**Enseignement d’intégration *HeadMind Partners***

Rapport du groupe 4

Ali Baklouti, Anas Himmi, Aubin Ollivier, Alexandre Pedone

Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement

Tables des matières

1. Réponses aux questions des notebooks Day 1, Day 2 et Day 3…………………….3
2. Notre moteur de recherche (Cobra Search)…………………………………………...5

Bibliographie………………………………………………………………………………9

1. Réponses aux questions des notebooks Day 1, Day 2 et Day 3 :

**Jour 1 – Day 1**

|  |  |  |
| --- | --- | --- |
| **Question** | **Requête** | **Résultat** |
| Combien y a-t-il de lignes dans les tables Posts, Tags et Comments ? | SELECT COUNT(\*) FROM PostsSELECT COUNT(\*) FROM Tags  SELECT COUNT(\*) FROM Comments | Posts: 76613  Tags: 693  Comments: 79140 |
| Combien de commentaires ont été écrits depuis le début de l’année ? | SELECT COUNT(\*) FROM Comments WHERE CreationDate LIKE '%2023%' | 2165 |
| Combien y a-t-il d’utilisateurs ? | SELECT COUNT(\*) FROM Users | 127712 |
| Combien de nouveaux utilisateurs y a-t-il chaque année depuis 2020 ? | SELECT COUNT(\*) FROM Users WHERE CreationDate LIKE '%2020%'    SELECT COUNT(\*) FROM Users WHERE CreationDate LIKE '%2021%'    SELECT COUNT(\*) FROM Users WHERE CreationDate LIKE '%2022%'    SELECT COUNT(\*) FROM Users WHERE CreationDate LIKE '%2023%' | 2020 : 19856      2021 : 16623      2022 : 13823      2023 : 4656 |
| Quel est le contenu du plus petit post ? | SELECT TOP 1 Body FROM Posts ORDER BY LEN(Body) ASC | "" |
| Quel est le post avec le plus de votes ? | SELECT TOP 1 Id, Score FROM Posts ORDER BY Score DESC | 42051, 454 |
| Quels sont les dix tags les plus fréquents en 2022, ordonnées du plus au moins fréquent ? | SELECT TOP 10 Tags FROM Posts WHERE YEAR(CreationDate)=2022 GROUP BY Tags ORDER BY COUNT(\*) DESC | machine-learning  python  nlp  deep-learning  python  pandas  time-series  logistic-regression  neural-network  dataset |

**Extraction de données : Les données sont au format XML. Selon vous, pourquoi les développeurs préfèrent ce format ? Quels auraient pu être d'autres formats possibles ?**

Les développeurs préfèrent le format XML car il est flexible pour le stockage et la transmission de données entre applications, plates-formes et organisations. De plus, la transmission est légère sur Internet. De surcroît, les fichiers XML peuvent décrire des données dans une structure qui peut être facilement comprise à la fois par les machines et les humains. HTML, JSON ou YAML sont d'autres formats qui auraient pu être utilisés pour remplacer le format XML.

**Jour 2 – Day 2**

Une méthode naïve consisterait à compter le nombre de mots communs entre la requête et chaque publication. Ensuite, pour classer les publications, on pourrait directement sélectionner la publication qui maximise le nombre de mots communs.

**Avantages et inconvénients de cette méthode naïve :**

|  |  |
| --- | --- |
| **Avantages** | **Inconvénients** |
| * Facile à implémenter * Exécution rapide pour chaque requête : 1 minute 30 par requête | * Accorde la même importance et le même poids à tous les mots même les mots communs comme ‘’the’’ ou ‘’is’’ * Ne prend pas en compte l’ordre des mots dans la requête * Ne prend pas en compte l’ordre des mots dans les documents * Ne prend pas en compte le nombre de fois où un mot apparaît dans un document * Ne prend pas en compte le nombre de fois où un mot apparaît dans le corpus |

Suggestions d’amélioration :

* Accorder moins d’importance aux mots communs (‘’the’’, ‘’is’’…)
* Mettre des poids aux mots (TF-IDF par exemple) qui prennent en compte leur nombre d’apparition dans chaque document et dans tout le corpus.
* Prendre en compte l’ordre des mots dans les requêtes (par exemple en implémentant un algorithme de machine learning qui est entraîné sur des phrases complètes)

**Jour 3 – Day 3**

**Réponses dans le notebook**

1. Notre Moteur de recherche (Cobra Search)

Notre but serait d’utiliser un modèle probabiliste parce qu’il est bien adapté aux indexes inversés.

