Université Claude Bernard Lyon I

Master Data Science

Projet Data Mining

Détection d'événements sur Twitter

Auteurs :
Gregory Howard *11207726*
Marine Ruiz *11208141*
Jules Sauvinet *11412086*

Professeur :
Marc Plantevit

22 janvier 2017

Résumé

La détection d'évènements est un des sujets de recherche les plus importants dans l'analyse des réseaux sociaux. Les flux de données provenant de ces plates-formes contiennent beaucoup d'informations permettant de faire de la détection d'évènements. Néanmoins, une grande partie des données à traiter est bruitée, notamment par des comptes Twitter entretenus par des robots qui tweetent abondamment et qui ont une fréquence de publication constante [1]. Ces robots peuvent dissimuler l'information d'un réel évèvenemnt voire simuler la création d'un faux évènement. Voila pourquoi il est important de comprendre comment atténuer l'influence du bruit pour faire de la détection évènements. Notre objectif ici est de détecter des évènements en exploitant les informations spatio-temporelles et textuelles de tweets en essayant d'effectuer un pré-filtrage des tweets de robots.

Table des matières

1 Introduction

Notre objectif est de détecter des évènements à partir de tweets en exploitant des informations spatio-temporelles et textuelles. Notre approche utilise la technique de détection d'événements multi-échelles dans les réseaux sociaux introduite dans l'article de recherche [2] et développée par Ahmed Anes Bendimerad et Aimene Belfodil. Nous avons adapté cette version à nos données à laquelle nous avons ajouté des mécanismes permettant de filtrer les tweets envoyés par des robots en amont du clustering permettant la détection d'évènements. Les comptes automatisés, appelés robots, abusent des réseaux sociaux en affichant du contenu non-éthique utilisant leur compte pour tenter de faire de la vente par exemple. Ces robots bruitent l'information du déroulement d'un évènement et rendent la détection plus compliquée.

Nous avons également développé une approche de détection spatio-temporelle d'évènements avec l'algorithme de clustering DBScan, puis nous avons comparé les résultats des deux types de clustering.

Nous appelons évèvenement (au sens ici évèvenement que nous souhaitons détecter) un phénomène physique survenant en un point et pendant une durée bien déterminée. Un événement est intéressant s'il suscite un nombre suffisant de tweets. Ces tweets peuvent se situer sur le lieu de l'évènement ou bien ailleurs, notamment si celui-ci est retransmis par les médias. Les tweets sur l'évènement peuvent se produire avant, pendant et après l'événement. Ainsi, la détection d'évènements doit réussir à intégrer ces dispersions pour localiser dans le temps et l'espace les évènements et c'est ce qui la rend si complexe.

2 Description de nos données

Nous avons effectué notre détection d'évèvenents à partir d'un fichier de 1,7 Go comportant 10 982 005 tweets récupérés sur Twitter. Ces tweets ont été postés dans l'agglomération newyorkaise ou par des utilisateurs newyorkais du *21 juillet 2015* au *16 novembre 2015*. Chaque tweet contient les informations suivantes :

— un identifiant
— la date et l'heure à laquelle le tweet a été posté
— la position de la personne quand le tweet a été posté (latitude et longitude)
— l'identifiant du lieu géographique où a été posté le tweet
— l'ensemble des références et hashtags du tweet
— l'identifiant de la personne qui a posté le tweet
— le pseudo de la personne qui a posté le tweet
— le nom complet de la personne qui a posté le tweet
— l'identifiant du lieu de création du compte de la personne qui a posté le tweet
— l'information de la certification ou non du compte de l'utilisateur ayant posté le tweet
— le nombre de personnes qui suivent la personne qui a posté le tweet
— le nombre d'amis de la personne qui a posté le tweet
— le nombre de tweet déjà posté par la personne qui a posté le tweet

Pour la détection d'évènements, nous nous sommes servi de l'identifiant du tweet, du nom de la personne qui a posté le tweet, des hashtags, de la date et de la position d'envoi du tweet.

3 Le pré-traitement des données

Nous avons utilisé Knime pour effectuer un premier filtrage des robots. Pour cela, nous classons dans l'ordre décroissant les utilisateurs en fonction du nombre de tweets qu'ils ont postés. Puis, nous supprimons les 0.5% premiers utilisateurs à partir de ce classement. Ce pré-traitement nous permet d'homogénéiser nos données mais aussi d'augmenter de façon évidente la pertinence des évènements détectés.

Nous avons basé ce choix de filtrage sur deux critères. D'une part, il y a une probabilité importante qu'un utilisateur qui poste en très grande quantité des tweets soit un robot. D'autre part, nous nous sommes dit qu'un utilisateur réel ayant une activité sur Twitter bien supérieure à celle des autres utilisateurs, avait en moyenne moins de contenu descriptif d'évènevements dans ses tweets et plus de "tweets bruits", c'est à dire des tweets nous donnant aucune information sur le déroulement d'un quelconque évènevement (e.g "#lol", ou encore "#oklm avec mon refré"). Nous avons ensuite confirmé cette intuition empiriquement en observant notre jeu de données.

