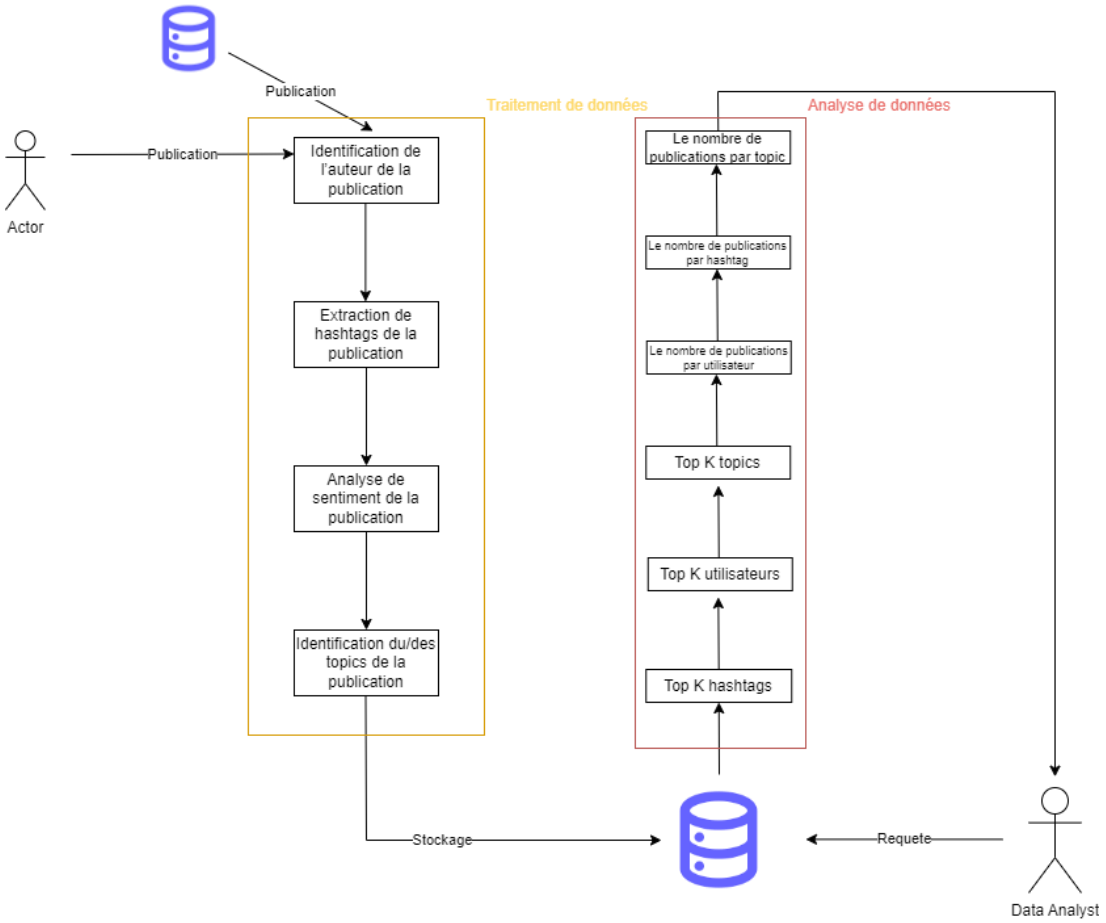
Gwennael CANNENPASSE 21804324

Sujet : Une application de traitement et d'analyse de tweet stocké en local via MongoDB

Description

Cette application se décline sous 2 processus le traitement de données, analyse de données.



L'utilisateur va publier un tweet et cela va démarrer le processus traitement de données. De plus pour les publications déjà données par votre fichier JSON nous les incorporons directement dans le processus.

Dans un premier temps, l'application va devoir identifier l'auteur de la publication ici un id.

Puis dans un second temps, il va devoir extraire les hashtags de la publication.

Par la suite, il faudra effectuer l'analyse de sentiments de la publication.

Enfin, nous identifierons des topics de la publication.

Tout cela sera enregistré dans une nouvelle base de données afin que le data analyst puisse analyser les données.