Le modèle probabiliste le plus utilisé est le modèle BM25. Toutefois, ce modèle suppose que tous les documents ont la même pertinence à priori. Dans notre cas, nous aimerions prendre en compte les métadonnées non textuels des documents.

Une recherche bibliographique approfondie nous a conduit à utiliser le modèle suivant basé sur le document suivant : <https://dl.acm.org/doi/10.1561/1500000019> [1](sections 3.6 et 3.7)

Ce modèle est inspiré du modèle BM25 et tient compte du texte d’un document, de son titre, de ses tags et des métadonnés non textuels tels que le nombre de votes, les commentaires etc.

**Démarche :**

|  |  |
| --- | --- |
| Etape | Fonction(s) associée(s) |
| Tokenization des textes des documents, de leurs titres et de leurs tags | text\_process( ) |
| Réalisation des index inversés pour les textes, titres et tags des documents | get\_inverted\_index( )  get\_inverted\_index\_tags() |
| Racinisation des termes des 3 index | stem\_inverted\_index() |
| Obtention des métadonnées non textuelles :   * Réputation de l’utilisateur * Nombre de votes sur un post * Nombres de réponses à un post * Badges d’un utilisteur * Nombre de vues * Si on a répondu au post ou non | get\_reputation( )  get\_votes( )  get\_number\_answers( )  get\_badges\_user( )  get\_views( )  get\_answered( ) (renvoie 1 s’il y’a une réponse et 0 sinon) |
| Calculer le score de chaque métadonnée non textuelle (1) | precompute\_features\_score() |
| Calculer le score des métadonnées textuelles (2) | CobraSearch( ) |
| Rechercher les documents pertinents pour une requête |

1. Pour calculer le score de chaque métadonnée nous avons attribué deux poids à chacune de ces données et on a calculé son score comme suit :

* Pour la réputation, le nombre de réponses, les badges, le nombre de vues :
* Pour “si on a répondu au poste ou pas ‘’ :
* Pour le nombre de votes (nous avons remarqué qu’il pouvait être négatif) :

1. Pour les métadonnées textuelles, nous avons attribué 2 poids un correspondant à la longueur du texte de la donnée et un correspondant à la fréquence du terme dans un document .

* Ensuite, on a calculé une valeur qu’on a appelé pour chaque post:
* On a aussi défini m la moyenne des Ld pour tous les posts
* On détermine également la fréquence d’un terme dans le texte d’un post, son titre et ses tags et on obtient la variable
* Enfin on peut calculer le score du titre, texte et des tags d’une requête comme suit :
* Ce qui nous permet enfin de calculer le score RSV adaptée d’un post par rapport à une requête
* Et c’est à partir de ce score que nous classons les documents du plus pertinent au moins pertine,nt en prenant l’ordre décroissant des scores RSV

**Remarques :**

* Pour chaque requête, les posts sont d’abord filtrés pour garder les posts qui contiennent le plus grand score RSV dans la requête afin d’accélérer les calculs.
* Toutes les données textuelles sont traités de la manière suivante : une tokenization avec bert-base-uncased (car les posts sont en anglais), suppression des stopwords puis racinisation.

**Visualisation de certains résultats :**

Voici quelques résultats de requêtes classés du plus pertinent au moins pertinent( les résultats sont des listes contenants les id des posts les plus pertinents) **:**

object detection machine learning

['41187', '51384', '24930', '39841', '30749', '15750', '48650', '104634', '80451', '48645']

how to do linear regression in python

['8266', '50829', '64750', '42577', '40394', '8625', '8457', '72855', '38237', '18258']

similarity scores by clustering

['87651', '78448', '75945', '36660', '75988', '18178', '64458', '80926', '61336', '93126']

map reduce algorithm database

['6910', '14942', '13053', '137', '2632', '60820', '114367', '80591', '22228', '13687']

Après consultation des posts qui nous ont été renvoyés, nous jugeons que les posts sont en effet pertinents et donc que le moteur de recherche fontionne correctement.

**Interface graphique :**

Nous avons également développé une interface grâce à la bibliothèque tkinter de Python. Voici un apreçu de cette interface. Il suffit d’entrer la requête dans la zone de texte, appuyer sur le bouton Search et notre algorithme de recherche s’exécute et renvoie les liens des posts pertinents dans le cadre juste en dessous du bouton search. En cliquant sur un de ces liens, nous sommes renvoyés vers le post en question sur un navigateur

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 2. Saisie d’une requête et résultats

Figure 1. Interface graphique ‘’vide’’

Bibliographie

[1] :ROBERTSON Stephen and ZARAGOZA Hugo,The Probabilistic Relevance Framework BM25 and Beyond,2009: <https://dl.acm.org/doi/10.1561/1500000019>