NLTK et filtrage des *stop-words***

- Racine carrée de PageRank : utiliser la racine carrée du score de PageRank améliore significativement les résultats pour quasiment toutes les requêtes. En effet, on remarquait dès la section précédente que le score PageRank avait tendance à pondérer trop fortement les noeuds importants du graphe (eg., des pages comme "Albert Einstein" s'introduisent dans les résultats d'une requête quant à la multiplication de matrices).
- Racine carrée ou carré de TF : ces deux méthodes (ainsi que le fait de ne pas modifier TF) améliorent ou dégradent certaines requêtes. Par exemple, le fait d'utiliser la racine carrée de TF fait apparaître "Produit matriciel" dans les résultats de la requête 2. En revanche, ce résultat n'apparaît toujours pas pour la requête 1. Cette métrique fait également disparaître "machine de Turing" de la requête 5. A l'inverse, l'utilisation du carré permet de placer des pages pertinentes plus haut dans le classement pour la majorité des requêtes, mais dégrade les performances pour les requêtes suivantes.
- Multiplication par le compte de mots dans la page ou par son carré : afin de ne pas alourdir le temps de recherche, nous n'avons calculé cette pondération que pour les 20 premiers résultats de Pagerank × TF-IDF. Ces deux méthodes améliorent les résultats pour certaines requêtes courtes (pour la requête 2, le produit matriciel apparaît dans les résultats) ; cependant, pour les requêtes plus longues, les résultats sont fortement dégradés, en particulier du fait de la présence de mots très courants dans l'ensemble des pages ("mathématicien", "calculs", "entiers").

— Bonus si les mots recherchés sont présents dans le titre : nous avons expérimenté avec une pondération additive ($x+ = 0.1$) et une pondération multiplicative ($x* = 1.1$). Cette méthode ne semble pas avoir un effet significatif sur les résultats.

**Après stemming NLTK et filtrage des *stop-words*** Les résultats sont visiblement améliorés pour l'ensemble des requêtes.

— Racine carrée du PageRank : cette méthode est à nouveau strictement meilleure que la méthode sans racine carrée.
— Racine carrée ou carré de TF : la racine carrée dégrade significativement les résultats, tandis que le carré les modifie (certains résultats pertinents progressent dans le classement, d'autres régressent voire disparaissent).
— Multiplication par le compte de mots dans la page ou par son carré : cette fois-ci, multiplier par le compte ou par son carré ne semble quasiment pas modifier les résultats. Le filtrage des *stop-words* fait que les mots sont déjà quasiment tous présents dans les pages, la pondération n'a donc presque aucun effet.
— Bonus si les mots recherchés sont présents dans le titre : à nouveau, cette méthode ne semble pas modifier significativement les résultats.

**Calcul et prise en compte de la localité** Nous avons calculé une métrique de localité et modifié les scores tel que :

$$\text{score}_{\text{local}} = \text{score} \times (1 + \frac{1}{\text{window}})$$

avec window la fenêtre la plus courte qui contient l'ensemble des keywords. Nous aurions pu reclasser les 20 premiers résultats directement à l'aide de ce critère (avec window $= +\infty$ pour les pages qui ne contiennent pas tous les mots). Cependant, ce classement n'aurait pas bien réagi aux requêtes sous la forme de questions/phrases. D'autres métriques de localité auraient été pertinentes, mais le choix de cette méthode a été guidé par le fait que la complexité en temps du calcul dépend de la longueur de la page et non de la longueur de la requête (contrairement à la moyenne des distances minimales entre toutes les paires de mots, par exemple). Dans le cas où tous les mots de la requête (filtrée) ne sont pas contenus dans la page, le score n'est pas modifié. Pour les requêtes précédentes, cette pondération n'a pas d'effet significatif.

En conclusion, nous notons que ces remarques ne sont valables que pour les requêtes proposées ci-dessus. En modifiant l'ensemble de requêtes à tester, on modifiera les améliorations choisies. Idéalement, des utilisateurs pourraient utiliser notre moteur via une interface web, et nous pourrions collecter leurs données d'utilisation afin d'améliorer notre outil...

### 5.3 Interface web

**Présentation de l'interface** A l'aide de flask, nous avons également implémenté une interface web très simple, permettant dans une page d'accueil (index.html) de saisir sa recherche, en précisant l'algorithme à utiliser (TF-IDF uniquement ou avec PageRank, qui est le choix par défaut). Lorsque la requête est envoyée, les résultats sont affichés dans une page différente, qui contient également un champ de recherche, un lien de retour à l'accueil, et un rappel de la requête tapée. Lorsque la recherche ne renvoie pas de réponse, l'utilisateur reste sur (ou est renvoyé vers) la page d'accueil et une notification apparaît afin de le lui indiquer.

**Modifications apportées pour l'obtenir** Afin d'adapter les scripts précédemment réalisés à l'interface web, quelques modifications ont été apportées :

— création d'un script "web_server.py", qui se charge d'exécuter "index.py" et "pagerank.py" si on ne dispose pas déjà d'une base de données complète (ce qui est donc le cas par défaut, avec uniquement le résultat du crawling). Ensuite le script met en place le serveur, et appelle le script "query.py".
— le script "query.py" a été modifié par simple ajout d'une fonction, similaire au "main" utilisé quand on appelle juste le script dans le terminal, mais qui retourne d'autres données en sortie (pas uniquement la liste des URL renvoyées par la requête, mais aussi le titre associé à la page pointée, et son résumé), afin de nourrir directement l'interface web.

**Utilisation du code global** Si on veut utiliser l'interface, il suffit donc :
— d'ajouter la database issue du crawling dans le dossier,
— de lancer le script "web_server.py" depuis son terminal,
— de laisser la création des tables s'exécuter (ce qui peut prendre 5 à 20 minutes selon le stemmer utilisé),
— puis d'ouvrir l'URL vers le serveur.