Cette technique étant très approximative, nous ne l'utilisons que pour filtrer les utilisateurs qui publient, vraiment de façon évidente, plus que tous les autres utilisateurs. En fait, puisque nous faisons de la détection in fine et non de la suppression de compte, nous pouvons nous permettre d'avoir des faux positifs (utilisateurs considérés comme robot alors qu'ils ne le sont pas).

La dimension géographique étant cruciale dans notre méthode de clustering, nous avons également filtré les tweets ne possédant pas de position géographique dans le module Knime. Nous avons donc donné en entrée à Knime notre fichier de 1,7 Go contenant 10 982 005 tweets et nous avons obtenu, à la fin du traitement, un fichier de 250 Mo contenant 1 600 000 tweets.

Nous avons joint le workflow Knime robotFilter.knwf permettant de faire ce filtrage à l'archive de rendu du projet.

Enfin, notre jeu de données de tweets s'étale sur 103 jours. Puisque nous voulons faire de la détection d'évènements journaliers et localisés et que nous voulons aussi optimiser les performances de notre algorithme, nous avons découpé notre jeu de données initial en 103 sous-ensembles journaliers. Chaque sous-ensemble de tweets contient plus de 15 000 tweets.

4 Détection de robots selon la fréquence de tweets

Avant de lancer notre algorithme de clustering, nous appliquons un filtre supplémentaire pour détecter les robots que nous n'avons pas supprimé du premier filtre fait avec Knime.

4.1 Filtrage avec le coefficient de corrélation de Pearson

L'idée est de filter les utilisateurs qui tweetent de manière très synchronisée [3] d'un jour à l'autre, en supposant qu'un tel comportement est caractéristique de celui d'un robot.

Pour ce faire, pour chaque jour, nous récupérons les tweets de chaque utilisateur pour le jour courant et le jour d'après. Nous calculons ensuite la corrélation entre les deux séries temporelles correspondant aux instants d'envoi des tweets lors de la première journée et lors de la deuxième journée.

Si ce coefficient est supérieur à un seuil fixé (à 0.8 couramment), nous considérons que l'utilisateur à une forte probabilité d'être un robot et nous supprimons ses tweets du processus de clustering.

4.2 Filtrage avec une loi géométrique

L'idée ici est de filter les utilisateurs qui, sur une journée, postent des tweets de manière régulière. Nous avons considéré que ce comportement était, de manière complémentaire à ce qui a été dit au paragraphe précédent, caractéristique d'un comportement automatisé.

Pour ce faire, pour l'ensemble des tweets journaliers de chaque utilisateur, nous calculons les intervalles de temps d'envoi entre chaque paire de tweets, mais uniquement si le nombre de tweets de l'utilisateur est supérieur à 10. Nous trions ensuite le tableau en fonction du nombre d'occurence de chaque intervalle de manière décroissante (e.g $[10, 10, 10, 5, 5, 2, 7, 9]$. 10 est l'intervalle le plus fréquent, 5 est le deuxième intervalle le plus fréquent etc...). Pour finir, nous remplaçons les valeurs des intervalles dans le tableau précédent par leur ordre de fréquence (e.g [1, 1, 1, 2, 2, 3, 4, 5] pour le tableau précédent).

Ainsi, nous pouvons faire un test d'adéquation du $khi - deux$ entre le tableau obtenu et une distribution de loi géometrique de probabilité fixée à un certain seuil (0.23). Si la $p - value$ de ce test d'adéquation est supérieure à 0.20, nous considérons que la distribution des intervalles de temps est suffisamment proche d'une distribution "régulière". Nous classons alors l'utilisateur en question comme étant un robot. Nous supprimons donc, comme dans la méthode précédente, ses tweets pour le clustering.

4.3 Le principe de la blacklist

Chacun de nos événements est décrit par des hashtags pertinents. Ces hashtags sont définis comme étant les plus récurrents dans les tweets contenus dans l'événement. Cependant, certains hashtags sont présents dans la description de beaucoup d'autres événements, ce qui les rend moins pertinents. Nous voulons donc les détecter et ne pas les prendre en compte lors du clustering afin d'avoir une vraie description de l'événement.

Pour cela, nous réalisons un premier clustering sur nos données filtrées. Nous récupérons les descriptions de tous les évènements, puis nous calculons le nombre d'occurences de chaque hashtag. Nous obtenons alors, pour chaque hashtag, le nombre de tweets dans lequel il apparait. Nous avons défini, de manière empirique, que, si un hashtag apparait dans plus d'un évènement sur quatre, alors celui-ci est non pertinent.

