Projet MongoDB

Il s'agit pour vous (2 personnes max.) de montrer votre expertise en MongoDB en constituant un dossier de réponse à l'appel d'offre relatif au cas d'étude décrit ci-dessous.

1 Cas d'étude – Modélisation

Le cas d'étude proposé concerne le site participatif d'avis Yelp décrit dans le document adéquat disponible sur l'espace Moodle du cours. Globalement, des utilisateurs donnent leur avis sur des restaurants. Ces utilisateurs possèdent ou non des amis qui sont aussi des utilisateurs du site. La première étape cruciale est de définir une modélisation adaptée aux besoins applicatifs définis dans les sections suivnates 1.3 et 2.

1.1 Importation des données

Nous allons d'abord commencer par charger les données dans la base. Pour ce faire, appliquez le protocole suivant :

1. Télécharger l'archive contenant les fichiers yelp_restaurants.json, yelp_user.json et yelp_review.json correspondant aux données Yelp pour l'état du Delaware.
2. Écrire le programme permettant d'extraire et de transformer les données Yelp suivant la modélisation adaptée aux besoins applicatifs définis dans les sections suivnates 1.3 et 2. Les données produites doivent également être au format Json.
3. Dans un premier terminal, démarrer le serveur MongoDB qui stockera nouvelle la base de données.
4. Dans un second terminal, exécuter l'outil (programme) d'importation MongoDB pour chaque collection modélisée. La commande ressemblera à :

   ```
   ./mongoimport --uri "mongodb://localhost:27017/nom_de_base"
   --collection "nom_de_collection" "chemin_absolu_vers_fichier_json" --drop -jsonArray
   ```

5. Si aucun message d'erreur n'apparaît, vous avez réussi !
6. Dans un troisième terminal, il est possible de démarrer le client MongoDB pour vérifier que les données ont été correctement importées.
7. Dans un terminal, exécuter votre programme Python pour la recommandation de restaurants.

1.2 Découverte des données

À l'aide des commandes de base vous pouvez vérifier que les données importées sont correctes par rapport à la modélisation définie pour répondre aux besoins applicatifs de recommandation de restaurants.

1.3 Interrogation des données

Maintenant que les données relatives à l'état du Delaware sont insérées, vous devez définir les requêtes MongoDB permettant de répondre aux besoins suivants :

1. obtenir la liste des restaurants de la ville de Greenville
2. obtenir les utilisateurs qui ont rédigé des reviews avec une note inférieure ou égale à 2.0
3. obtenir les noms et adresses des restaurants de la ville de New Castle qui entrent dans la catégorie Mexican
4. obtenir le nombre d'utilisateurs qui n'ont pas d'ami
5. obtenir le nombre de restaurants pour chaque ville
6. obtenir la note globale le plus élevée obtenue par un restaurant
7. obtenir les restaurants qui proposent au moins 3 ambiances différentes
8. obtenir les identifiants des restaurants qui ont été évalués par l'un des utilisateurs correspondants aux identifiants Yelp n2UpKhkU2N-66a1QQzrjYw et ucD25otZ0uqWPSJnl4muQQ
9. obtenir les identifiants Yelp et noms des utilisateurs qui ont évalué les deux restaurants d'identifiants Yelp UBX73ZWgCdgom4nv0UeGvg et 51Fo_uY52A5oaI8YTHU5yg
10. obtenir le nombre de restaurants qui ont reçu des notes au-dessus de 4.0 et également en-dessous de 2.0
11. obtenir les différentes catégories associées aux restaurants évalués par l'utilisateur d'identifiant Yelp FlXBpK_YZxLo27jcMdII1w
12. obtenir le nombre d'amis de chaque utilisateur (y compris pour ceux qui n'en ont pas)
13. obtenir les identifiants des amis communs aux utilisateurs correspondant aux identifiants Yelp FlXBpK_YZxLo27jcMdII1w et 6Mv-qMJyxSokCu8YFM1o0A
14. obtenir les commentaires d'utilisateurs qui mentionnent le mot amazing
15. obtenir les 3 prénoms d'utilisateurs les plus présents dans la base

