Université Claude Bernard Lyon I

Master Data Science

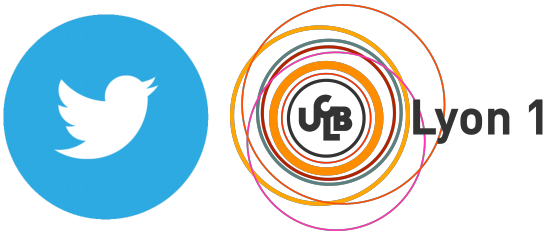
Projet Data Mining

Détection d'événements sur Twitter

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

Professeur :
Marc Plantevit

21 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 des plates-formes de ces réseaux contiennent, la plupart du temps, beaucoup d'informations. On peut traiter et analyser ces informations dans le but de faire de la détection d'évènement(s).
Néanmoins, une grande partie des données à traiter est bruitée, notamment par des comptes Twitter entretenus par des robots. Ces comptes postent des tweets en masse avec une fréquence de publication constante, ce qui nous empêchent de donner une bonne description dún événement ou alors on peut créer de faux évènements.
Voila pourquoi il est important de comprendre comment atténuer l'influence du bruit pour faire de la détection évènements. Notre objectif a été de détecter des évènements à partir de tweets en exploitant des informations spatio-temporelles et textuelles en essayant d'écarter les tweets de robots.

Table des matières

1 Introduction

Notre objectif a été 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 [1] et développé 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, soutenant des activités parrainées ou utilisant leur compte pour de la vente. Ces robots ne font qu'apporter une couche de bruits puisqu'il ne dévrivent pas d'évènement, faussant aisni notre clustering.

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 évèvenement que nous souhaitons détecter) un phénomène physique survenant en un point et pendant une durée bien déterminé. 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'évènement doit réussir à intégrer ces dispersions pour localiser dans les temps et l'espace les évènements et c'est ce qui la rend complexe.

2 L'approche globale

Pour notre premier algorithme nous appliquons, sur chaque ensemble journalier de tweets l'algorithme de détection d'évènement multi-échelle décrite ici section 5.

Pour notre second algorithme, nous appliquons le premier algorithme mais nous supprimons en amont les comptes robots que nous sommes susceptibles d'avoir dans nos données à l'aide d'un worlflow Knime basé sur le nombre de tweets postés par les utilisateurs puis par autre filtre basé sur la récurrence de post des tweets.

De plus, 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.

3 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 le 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.

4 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 les deux critères suivants. En effet, 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'évèvenement dans ses tweets et plus de "tweets bruits", c'est à dire des tweets nous donnant aucune information sur le déroulement d'un quelconque évèvenement (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 par exemple, nous pouvons nous permettre d'avoir des faux négatifs (utilisateurs considéré 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.

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

5 La détection d'évènements multi-ćhelle

La technique de détection d'évenements n'est autre que celle évoquée dans l'article [1].

L'algorithme calcule la matrice de similarité et construit les clusters.

MOI
MOI
MOI

5.1 Adaptation de l'algorithme de détection multi-échelle et intégration du filtrage des robots

On enlève ceux qui sont régi par une loi Géométrique (ici ou dans pretraitement des donnees)

6 Description des éléments utiles à la détection d'événements

6.1 Le tweet

Un tweet est décrit par :

— son identifiant
— le nom du Tweetos
— les hashtags
— la date et l'heure à laquelle le Tweetos a tweeté
— sa position au format latitude/longitude au moment du tweet
— La date

Nous appliquons donc notre algorithme de détection d'évenements sur chacun de nos sous-ensemble de tweets journaliers.

6.2 La matrice de similarité

La matrice de similarité est une matrice triangulaire supérieure de taille (nombre de tweets) * (nombre de tweets).
Rappeler ce qui a ete dit dans la partie 5
— Décrire construction
— reciter l'article en parlant rapidement de la compression de Haar

6.3 Le clustering

Le clustering se fait à partir de la matrice de similarité qui elle même a été obtenue en traitant un sous-ensemble de tweets journaliers.
Rappeler ce qui a ete dit dans la partie 5
— Redonner l'algo
— Citer le jar et expliquer rapidement ce qu'on maximise / minimise dans l'algorithme

6.4 L'événement

On décrit un événement 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 posté des tweets
— la liste des 20 hashtags les plus importants permettant de le décrire

Nous récupérons tous les clusters obtenus lors du clustering et ne gardons que les clusters pertinents. Un cluster est pertinent si le nombre de personnes ayant tweeté est assez important, s'il y a assez de tweets et si la proportion de tweets par personne n'est pas éxagérée. Ce cluster devient alors un événement.

7 Les options sur le clustering

7.1 L'élasticité

On suppose qu'un hashtag est fréquent pour un ensemble de tweets donné s'il apparait plus de 20 fois.

Ce nombre est statique, ce qui ne permettrait pas de détecter de petits événements dans une journée où il y a eu peu de tweets. A l'inverse, on détecte 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 d'adapter au nombre de tweets postés dans une même journée. Ainsi, on définit un hashtag fréquent s'il apparait dans plus de 20% des tweets.

Pour coder cette option nous avons définit un booléen, que l'on met à Vrai pour activer l'élasticité et à Faux pour ne pas l'activer.

Les valeurs 20 et 20% peuvent être modifées grâce aux variables globales définies au début du module d'éxécution Python.

7.2 La géolocalisation

7.3 La loi de Poisson géographique

8 La détection dévènement avec DBScan

9 Résultats

10 Conclusion

Les combinaisons intéressantes
Quand, comment

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