Finalement, nous stockons tous les hashtags caractérisés comme non-pertinent dans une liste nommée *blacklist*. Ainsi, quand nous exécutons le second clustering, nous prenons soin de ne pas prendre en compte les hashtags *blacklistés*.

5 La détection d'évènements multi-échelle

5.1 Description de la méthode

La technique de détection d'évenements est celle évoquée dans l'article [2].

La première étape consiste en la création d'un graphe de similarité entre tous les tweets. Les sommets du graphe sont les tweets et les arcs sont valués par des poids indiquant la similarité entre deux tweets.

La métrique entre deux tweets est résumée ci-dessous Équation 1. Pour plus de détails sur le procédé, se référer à l'article [2].

$$S_2(t_i, t_j) = f_1(t_i, t_j) * f_2(t_i, t_j), \tag{1}$$

avec f_1 qui définit la similarité textuelle entre deux tweets t_i et t_j. Pour chaque terme partagé par t_i et t_j, une similarité entre les deux séries temporelles des instants d'apparition des termes dans des tweets est calculée.
f_2 définit alors la similarité maximale des séries temporelles entre les termes partagés par t_i et t_j.

La matrice d'adjacence est ensuite calculée comme suit :

$$W_2(i, j) = \begin{cases} S_2(t_i, t_j) & \text{si } i = j \\ 0 & \text{sinon} \end{cases} \tag{2}$$

Sur la base de ce graphe de similarité, la deuxième étape est l'application de la méthode de détection de communauté de Louvain. Elle permet de détecter des communautés formant des clusters d'évènements. L'algorithme complet de l'approche multiscalaire proposée pour la détection d'événements (dans l'article [2]) est résumée dans algorithm 1.

L'algorithme utilise des séries temporelles et spatiales pour déterminer des niveaux de correspondances entre deux tweets grâce notamment à la transformée discrète de Wavelet (DWT) qui permet de mesurer la similarité entre des séries de signaux - ici - temporels et spatiaux. L'espace est découpé en cellules géographiques et en séries temporelles. Chaque tweet est associé à différents patterns en fonction des distributions spatiales et temporelles auxquelles répondent les termes le constituant.

Une fois le processus de clustering terminé, nous ne gardons que les clusters pertinents. Un cluster est pertinent si le nombre de tweets le constituant et si le nombre de personnes ayant envoyé ces tweets sont assez importants (supérieurs à un certain seuil), et si la proportion de tweets par personne n'est pas éxagérée. Ce cluster devient alors un événement.

5.2 Explicitation de l'algorithme

Nous citons l'algorithme défini dans l'article [2] pour introduire ses principes généraux et montrer que son fonctionnement réside dans des concepts relativement compliqués. Nous n'avons pas eu la prétention de pouvoir refaire une solution existante fonctionnelle en un temps limité, et c'est pourquoi nous avons préféré la réutiliser.

Le principe peut être résumé ainsi :
"Pour que deux tweets soient considérés similaires, il faut qu'ils soient suffisamment similaire au regard d'au moins une des dimensions spatiale ou temporelle mais pas nécessairement des deux simultanément".

Algorithm 1: Multiscale Event Detection using Wavelets (MED)

1 **Input:**
  T : un ensemble de tweets avec des informations spatiales, temporelles et textuelles
  Δt : la résolution temporelle initiale (pour discrétiser l'espace temporel)
  Δd : la résolution spatiale initiale (pour discrétiser l'espace temporel)
  n_{scale} : nombres d'échelles spatiales
2 Pour chaque paire de tweets t_i et t_j dans T, extraire les termes communs $w_{i\,i=1}^{k}$
3 Pour chaque w_i, calculer en utilisant Δ_t la série temporelle de ses occurences en intégrant leur distance géographique (en utilisant Δ_d)) en utilisant les cellules àuxquelles t_i et t_j appartiennent (cela permet de calculer la similarité entre deux tweets en prenant en compte les échelles géographique et temporelle).
4 Appliquer DWT (la transformée discrète de Wavelet) aux deux séries temporelles, et calculer la similarité entre elles comme la corrélation entre un ensemble spécifique de coefficients DWT au niveau correspondant à S_s.
5 Calculer $s_{st}(t_i, t_j)$ comme la similarité maximale des séries temporelles parmi $w_{i\,i=1}^{k}$. Calculer $S_2(t_i, t_j)$ en utilisant Équation 1, et la matrice d'adjacence W_2 en utilisant Équation 2.
6 Appliquer la méthode non-récursive de Louvain á W_2, et ne retenir que les clusters les plus intéressants $c_{i\,i=1}^{m}$ après des étapes de post-traitement.
7 **Output:**
  $c_{i\,i=1}^{m}$: les clusters qui correspondent aux évènements à différentes échelles spatiales et temporelles.

