Université de Montréal

Détection d'évènements à partir de Twitter

par
Houssem Eddine Dridi

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

juin, 2014

© Houssem Eddine Dridi, 2014.

RÉSUMÉ

Nous proposons dans cette thèse un système permettant de déterminer, à partir des données envoyées sur les microblogs, les évènements qui stimulent l'intérêt des utilisateurs durant une période donnée et les dates saillantes de chaque évènement.

Étant donné son taux d'utilisation élevé et l'accessibilité de ses données, nous avons utilisé la plateforme Twitter comme source de nos données. Nous traitons dans ce travail les tweets portant sur la Tunisie dont la plupart sont écrits par des tunisiens.

La première tâche de notre système consistait à extraire automatiquement les tweets d'une façon continue durant 67 jours (de 8 février au 15 avril 2012).

Nous avons supposé qu'un évènement est représenté par plusieurs termes dont la fréquence augmente brusquement à un ou plusieurs moments durant la période analysée. Le manque des ressources nécessaires pour déterminer les termes (notamment les hashtags) portant sur un même sujet, nous a obligé à proposer des méthodes permettant de regrouper les termes similaires. Pour ce faire, nous avons eu recours à des méthodes phonétiques que nous avons adaptées au mode d'écriture utilisée par les tunisiens, ainsi que des méthodes statistiques. Pour déterminer la validité de nos méthodes, nous avons demandé à des experts, des locuteurs natifs du dialecte tunisien, d'évaluer les résultats retournés par nos méthodes. Ces groupes ont été utilisés pour déterminer le sujet de chaque tweet et/ou étendre les tweets par de nouveaux termes.

Enfin, pour sélectionner l'ensemble des évènements (*EV*), nous nous sommes basés sur trois critères : *fréquence*, *variation* et *Tf-Idf*. Les résultats que nous avons obtenus ont montré la robustesse de notre système.

Mots clés: Twitter, hashtags, évènement, TALN

ABSTRACT

In this thesis, we propose a method to highlight users' concerns from a set of Twitter messages. In particular, we focus on major events that stimulate the user's interest within a given period. Given its rate of use and accessibility of data, we used Twitter as a source of our data. In this work, we use tweets related to Tunisia, most of them being written by Tunisians.

The first task of our system was to continuously extract tweets during 67 days (from February 8th to April 15th, 2012).

We assumed that an event is represented by several terms whose frequency sharply increases one or more times during the analyzed period. Due to the lack of resources that allow determining the terms (including hashtags) referring to the same topic, we propose methods that help grouping similar terms. To do this, we used phonetic methods adapted to the way Tunisians write and statistical methods. To determine the validity of our methods, we asked the experts, who are native speakers of the Tunisian dialect, to evaluate the results returned by our methods. These clusters are used to determine the subject of each tweet and/or expand the tweets by new terms.

Finally, to select the set of events (*EV*), we relied on three criteria: *frequency*, *variation* and *Tf-Idf*. The results that we obtained show the robustness of our system.

Keywords:Twitter, Hashtags, event, NLP

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

INTRODUCTION

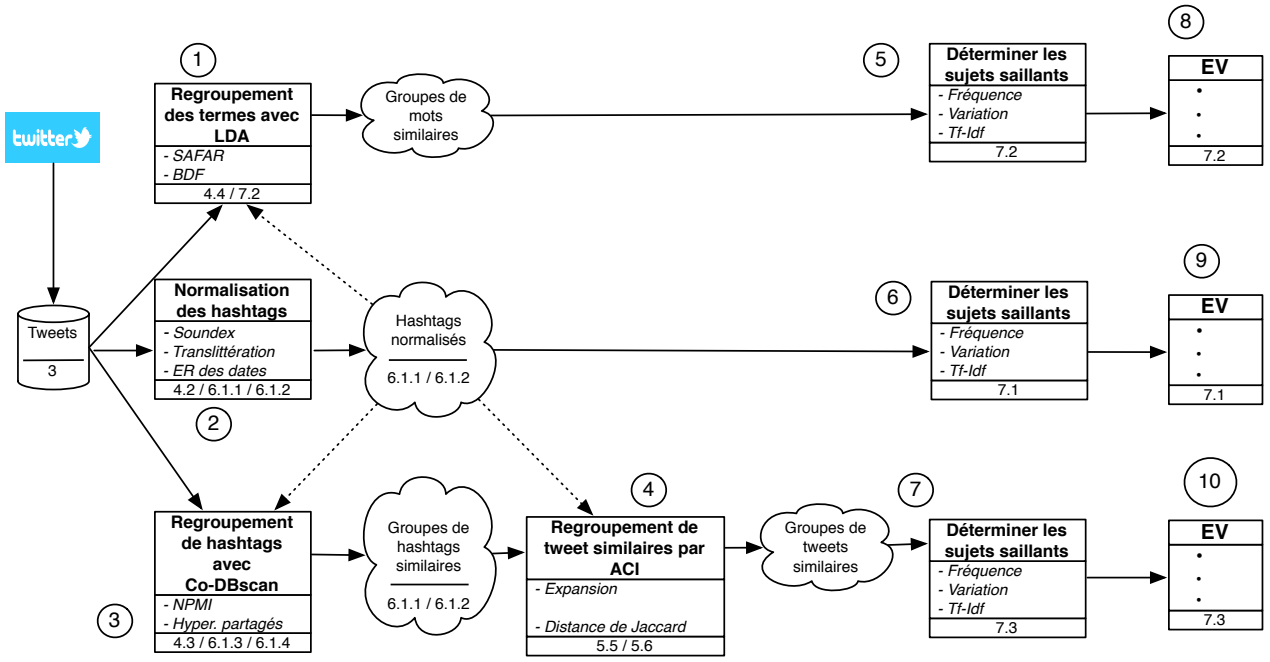
Plusieurs recherches ont montré que les données publiées par les internautes sur les sites de médias sociaux, notamment Twitter, reflètent presque en temps réel l'intérêt du public. Twitter est actuellement la plateforme de microblogage la plus populaire. Elle limite le nombre de caractères utilisés dans un message, appelé tweet, à 140 pouvant contenir également des hyperliens. Dans un tweet on peut étiqueter les sujets avec un mot *hashtag*, mot précédé par un dièse `#` (*hash* en anglais). En cliquant sur un hashtag, la liste des tweets ayant le même hashtag s'affiche. Voici un exemple de tweet : "`les manifestants se sont dispersés. #manifencours #ggi`". les sujets sont `manifencours` et `ggi`. En cliquant sur `#ggi` la liste des tweets ayant comme sujet `ggi` s'affiche.

