TD MongoDB

Analyse de données Twitter

1 Description du cas d'étude

Les administrateurs de la plateforme Twitter envisagent d'ajouter une fonctionnalité pour faire de la recommandation de personnes à suivre aux utilisateurs de la plateforme. Une réflexion succinte sur cette recommandation les a conduit à proposer de calculer le score d'intérêt de chaque utilisateur $u_i (\in U$ l'ensemble des utilisateurs) pour l'utilisateur destinataire de la recommandation. Les u_i ayant le meilleur score d'intérêt seront ensuite recommandés à l'utilisateur.

Le score d'intérêt de l'utilisateur u_2 pour l'utilisateur u_1 est calculé de la façon suivante:

$$scoreInteret(u_2/u_1) = score_1(u_2) + score_2(u_2, u_1) + score_3(u_2) + score_4(u_2, u_1) + score_5(u_2, u_1) \quad (1)$$

où :

$$score_1(u_2) = \frac{nbFollowers(u_2)}{max_i(nbFollowers(u_i))}, \forall i \in U \quad (2)$$

$$score_2(u_2, u_1) = \begin{cases} \frac{nbFollowersEnCommun(u_2,u_1)}{max_i(nbFollowersEnCommun(u_i,u_1))} & \text{si } max_i(nbFollowersEnCommun(u_i, u_1)) <> 0 \\ 0 \text{ sinon} \end{cases} \quad (3)$$

$$score_3(u_2) = \frac{nbTweets(u_2)}{max_i(nbTweets(u_i))}, \forall i \in U \quad (4)$$

$$score_4(u_2, u_1) = \begin{cases} \frac{nbhashtagsEnCommun(u_2,u_1)}{max_i(nbhashtagsEnCommun(u_i,u_1))} & \text{si } max_i(nbHashtagsEnCommun(u_i, u_1)) <> 0 \\ 0 \text{ sinon} \end{cases} \quad (5)$$

$$score_5(u_2, u_1) = \begin{cases} \frac{nbTermessEnCommun(u_2,u_1)}{max_i(nbTermesEnCommun(u_i,u_1))} & \text{si } max_i(nbTermesEnCommun(u_i, u_1)) <> 0 \\ 0 \text{ sinon} \end{cases} \quad (6)$$

2 Modélisation

En tant que futur(e) « data scientist », en considérant :

- les données accessibles sur la plateforme Twitter (voir document de présentation),
- les objectifs décrits ci-dessus,
- les interrogations type ci-dessous (Q1 à Q10),

quelle modélisation orientée document mettriez-vous en oeuvre ?

3 Interrogation

On suppose la base **users** implémentée dans MongoDB suivant la modélisation définie précédemment (cf. section 2). Donnez les requêtes MongoDB permettant de répondre **aux besoins Q1 à Q10** :

Q1. récupérer les tweets de l'utilisateur @Alice.

Q2. récupérer les noms des utilisateurs ayant plus de 1000 ou moins de 10 followers.

Q3. récupérer les noms des followers de l'utilisateur @Alice.

Q4. récupérer les hashtags des tweets de l'utilisateur @Alice.

Q5. récupérer les identifiants des tweets contenant le hashtag #AliceInWonderland.

Q6. récupérer le nombre de followers de l'utilisateur @Alice.

Q7. récupérer le nombre de tweets de l'utilisateur @Alice.

Q8. connaître le nombre de hashtags différents par utilisateur.

Q9. récupérer les noms des followers communs entre l'utilisateur @Alice et l'utilisateur @theHatter.

Q10. connaître le nombre d'utilisateurs ayant plus de 1000 ou moins de 10 followers.

4 Comparaison au modèle relationnel

Comparez et discutez les réponses aux questions précédentes utilisant le langage Cypher à celles présentées ci-dessous en SQL. Pour rappel, le modèle relationnel considéré est le suivant :

```
User (idUser, screenName, name, description, createdAt, url, location, lang)
Tweet (idTweet, text, createdAt, urlEnd, source, lang, nbFavorites, #idUser)
Tweet_Url (#idTweet url, indiceStart, indiceEnd, urlExpanded)
Tweet_Media(#idTweet, idMedia, indiceStart, indiceEnd, type, url, urlExpanded, urlMedia)
Tweet_Hashtag(#idTweet, hahstag, indiceStart, indiceEnd)
Tweet_Mention(#idTweet, #idUser, indiceStart, indiceEnd)
User_Follow(#iduser, #idUserFollow)
Tweet_retweet(#idTweet, #idUser, idRetweet, createdAt, urlEnd, source)
```

```
[Q1] select count(*) from User;

[Q2] select count(*) from Tweet_Hashtag
      where hashtag='WhiteRabbit';

[Q3] select count(distinct idUser)
      from Tweet T, Tweet_Hashtag TH
      where T.idTweet = TH.idTweet
      and hashtag='AliceInWonderland';

[Q4] select U2.screenName
      from User U1, User U2, User_Follow UF
      where U1.screenName='Alice'
      and U1.iduser = UF.idUser
      and UF.idUserFollow = U2.idUser;

[Q5] select U2.screenName
      from User U1, User U2, User_Follow UF
```

```
      where U1.screenName='Alice'
      and U1.iduser = UF.idUserFollow
      and UF.idUser = U2.idUser;
```

```
[Q6] select screenName
      from User, User_Follow UF
      where U.idUser = UF.idUser
      group by U.idUser, screenName
      having count(*)> 10;
```

```
[Q7] select U2.screenName
      from User U1, User U2, User_Follow UF
      where U1.screenName='Alice'
      and U1.iduser = UF.idUser
      and UF.idUserFollow = U2.idUser
      insersect
      select U2.screenName
      from User U1, User U2, User_Follow UF
      where U1.screenName='Alice'
      and U1.iduser = UF.idUserFollow
      and UF.idUser = U2.idUser;
```

```
[Q8] select *
      from (select *
        from tweet
        order by nbFavorites desc)
      where rownum <11;
```

```
[Q9] select * from (select hashtag
        from Tweet_Hashtag
        group by hashtag
        order by count(*) desc)
      where rownum <11;
```

```
[Q10] select U5.screenName
      from User U1, User U4,
      User_Follow UF1, User_Follow UF2, User_Follow UF3, User_Follow UF4,
      where U1.screenName='Alice'
      and U1.iduser = UF1.idUser
      and UF1.idUserFollow = UF2.idUser
      and UF2.idUserFollow = UF3.idUser
      and UF3.idUserFollow = UF4.idUser
      and UF4.idUserFollow = U4.idUser;
```

[Q11]

Similaire à [Q10] mais non généralisable en utilisant seulement SQL...