5.3 Description d'un événement

Finalement, un événement est décrit par :

- Une heure de début et une heure de fin qui permettent de calculer une durée.
- Une position centrale (longitude, latitude) calculée à partir de la moyenne de toutes les positions (moyenne des longitudes, moyenne des longitudes).
- Le nombre de personnes différentes ayant postées des tweets.
- La liste des hashtags les plus importants permettant de le décrire.

6 Les options sur le clustering

6.1 L'élasticité

On suppose qu'un hashtag est fréquent pour un ensemble de tweets donné, s'il apparait plus d'un certain nombre de fois (fixé à 20). Ce nombre est statique, ce qui ne permet pas de détecter de petits événements dans une journée où il y a eu peu de tweets. A l'inverse, nous détectons beaucoup d'événements sur une journée où il y a eu beaucoup de tweets. Cela peut être intéressant, mais il est parfois plus judicieux de s'adapter au nombre de tweets postés dans une même journée. L'option d'élasticité permet cela.

L'activation de l'option d'élasticité définit cette fois-ci un hashtag fréquent s'il apparait dans plus de 20% des tweets du cluster de l'évènement. La valeur statique et le pourcentage peuvent être modulés.

6.2 La géolocalisation

Grâce au clustering MED algorithm 1, nous obtenons un ensemble de tweets dit "voisins" qui seront contenus dans le même événement à la fin de l'algorithme. Toutefois, nous obtenons des événements trés dispersés avec un rayon allant parfois jusqu'à 389 330,05 m.

Pour atténuer cette dispersion, nous utilisons la méthode *NearestNeighbors* (du package *scikitlearn.neighbors* de Python). Cette méthode prend en entrée notre ensemble de tweets journalier et une distance seuil. Nous obtenons en sortie, des clusters de tweets de rayon maximum notre distance seuil.

A la fin, nous faisant l'intersection entre le clustering MED et le clustering des plus proches voisins. Nous obtenons un cluster de tweets qui sont beaucoup plus proches géographiquement.

7 Résultats

Nous allons dans cette section exploiter et comparer certains des résultats que nous avons obtenus. Deuxièmement, nous allons mettre en évidence que nous filtrons des comptes dit robots.

7.1 Paramétrage du clustering

Nous allons comparer les résultats pour une même journée du *6 septempbre 2015*. Nous allons mettre en avant les spécificités que le paramétrage d'une option a pu apporter.

N.B. Les résultats ont été obtenus à partir d'un jeu de tweets plus petit car il était trop long de réaliser autant de sorties avec notre fichier complet. En outre, cela n'importe peu car nous pouvons observer l'influence des paramètres sur le jeu de données simplifié.

7.1.1 Sans option activée

```
date : 2015-09-06
----------------------------------------
4 Event detected :
----------------------------------------
->Temps median : 2015-09-06 08:00:00
|Duree estimee : 1140
|Position du centre de l'event : (-74.6219551895,41.4179940526)
|Rayon de l'event : 337263.18
|Nombre de twittos : 88
|Nombre de tweets : 95
|Hashtags importants : #job,#hiring,#careerarc,#jobs,#jerseycity,#albany,#rochester,#retail,#education,#newyork
|
********************************************************************************
->Temps median : 2015-09-06 08:08:00
|Duree estimee : 1560
|Position du centre de l'event : (-74.128950473,40.9225881351)
|Rayon de l'event : 394138.67
|Nombre de twittos : 54
|Nombre de tweets : 74
|Hashtags importants : #job,#hiring,#newyork,#careerarc,#jobs,#retail,#veterans,#sales,#marketing,#makeup
|
********************************************************************************
->Temps median : 2015-09-06 07:48:00
|Duree estimee : 2880
|Position du centre de l'event : (-73.9747555833,40.7562059167)
|Rayon de l'event : 6535.25
|Nombre de twittos : 18
|Nombre de tweets : 24
|Hashtags importants : #nyc,#misteremma,#architecture,#streetart,#newyork,#aesthetic,#house,#priceless,#statueofliberty,#labordayinthecity
|
********************************************************************************
->Temps median : 2015-09-06 08:15:00
|Duree estimee : 3540
|Position du centre de l'event : (-73.9850065714,40.7449247143)
|Rayon de l'event : 7002.41
|Nombre de twittos : 6
|Nombre de tweets : 7
|Hashtags importants : #nyc,#bronx,#day3,#iphone6,#manhattan,#marianorivera,#doulas,#carriagehousebirth,#classes,#stateofliberty
|
********************************************************************************
----------------------------------------
Elapsed time : 1.97099995613s
----------------------------------------
```

Fig. 1

Dans un premier temps, lorsqu'aucune option n'est activée, on obtient des évènements de tailles différentes, tant par le nombre de tweets qu'ils contiennent que par la taille de leur rayon.