Gwennael CANNENPASSE 21804324

Implementation

Service de récupération de l'auteur :

```
class Recup_Auteur_Service(ServiceBase):
    @rpc(Unicode, _returns=Unicode)
    def Recup_Auteur(ctx, id_publication):
        result = client["SOA"]["TWEETS"].find_one({"id":id_publication}, {"_id": 0, "text": 1, "author_id": 1})
        return result.get("author_id")
```

Service de récupération des hashtags :

```
class Recup_Hashtags_Service(ServiceBase):
    @rpc(Unicode, _returns=Array(Unicode))
    def Recup_Hashtags(ctx, id_publication):
        result = client["SOA"]["TWEETS"].find_one({"id":id_publication}, {"_id": 0, "text": 1, "author_id": 1})
        publication = result.get("text")
        Hashtags = re.findall(r"(#+[a-zA-Z0-9(_)]{1,})", publication)
        return Hashtags
```

Nous utilisons une regex expression pour déterminer les hashtags d'un tweet.

Service d'analyse de sentiment :

```
class Analyse_Sentiment_Service(ServiceBase):
    @rpc(Unicode, _returns=float)
    def Analyse_Sentiment(ctx, id_publication):
        result = client["SOA"]["TWEETS"].find_one({"id":id_publication}, {"_id": 0, "text": 1, "author_id": 1})
        publication = result.get("text")
        blob = TextBlob(publication,pos_tagger=PatternTagger(), analyzer=PatternAnalyzer())
        return blob.sentiment.polarity
```

Nous appliquons le module blob et récupérons la polarity compris entre -1 et 1.

Service d'identification de topics :

```
class Identification_Topics_Service(ServiceBase):
    @rpc(Unicode, _returns=Array(Unicode))
    def Identification_Topics(ctx, id_publication):
        result = client["SOA"]["TWEETS"].find_one({"id":id_publication}, {"_id": 0, "text": 1, "author_id": 1})
        Topics = ["Politique", "Scientifique", "Finance", "Santé", "Culture", "People", "Sport"]
        Pics = random.sample(Topics, random.randint(1, 4))
        return Pics
```

Nous associons au tweet entre 1 et 4 topics au hasard parmi une liste de topics

Gwennael CANNENPASSE 21804324

Service de traitement des données :

```python
class Traitement_Donnees_Service(ServiceBase):
    @rpc(Unicode, _returns=Unicode)
    def Traitement_Donnees(ctx, id_publication):
        result = client["SOA"]["TWEETS"].find_one({"id":id_publication}, {"_id": 0, "text": 1, "author_id": 1})
        auteur = result.get("author_id")
        Hashtags = Recup_Hashtags_Service.Recup_Hashtags(ctx,id_publication)
        Feel = Analyse_Sentiment_Service.Analyse_Sentiment(ctx,id_publication)
        Pics = Identification_Topics_Service.Identification_Topics(ctx,id_publication)
        key = {"Publication": id_publication}
        data = {"Publication": id_publication, "Auteur": auteur, "Hashtags": Hashtags, "Sentiments": Feel,
                "Topics": Pics};
        client["SOA"]["TraitementDATA"].update_one(key, {"$setOnInsert": data}, upsert=True)
        return "Auteur: " + auteur+"\n"+"Hashtags: "+', '.join(Hashtags)+"\n"+"Topics: "+', '.join(Pics)+"\n"+"Sentiments: "+str(Feel)
```

Il fait l'appel de tous les autres services et enregistre les informations dans une autre base de données « TraitementDATA ».

Service Top K Hashtags :

```python
class Top_K_Hashtags_Service(ServiceBase):
    @rpc(int, _returns=Array(AnyDict))
    def Top_K_Hashtags(ctx, K):
        Top_k_Hashtag = client["SOA"]["TraitementDATA"].aggregate([{"$unwind": "$Hashtags"},
                                                                   {"$group": {"_id": "$Hashtags",
                                                                               "Ocurence": {"$sum": 1}}},
                                                                   {"$project": {"_id": 0, "Ocurence": 1,
                                                                                 "Hashtags": "$_id"}},
                                                                   {"$sort": {"Ocurence": -1}}, {"$limit": K}])
        liste = [val for val in Top_k_Hashtag]
        return liste
```

Il effectue une requête pour récupérer les Hashtags et leur nombre d'occurrence.

