# Table des matières

# Introduction

Ce document permet de recenser les différents fichiers nécessaires à la partie Tweet Collector du projet Tweetoscope ainsi que le fonctionnement de chacun de ces fichiers.

# Note de fonctionnement de Tweet Collector :

## Algorithme de fonctionnement de Tweet Collector

1. Récupérer un tweet/retweet (=message) depuis le topic « tweet »
2. Obtenir l'attribut « source » du message pour l'utilisation d'une map (équivalent d'un dictionnaire python). La clé de la map étant l'identifiant du message (unique pour chaque tweet) et la valeur étant un vecteur de pair d'entier
3. Ajouter dans la map le doublet « time », « magnitude »
4. Quand ti-t0 > 600, on envoie la série partielle sur le topic « cascade_series »
5. Arrêter d'ajouter des éléments dans le conteneur quand le temps du message en cours est supérieur à 1000 + temps du dernier élément du conteneur
6. Supprimer de la map la cascade

myMap : std ::map<int = key,std ::vector<std ::pair<int = time, int = magnitude>>>

## Cascade_series

C'est un topic permettant d'envoyer les 600 premières secondes de la cascade (cascade partiel).

Le message est structuré de la façon suivante :

- type : le type du message (str constant égale à « serie »)
- cid : l'identifiant de la cascade (str)
- T_obs : le temps observation de la cascade (int), d'après le cahier des charges ce temps doit valoir au maximum 10 minutes
- tweets : un tuple contenant une paire « time », « magnitude »

Ce message doit être réceptionner par le programme « Hawkes_Process » afin de prédire la viralité d'un tweet.

## Cascade_properties

C'est un topic permettant d'envoyer le nombre total de retweet.

Le message est structuré de la façon suivante :

- type : le type du message (str constant égale à « size »)
- cid : l'identifiant de la cascade (str)
- n_tot : la taille finale de la cascade (int)
- t_end : le temps de réception du dernier tweet (int)

On considère que la cascade est terminée lorsque la différence entre le temps actuel et le temps du dernier retweet est supérieur à 1000.

# Architecture fonctionnelle de la class Tweet

Pour la réalisation de cette partie du projet, on a créé une class « Tweet ».

Cette classe contient les méthodes suivantes :

- getType() : pour obtenir l'attribut « type » du message venant du topic « tweet »

- getMsg() : pour obtenir l'attribut « msg » du message venant du topic « tweet »
- getTime() : pour obtenir l'attribut « t » du message venant du topic « tweet »
- getMagnitude() : pour obtenir l'attribut « m » du message venant du topic « tweet »
- getSource() : pour obtenir l'attribut « source » du message venant du topic « tweet »
- getInfo() : pour obtenir l'attribut « info » du message venant du topic « tweet »
- Tweet(string jsonString) : permet de séparer le message en les différents attributs ci-dessus
- Tweet(string type, string msg, int time, int magnitude, int source, string info) : un constructeur
- delta(map, key) : qui donne la différence entre le temps d'émission du tweet et le temps d'émission du dernier retweet
- tuple_to_string(map,key) : qui transforme le vecteur de pair d'entier en une chaîne de caractère (utilisé pour l'envoi de message Kafka)
- add_in_map(map, key, time, magnitude) : pour ajouter le doublet (time,magnitude) dans la map
- erase_map(map,key) : pour supprimer la cascade de retweet (pour ne pas saturé la mémoire)
- last_time(map,key)

### getTime()

| |
|---|
| Variable d'entrée : |
| Variable sortie :<br>• L'attribut « time » du message Kafka du topic « tweet » |
| Algorithme : |

### getMagnitude()

| |
|---|
| Variable d'entrée : |
| Variable de sortie :<br>• L'attribut « magnitude » du message Kafka du topic « tweet » |
| Algorithme : |

### getSource()

| |
|---|
| Variable d'entrée : |
| Variable de sortie :<br>• L'attribut « source » du message Kafka du topic « tweet » |
| Algorithme : |

### getInfo()

| |
|---|
| Variable d'entrée : |
| Variable de sortie :<br>• L'attribut « time » du message Kafka du topic « tweet » |
| Algorithme : |

### Tweet(jsonString)

| |
|---|
| Variable d'entrée :<br>• Un message Kafka |
| Variables de sortie :<br>• Les différents attributs du message Kafka du topic « tweet » |
| Algorithme :<br>1. On récupère différentes chaînes de caractères correspondant à chacun des attributs du message Kafka |

## delta(map,key)

| |
|---|
| Variable d'entrée :<br>• Une map c++<br>• Une clé de la map |
| Variable sortie :<br>• d : la différence de temps entre le premier tweet et le dernier retweet |
| Algorithme :<br>1. On récupère le vecteur contenu dans map[key]<br>2. On récupère la première et la dernière valeur du vecteur |

## tuple_to_string(map,key)

| |
|---|
| Variable d'entrée :<br>• Une map c++<br>• Une clé de la map |
| Variable sortie :<br>• str : l'équivalent de la cascade en une chaine de caractère |
| Algorithme :<br>1. On récupère le vecteur contenu dans map[key]<br>2. On transforme chaque doublet d'entier en une chaine de caractère |

## add_in_map(map,key,time,magnitude)

| |
|---|
| Variable d'entrée :<br>• Une map c++<br>• Une clé de la map<br>• L'attribut « time » du message Kafka actuel<br>• L'attribut « magnitude » du message Kafka actuel |
| Variable sortie :<br>• map : la map d'entrée où l'on a ajouté le doublet (time, magnitude) |
| Algorithme :<br>1. On récupère le vecteur contenu dans map[key]<br>2. On écrase le vecteur contenu dans map[key] par le même vecteur avec le doublet (time, magnitude) ajouté à la fin du vecteur |

## erase_map(map,key)

| |
|---|
| Variable d'entrée :<br>• Une map c++<br>• Une clé de la map |
| Variable sortie :<br>• map: une map, où l'on a effacé les données contenues dans map[key] |
| Algorithme :<br>1. On récupère le vecteur contenu dans map[key]<br>2. On transforme chaque doublet d'entier en une chaine de caractère |

10/11/2020 : Cette méthode n'est pas encore utilisée dans le code et sert à ne pas saturer la mémoire

## last_time(map,key)

| |
|---|
| Variable d'entrée :<br>• Une map c++<br>• Une clé de la map |
| Variable sortie :<br>• dernier_temps : le temps du dernier élément de la cascade |
| Algorithme : |

1. On récupère le vecteur contenu dans map[key]
2. On transforme chaque doublet d'entier en une chaine de caractère