7.1.2 Influence de l'élasticité

```
date : 2015-09-06
----------------------------------------
2 Event detected :
----------------------------------------
->Temps median : 2015-09-06 08:03:00
|Duree estimee : 1500
|Position du centre de l'event : (-74.4242489753,41.2211033642)
|Rayon de l'event : 360376.2
|Nombre de twittos : 128
|Nombre de tweets : 162
|Hashtags importants : #job,#hiring,#careerarc,#jobs,#newyork,#jerseycity,#albany,#retail,#rochester,#veterans
|
********************************************************************************
->Temps median : 2015-09-06 07:27:00
|Duree estimee : 840
|Position du centre de l'event : (-74.7195056667,41.1982306667)
|Rayon de l'event : 88767.83
|Nombre de twittos : 3
|Nombre de tweets : 3
|Hashtags importants : #job,#hiring,#careerarc,#hackensack,#jobs,#sales,#shamong,#retail,#waterville,#nursing
|
********************************************************************************
----------------------------------------
Elapsed time : 2.4880001545s
----------------------------------------
```

Fig. 2

En activant l'élasticité, un hashtag devient pertinent s'il apparait dans plus de 20% des tweets du cluster de l'évènement. Nous remarquons que certains hashtags comme *marketing, labordayinthecity, day3* ont disparu, permettant l'agglomération de plusieurs évènements afin d'en créer un plus grand. L'elasticité permet notamment d'être plus synthétique en ne renvoyant que les évènements les plus importants d'une journée.

7.1.3 Influence de la géolocalisation

```
date : 2015-09-06
---------------------------------------
12 Event detected :
---------------------------------------
->Temps median : 2015-09-06 08:06:00
|Duree estimee : 1560
|Position du centre de l'event : (-74.005973,40.714353)
|Rayon de l'event : 0.09
|Nombre de twittos : 23
|Nombre de tweets : 27
|Hashtags importants : #newyork,#job,#hiring,#careerarc,#jobs,#veterans,#marketing,#sales,#art,#physician
|
********************************************************************************
->Temps median : 2015-09-06 07:59:00
|Duree estimee : 900
|Position du centre de l'event : (-77.610922,43.16103)
|Rayon de l'event : 0.0
|Nombre de twittos : 14
|Nombre de tweets : 14
|Hashtags importants : #job,#rochester,#hiring,#jobs,#careerarc,#skilledtrade,#veterans,#accounting,#generalscience,#education
|
********************************************************************************
->Temps median : 2015-09-06 08:01:00
|Duree estimee : 1080
|Position du centre de l'event : (-73.971249,42.6285)
|Rayon de l'event : 0.0
|Nombre de twittos : 11
|Nombre de tweets : 12
|Hashtags importants : #albany,#job,#hiring,#careerarc,#jobs,#sales,#finance,#construction,#businessmgmt,#legal
|
********************************************************************************
```

Fig. 3

Les évènements détectés sont plus denses et précisément localisés géographiquement : la taille de leur rayon est plus faible voire nulle.

N.B. Nous n'affichons pas les 12 évènements détectés car cela n'est pas nécessaire pour constater les effets du paramètre de géolocalisation.

7.1.4 Interprétations

Chaque option a un sens selon le type de détection d'évèvenements voulu. La géolocalisation permet de privilégier la dimension géographique mais cela augmente considérablement le nombre d'évènements renvoyés (plusieurs évènements renvoyés codent le même évèvenement physique).
L'élasticité permet d'être plus pertinent sur le nombre d'évènements en en renvoyant moins. Cependant, cela va au détriment de la détection de petits évènements qui sont agglomérés dans des évènements plus importants voire même non détectés.

7.2 Filtrons-nous bien les robots ?

Nous nous sommes concentrés sur la journée du *11 septembre 2015*, jour symbolique aux Etats-Unis. Nous allons comparer pour cette journée les évènements obtenus par notre méthode avec et sans filtre. Nous avons 17 632 tweets pour ce jour.