Résultat : J'ai rajouté dans la BD le #toto a plusieurs tweets

```
(anyTypeArray){
   anyType[] =
      (anyType){
         Ocurence = "6"
         Hashtags = "#toto"
      },
      (anyType){
         Ocurence = "3"
         Hashtags = "#CIV"
      },
      (anyType){
         Ocurence = "2"
         Hashtags = "#SupportriceMazo"
      },
      (anyType){
         Ocurence = "2"
         Hashtags = "#domie"
      },
      (anyType){
         Ocurence = "1"
         Hashtags = "#versailles"
      },
}
```

Gwennael CANNENPASSE 21804324

Service Top K utilisateurs :

```
class Top_K_utilisateurs_Service(ServiceBase):
    @rpc(int, _returns=Array(AnyDict))
    def Top_K_utilisateurs(ctx, K):
        Top_k_auteur = client["SOA"]["TraitementDATA"].aggregate([
            {"$group": {"_id": "$Auteur", "NB_Tweets": {"$sum": 1}}}, {"$sort": {"NB_Tweets": -1}}, {"$limit": K}])
        liste = [val for val in Top_k_auteur]
        return liste
```

Il effectue une requête qui compte le nombre d'occurrence de chaque utilisateur donc son nombre de publication.

Résultat :

```
(anyTypeArray){
   anyType[] =
      (anyType){
         _id = "1339914264522461187"
         NB_Tweets = "4"
      },
      (anyType){
         _id = "992984738516717570"
         NB_Tweets = "4"
      },
      (anyType){
         _id = "717025418"
         NB_Tweets = "2"
      },
      (anyType){
         _id = "3169236915"
         NB_Tweets = "2"
      },
      (anyType){
         _id = "372993152"
         NB_Tweets = "2"
      },
}
```

Service Top K topics :

```
class Top_K_Topics_Service(ServiceBase):
    @rpc(int, _returns=Array(AnyDict))
    def Top_K_Topics(ctx, K):
        Top_k_Topicss = client["SOA"]["TraitementDATA"].aggregate([{"$unwind": "$Topics"},
            {"$group": {"_id": "$Topics", "Ocurence": {"$sum": 1}}},
            {"$project": {"_id": 0, "Ocurence": 1,"Topics": "$_id"}},
            {"$sort": {"Ocurence": -1}}, {"$limit": K}])
        liste = [val for val in Top_k_Topicss]
        return liste
```

Il effectue une requête pour récupérer les Topics et leur nombre d'occurrence.

Résultat :

```
(anyTypeArray){
   anyType[] =
      (anyType){
         Ocurence = "10"
         Topics = "People"
      },
      (anyType){
         Ocurence = "9"
         Topics = "Culture"
      },
      (anyType){
         Ocurence = "8"
         Topics = "Finance"
      },
      (anyType){
         Ocurence = "8"
         Topics = "Santé"
      },
      (anyType){
         Ocurence = "7"
         Topics = "Sport"
      },
}
```

Service nombre de publications par utilisateur :

```
class  nombre_de_publications_par_utilisateur_Service(ServiceBase):
    @rpc(_returns=Array(AnyDict))
    def nombre_de_publications_par_utilisateur(ctx):
        nombre_de_publications_par_utilisateurs = client["SOA"]["TraitementDATA"].aggregate([
            {"$group": {"_id": "$Auteur", "NB_Tweets": {"$sum": 1}}},{"$project": {"_id": 0, "Auteur": "$_id","NB_Tweets": 1}}])
        liste = [val for val in nombre_de_publications_par_utilisateurs]
        return liste
```

Ici nous reprenons le code du service top K utilisateurs mais nous enlevons le K et le tri.

Service nombre de publications par hashtags :

```
class  nombre_de_publications_par_hashtags_Service(ServiceBase):
    @rpc(_returns=Array(AnyDict))
    def nombre_de_publications_par_hashtags(ctx):
        nombre_de_publications_par_hashtag = client["SOA"]["TraitementDATA"].aggregate([{"$unwind": "$Hashtags"},
                                                                {"$group": {"_id": "$Hashtags",
                                                                            "Ocurence": {"$sum": 1}}},
                                                                {"$project": {"_id": 0, "Ocurence": 1,
                                                                              "Hashtags": "$_id"}},
                                                                ])
        liste = [val for val in nombre_de_publications_par_hashtag]
        return liste
```

Ici nous reprenons le code du service top K hashtags mais nous enlevons le K et le tri.