2 Application : moteur de recommandation de restaurants

La dernière partie de ce TP a pour objectif la construction d'un moteur de recommandation de restaurants. Le principe est le suivant : soit u un utilisateur à qui l'on souhaite recommander des restaurants. On souhaite que le système retourne une liste ordonnée $\langle (r_1, s_1), \ldots, (r_k, s_k) \rangle$ de restaurants, r_i (avec $1 \leq i \leq k$), associés à leur score de pertinence, s_i (avec $1 \leq i \leq k$). Les restaurants recommandés seront obligatoirement choisis parmi l'ensemble Λ des restaurants de la ville associée à l'utilisateur.

Le score de pertinence d'un restaurant est fonction de différents facteurs, énoncés et formalisés ci-dessous :

1. facteur « appréciation », lié à la note globale (rating) attribuée au restaurant pondérée par le nombre de reviews émises sur le restaurant. Formellement, le facteur social, noté $f_a(u, r)$, est calculé ainsi : $f_a(u, r) = \frac{Rating(r) \times |Reviews(r)|}{5.0 \times \max_{r' \in \Lambda} |Reviews(r')|}$, ainsi $f_a(u, r) \in [0.0, 1.0]$

2. facteur « préférence », lié à l'adéquation entre le type de restaurant suivant différents critères (Category+Ambience) et les critères préférés de l'utilisateur, c'est-à-dire les critères (Category+Ambience) associés à ces évaluations. On considère que si un utilisateur a évalué un restaurant alors les catégories et ambiences associées à ce restaurant font partie des préférences de l'utilisateur. On note Γ_u l'ensemble des critères (catégories+ambiences) préférés par l'utilisateur u. Un critère c sera dans Γ_u si u a donné une évaluation pour un restaurant auquel le critère c est associé. On note Γ_r l'ensemble des critères (catégories+ambiences) associés au restaurant r. Le facteur « préférence » est calculée en utilisant le coefficient de Jaccard, c'est-à-dire $f_p(u,r) = \frac{|\Gamma_u \cap \Gamma_r|}{|\Gamma_u \cup \Gamma_r|}$, ainsi $f_p(u,r) \in [0.0, 1.0]$.

3. facteur « social », lié aux avis donnés par les amis de l'utilisateur (à condition que l'utilisateur ait des amis bien sûr). Ce facteur retranscrit l'intuition que plus un restaurant r est apprécié par des personnes proches de u, plus l'utilisateur u sera susceptible d'être intéressé par ce restaurant. Ce facteur social, noté $f_s(u,r)$, est calculé comme la moyenne des notes attribuées par les amis (lorsqu'ils existent) de u, et 0 sinon.

4. facteur « commentaire », lié à la similarité entre les commentaires associés aux bonnes évaluations (5.0) données par l'utilisateur et les commentaires laissés par les autres utilisateurs. Ce facteur retranscrit l'intuition qu'un utilisateur u aura tendance à être intéressé par un restaurant r pour lesquels des commentaires proches de ceux rédigés par l'utilisateur pour les restaurants qu'il a appréciés ont été donnés. Notons $V_{u,r}$ le vocabulaire du commentaire donné sur le restaurant r (considéré comme un sac de mots) par l'utilisateur u. Le facteur « commentaire » est basé sur le coefficient de Jaccard défini entre deux vocabulaires, c'est-à-dire, $Jaccard(V_{u,r}, V_{u',r'}) = \frac{|V_r \cap V_{r'}|}{|V_r \cup V_{r'}|}$. Le facteur thématique, noté $f_c(u,t)$, est ainsi calculé comme suit :