7.2.1 Sans aucun filtre

Sans filtre, nous détectons 29 évènements. Nous obtenons deux types d'évènements bien différents (nous avons développé un module de visualisation pour visualiser plus facilement nos résultats) :

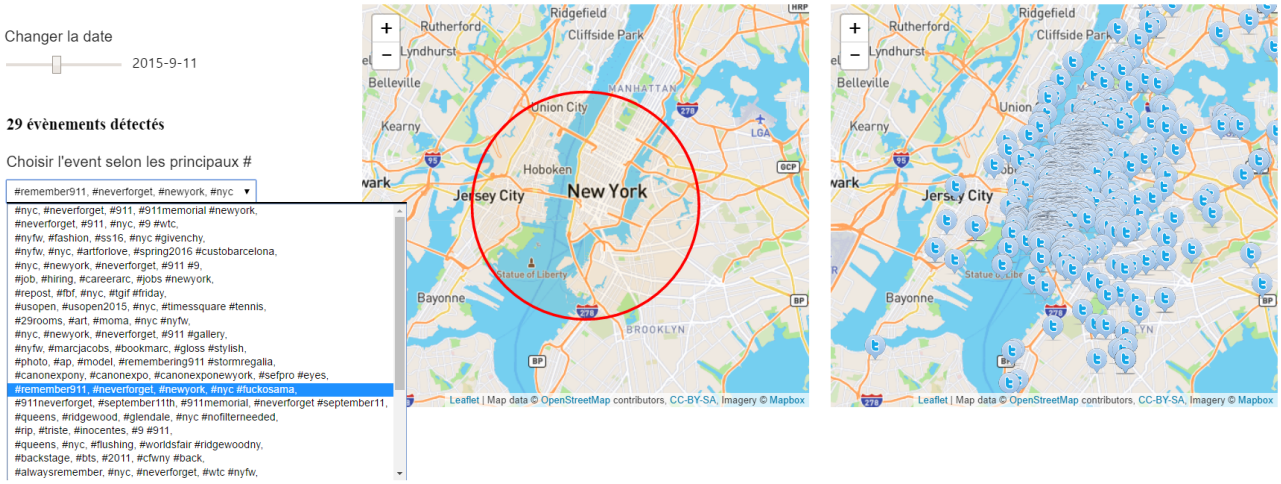


Fig. 4

Sur les 29 évènements :

- 8 décrivent le 11/09, mais ne sont pas nécessairement des événements physiques
- 3 décrivent un défilé de la marque Givenchy
- 3 sont à propos de jobs et d'embauches
- 2 sont à propos d'une maison de mode "29 Rooms"
- 1 décrit une exposition au Moma
- 1 à propos de l'US Open
- 1 à propos du vendredi
- 1 à propos d'une ligne de robe "custobarcelona"
- 1 à propos d'une librairie de Marc Jacobs
- 1 à propos d'une exposition canon
- 1 à propos de thé et de coiffure dans le Queens
- 1 rassemble les concepts "egypte", "style masculin" et le lieu de l'US Open
- 1 à propos de Rihanna et de photo

— 1 à propos de selfie
— 1 à propos de maquillage et de makeup artist
— 1 à propos d'un rassemblement à Central Park
— 1 à propos de sortie, du vendredi soir et de téquila

D'une part, on peut constater que les 29 évènements n'encodent pas tous de vrais évènements. D'autre part, ils ne sont pas toujours bien décrits. En effet, on s'aperçoit en comparant les descriptions que certains termes comme *#nyc* sont présents dans chacune d'entre elles. Ils rendent donc ces dernières peu intéressantes. De plus, ils occupent probablement la place de termes, utiliseés lors du clustering, qui seraient probablement plus pertinents pour décrire un évènement.

7.2.2 Avec filtre

Avec filtre (i.e. le premier filtre par Knime, la loi de Pearson, la loi Géométrique et le filtre des termes avec la "blacklist"), nous obtenons 10 évènements, c'est-à-dire trois fois moins.

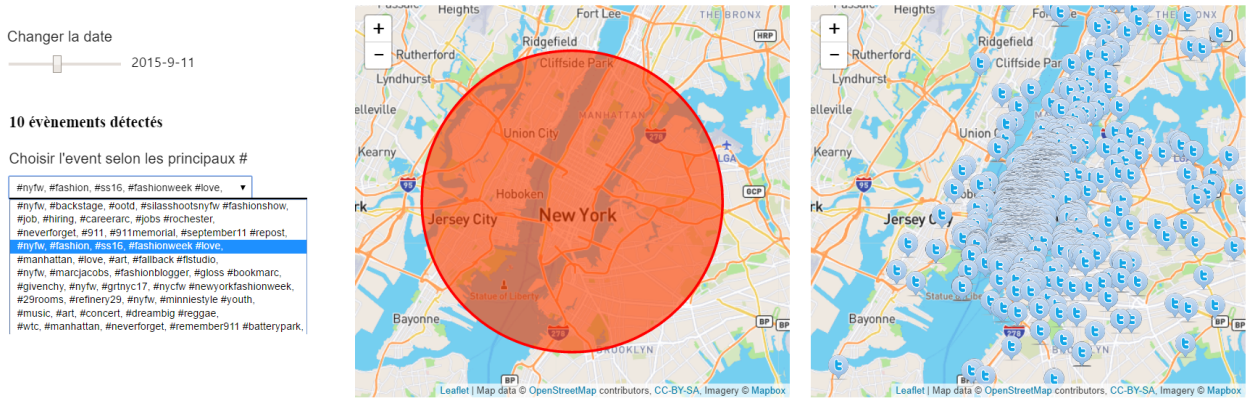


Fig. 5

Nous avons supprimé 787 comptes susceptibles d'être des robots sur les 103 jours de nos données.