Service nombre de publications par topics :

```
class  nombre_de_publications_par_topics_Service(ServiceBase):
    @rpc(_returns=Array(AnyDict))
    def nombre_de_publications_par_topics(ctx):
        nombre_de_publications_par_topicss = client["SOA"]["TraitementDATA"].aggregate([{"$unwind": "$Topics"},
            {"$group": {"_id": "$Topics", "Ocurence": {"$sum": 1}}},
            {"$project": {"_id": 0, "Ocurence": 1,"Topics": "$_id"}}])
        liste = [val for val in nombre_de_publications_par_topicss]
        return liste
```

Ici nous reprenons le code du service top K hashtags mais nous enlevons le K et le tri.

Service d’analyse des données :

```
class  Analyse_Donnees_Service(ServiceBase):
    @rpc(int,_returns=Array(AnyDict))
    def Analyse_Donnees(ctx, K):
        listeKuser = Top_K_utilisateurs_Service.Top_K_utilisateurs(ctx,K)
        listeKHashtags = Top_K_Hashtags_Service.Top_K_Hashtags(ctx,K)
        listeKTopics = Top_K_Topics_Service.Top_K_Topics(ctx,K)
        listeNBRUser = nombre_de_publications_par_utilisateur_Service.nombre_de_publications_par_utilisateur(ctx)
        listeNBRHashtags = nombre_de_publications_par_hashtags_Service.nombre_de_publications_par_hashtags(ctx)
        listeNBRTopics = nombre_de_publications_par_topics_Service.nombre_de_publications_par_topics(ctx)
        liste = listeKuser + listeKHashtags + listeKTopics + listeNBRUser + listeNBRHashtags + listeNBRTopics
        return liste
```

Il fait l’appel de tous les autres services d’analyse de données et renvoi la concaténation de tous les résultats.

Implementation du client :

```
def test_Recup_Auteur(id_publication):
    Recup_Auteur_client = Client("http://127.0.0.1:8000/Recup_Auteur_Service?wsdl")
    print(Recup_Auteur_client.service.Recup_Auteur(id_publication))
def test_Recup_Hashtags(id_publication):
    Recup_Hashtags_client = Client("http://127.0.0.1:8000/Recup_Hashtags_Service?wsdl")
    print(Recup_Hashtags_client.service.Recup_Hashtags(id_publication))
def test_Analyse_Sentiment(id_publication):
    Analyse_Sentiment_client = Client("http://127.0.0.1:8000/Analyse_Sentiment_Service?wsdl")
    print(Analyse_Sentiment_client.service.Analyse_Sentiment(id_publication))
def test_Identification_Topics(id_publication):
    Identification_Topics_client = Client("http://127.0.0.1:8000/Identification_Topics_Service?wsdl")
    print(Identification_Topics_client.service.Identification_Topics(id_publication))
def test_Traitement_Donnees(id_publication):
    Traitement_Donnees_client = Client("http://127.0.0.1:8000/Traitement_Donnees_Service?wsdl")
    print(Traitement_Donnees_client.service.Traitement_Donnees(id_publication))
def test_Top_K_utilisateurs(K):
    Top_K_utilisateurs_client = Client("http://127.0.0.1:8000/Top_K_utilisateurs_Service?wsdl")
    print(Top_K_utilisateurs_client.service.Top_K_utilisateurs(K))
def test_Top_K_Hashtags(K):
    Top_K_Hashtags_client = Client("http://127.0.0.1:8000/Top_K_Hashtags_Service?wsdl")
    print(Top_K_Hashtags_client.service.Top_K_Hashtags(K))
def test_Top_K_Topics(K):
    Top_K_Topics_client = Client("http://127.0.0.1:8000/Top_K_Topics_Service?wsdl")
    print(Top_K_Topics_client.service.Top_K_Topics(K))
def test_nombre_de_publications_par_utilisateur():
```

Appel des différents services.

Connection aux BD :

```
uri = 'mongodb+srv://admin:uvsqawsgroupe17@cluster0.nkdni.mongodb.net/?retryWrites=true&w=majority'
client = pymongo.MongoClient(uri)
```