   $f_c(u,r) = \max_{r' \in \Delta_u, u' \in \Upsilon} Jaccard(V_{u,r'}, V_{u',r})$

   où Δ_u est l'ensemble des restaurants notés 5.0 par l'utilisateur u et Υ'_r est l'ensemble des utilisateurs qui ont rédigé un commentaire sur le restaurant r

Finalement, le score d'un restaurant r est défini comme une somme pondérée des différents facteurs définis ci-dessus, comme suit :

$$score(u,r) = \alpha \times f_s(u,r) + \beta \times f_p(u,r) + \gamma \times f_s(u,r) + \lambda \times f_c(u,r)$$

Dans ce TP nous utiliserons les valeurs suivantes pour les constantes : $\alpha = 0,30$, $\beta = 0,35$, $\gamma = 0,20$, $\lambda = 0,15$. Nous vous demandons de produire une application, reposant sur le système NoSQL MongoDB, prenant en entrée un utilisateur et une ville présents dans la base (vous renverrez un message d'erreur dans le cas contraire) et retournant les restaurants (max. 20) recommandés à l'utilisateur donné situé dans la ville donnée. Les recommandations seront données dans l'ordre de préférence avec les scores associés. Le temps de calcul de la recommandation sera également indiquée.

Vous utiliserez le langage Python. et la librairie `pymongo`. Le code ci-dessous vous montre comment interfacer vos programmes Python avec une base MongoDB.

```python
from pymongo import MongoClient

try:
   conn = MongoClient()
   print("Connected successfully!!!")
except:
   print("Could not connect to MongoDB")

# database name: mydatabase
db = conn.mydatabase

# Created or Switched to collection names: myTable
collection = db.myTable

# To find() all the entries inside collection name 'myTable'
cursor = collection.find()
for record in cursor:
   print(record)
```

Un peu d'aide: `https://www.mongodb.com/languages/python`. Il vous faut bien sûr installer la librairie `pymongo`. Vous trouverez sur le Web toute l'aide nécessaire à la définition du code Python de l'application de recommandation de restaurants souhaitée.

3 Dossier de réponse : éléments à fournir

Le dossier à fournir (sous forme d'archive déposées sur Moodle) comme réponse à l'appel d'offre ci-dessus par votre équipe de développement (2 personnes max.) devra comprendre les éléments suivants, soit inclus le rapport principal (fichier PDF) soit sous forme de fichiers séparés :

1. **Modélisation :** description de la modélisation retenue en expliquant les choix effectués (cf. section 1). Ceci sera inclus dans le rapport.
2. **Importation des données :** script(s) Python (pas de notebook) pour l'importation des données (cf. section 1.1 en fonction de la modélisation retenue en expliquant les choix effectués. Chaque script sera fourni dans un fichier séparé (pas de notebook).
3. **Interrogation des données :** définitions des requêtes d'interrogation MongoDB répondant aux besoins exprimés en section 1.3 et résultats d'exécution de ces requêtes (copie d'écran) sur les données importées. Ceci sera inclus dans le rapport.
4. **Application :** script Python (pas de notebook) correspondant au moteur de recommandation de resturants décrit en section 2. Ce script sera fourni dans un fichier séparé (pas de notebook).
5. **Application :** exécution du moteur de recommandation pour les 5 exemples suivants. Ceci sera inclus dans le rapport en indiquant pour chaque cas (utilisateur, ville) la liste des restaurants recommandés avec les scores associés ainsi que le temps de calcul de la recommandation.
   - Utilisateur : kLm3B6LeboEZWX_4otpILA – Izzat, Ville : Greenville
   - Utilisateur : kLm3B6LeboEZWX_4otpILA – Izzat, Ville : Newark
   - Utilisateur : IpLRJY4CP3fXtlEd8Y4GFQ – Robyn, Ville : Wilmington
   - Utilisateur : IpLRJY4CP3fXtlEd8Y4GFQ – Robyn, Ville : New Castle
   - Utilisateur : GG0mFsEXb-02_dzFPqRV1Q – Mandy, Ville : Wilmington