On peut aussi constater, que cette fois-ci, les descriptions ont moins d'hashtags en commun. En effet, la blacklist permet d'empêcher la répétition de certains hashtags dans les descriptions.

Notre détection d'évènement est plus synthétique :
— 1 à propos d'un rassemblement au mémorial du 11 septembre
— 1 à propos d'une commémoration du 11 septembre au Battery Park
— 1 à propos d'un évènement à la maison de mode "29 Rooms"
— 1 à propos d'un défilé Givenchy
— 1 à propos de Marc Jacobs
— 2 à propos de la fashion week
— 1 à propos d'embauche et de jobs
— 1 à propos d'un concert de reggae
— 1 à propos de fête et du vendredi

Nous estimons avoir détecté une certaine quantité de comptes robots. Pour vérifier nos résultats, nous avons regardé les comptes associés aux utilisateurs suspects. Nous avons trouvé un certain nombre d'utilisateurs diffusant un flux important de contenu promotionnel pendant une certaine période. Nous avons aussi trouvé des comptes avec des "noms d'identifiant numérique" (i.e. dont le nom de l'user est 9098948492) et qui n'existent plus. Nous estimons également avoir éliminé 2 évènements potentiellement générés par des robots. En effet, nous avions 2 évènements à propos d'embauches et de jobs qui étaient détectés presque tous les jours et qui contenaient des hashtags suspects.

Nous avons également identifié beaucoup d'utilisateurs comme étant des robots alors qu'ils ne l'étaient pas. C'est pourquoi cette approche ne constitue qu'une simple tentative et n'est pas réellement efficace.

Voici le contenu de la blacklist à la fin de l'exécution sur les 103 jours :

— #brooklyn
— #newyorkcity
— #nyc
— #ny
— #newyork

7.3 Conclusion

Notre méthode est très simpliste (elle ne permet pas de faire sérieusement de la détection de robots) mais est satisfaisante dans notre approche d'exploration du domaine. En effet, nous arrivons à détecter des comptes automatisés et à filtrer des évèvenements qui ne sont pas des évèvenement physiques. Nous filtrons une grande quantité d'évèvenements par rapport à l'algorithme initial tout en gardant un nombre d'évènements raisonnable, i.e. permettant de décrire la plupart des évènements d'une journée.

8 La détection d'évènements avec DBSCAN

En parallèle des améliorations citées précédemment que nous avons développé, nous avons essayé d'utiliser Scikit-learn en Python. Nous avons donc réalisé un clustering DBSCAN. Le but était d'utiliser Scikit-learnest, principalement pour essayer plusieurs méthodes sensiblement différentes pour ensuite comparer les résultats.

8.1 Définition de la la métrique

DBSCAN a besoin de calculer des distances entre deux tweets pour remplir sa matrice de similarité. Nous ne voulions pas faire une simple distance euclidienne entre deux coordonnées, puisque c'est déjà ce que nous faisons dans l'autre algorithme. Nous avons donc integré les hashtags dans la mesure de distance entre deux tweets.

Soient les variables suivantes :

- a et b des tweets,
- α un coefficient positif,
- $malus$ un coefficient négatif,
- nb le nombre de hashtags en commun,
- δ_{ab} la distance euclidienne entre a et b.

On définit alors la distance entre a et b ainsi :

- Si nb est supérieur ou égal à 1 : $distance_{ab} = \frac{\delta_{ab}}{\alpha^{nb}}$
- Si nb est égal à 0 : $distance_{ab} = \frac{\delta_{ab}}{\alpha^{malus}}$

8.2 Les résultats

8.2.1 Évènements détéctés

Comme pour l'algorithme précédent, nous avons analysé avec DBScan la journée du *11 septembre 2015*. Voici donc les évèvenements que nous avons obtenus :

- *#ebaymotors*
- *#screenwriting*

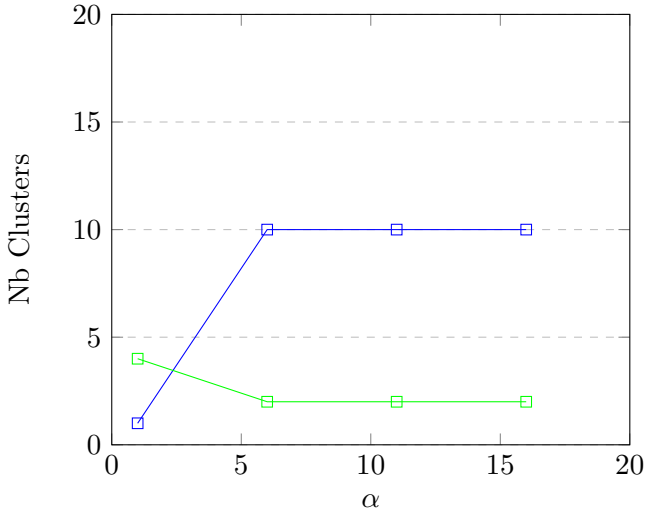
Comme nous n'avons pas d'options à activer ou désactiver avec notre algorihme DBScan, nous avons aussi détectés des évèvenements du *25 juillet 2015* afin d'avoir un comparatif :

- *#thingsbetterthannikkisreign*
- *#themarriott555canalstreetnola*

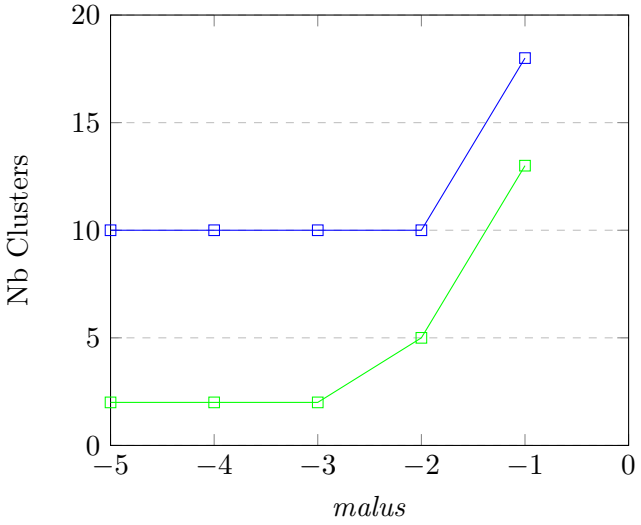
8.2.2 Influence des paramêtres

— Bleu : *11 septembre 2015*
— Vert : *25 juillet 2015*

Influence de α sur le nombre de clusters *malus* = -5



Influence de *malus* sur le nombre de clusters $\alpha = 15$



On peut observer que, pour α, il n'y a pas les mêmes résultats pour les deux courbes. Cela signifie que, selon les données, les coefficients influes différement sur le clustering.
Pour le *malus*, on retrouve un comportement similaire pour les deux courbes en faisant varier ce dernier. En effet, moins le *malus* est important, plus on augmente le nombre de clusters.

8.3 Les problèmes que nous avons rencontrés

Les évènements que nous avons détectés ne sont pas suffisaments importants. En effet, pour le *11 septembre 2015*, nous aurions du détecter la commémoration et ce n'est pas le cas.

Nous avons aussi un problème de passage à l'echelle dans le sens où, le traitement de 5 000 tweets nécessite 4 à 5 minutes d'attente (tests réalisés sur un ordinateur portable).

En rechanche, grâce aux coefficients, nous pouvons changer assez facilement les clusters et négliger plus ou moins les distances géographiques. Ainsi, nous pouvons obtenir des clusters éparpillés sur la carte.

Une idée serait de faire plusieurs culstering à la suite : l'un prenant fortement en compte la dimension géographique (pour détecter les concerts et/ou les bibliothèques par exemple) et l'autre pour détecter les évèvements nationaux.

9 Conclusion

Nous avons exploité un module existant permettant de détecter des événements à partir de tweets grâce aux informations spatio-temporelles et textuelles de ceux-ci. Nous avons exécuté l'implémentation de ce modèle sur des données de tweets du *21 juillet 2015* au *16 novembre 2015*.

Nous avons constaté la présence de nombreux évènements automatisés et de termes bruitant la détection d'évènements. Nous avons ainsi tenté d'éliminer les comptes automatisés dit "robots" pour atténuer ce bruit et faire ressortir davantage de vrais évènements.

Finalement, nous avons essayé de faire de la détection d'évènements de manière plus simple en utilisant l'algorithme de clustering DBScan. Nous avons ensuite comparé nos résultat pour montrer - entre autres - que la détection d'évènements reste limité si l'on ne se base que sur la densité pour la réaliser.

Acknowledgments

Nos remerciements vont à Marc Plantevit pour l'enseignement de son cours "Data Mining" à l'Université Claude Bernard Lyon I et son accompagnement durant ce projet.

Références

[1] How twitter bots fool you into thinking they are real people. 1

[2] X. Dong, M. D., Calabrese, and P. Frossard. Multiscale event detection in social media. *Data Min Knowl Disc*, June 2015. 2, 5, 6

[3] A. M. Nikan Chavoshi, Hossein Hamooni. Debot : Twitter bot detection via warped correlation, December 2016. :2016 :CHM