Comme tous les médias sociaux, les utilisateurs inscrits sont en mesure d'établir des relations entre eux, un utilisateur pouvant s'abonner à d'autres ce qui lui permet de consulter leurs messages au moment de sa connexion.

Le contenu d'un tweet peut être un avis, une information et/ou un témoignage. La vaste communauté de Twitter, le taux d'utilisation incroyable avec plus de 500 millions de tweets par jour et la variété des intérêts des utilisateurs conduisent à une accumulation d'informations sur des évènements locaux (p.ex. *grève sur la hausse des frais de scolarité au Québec*) ou à l'échelle internationale (p.ex. *décès de Mickael Jackson*). Plusieurs études ont montré que Twitter est une source riche pour dégager les intentions ou même les émotions des utilisateurs. Contrairement aux autres plateformes de médias sociaux (e.g *Facebook*), le contenu de Twitter est public et accessible via des interfaces de programmation offertes par Twitter. Tous ces facteurs nous ont encouragé à utiliser Twitter pour réaliser notre objectif dans cette thèse, soit l'identification d'évènements qui stimulent l'intérêt des utilisateurs à un moment donné.

Dans cette thèse nous traitons des tweets qui portent sur la Tunisie, la plupart étant écrits par des Tunisiens. Nous avons été amené vers ce type de textes à cause de nos

Figure 1.1 – Vue globale sur notre système de détection d'évènements. Une boite représente un traitement ; un nuage représente un résultat obtenu après un traitement ; *EV* est l'ensemble d'évènements sélectionnés ; chaque traitement, identifié par un numéro encerclé, est détaillé par la suite.



compétences qui nous permettent de comprendre le français et l'arabe, particulièrement la façon d'écrire des Tunisiens qui comporte des abréviations, des fautes de grammaire et d'orthographe, des mots arabes écrits avec des alphabets français et chiffres et différentes langues dans le même tweet. Il sera ainsi plus facile de déterminer la précision de notre système étant donné notre surveillance de l'actualité en Tunisie.

Contrairement aux travaux qui s'intéressent à la détection des évènements à partir de documents longs et structurés, notre tâche semble plus compliquée pour plusieurs raisons : la taille des tweets qui ne dépasse pas 140 caractères, le type d'écriture employé dans les médias sociaux, notamment Twitter et les tweets sont écrits dans un dialecte arabe.

À notre connaissance, il n'y a pas eu de travaux qui ont essayé de détecter des évènements à partir de tweets écrits dans un dialecte arabe. Les travaux qui s'intéressent à la détection d'évènements à partir des médias traditionnels se basent sur des ressources

linguistiques préexistantes et des techniques de traitement automatique de la langue (TAL) afin d'atteindre leurs objectifs. Cependant, l'absence de ressources linguistiques pour le dialecte tunisien complique l'application des techniques habituelles de TAL.

Dans ce travail, nous considérons qu'un évènement est représenté par un ensemble de termes dont la fréquence augmente brusquement à un ou plusieurs moments durant la période analysée. Comme les hashtags permettent de donner une idée générale sur les sujets discutés dans un tweet, la majorité de nos méthodes se basent sur ces éléments afin de déterminer les sujets saillants.

La figure 1.1 montre les principales étapes de notre système. La tâche initiale de notre système consiste à extraire, d'une façon continue, les tweets à partir de Twitter en utilisant la *streaming API*. Le corpus utilisé dans ce travail est composé de 276 505 tweets collectés pendant 67 jours (de 8 février à 15 avril 2012). Aucune raison particulière ne nous a amené à recueillir les données durant cette période et de ce fait, il représente un échantillon pour nos expérimentations.

Comme nous l'avons dit précédemment, l'augmentation brusque de la fréquence d'un terme devrait indiquer la présence d'un sujet saillant (ou évènement). Pour cela, nous avons commencé par calculer les fréquences de différents termes (notamment les hashtags). Effectivement, nous avons constaté que les fréquences des termes référant à un évènement ont tendance à augmenter brusquement au moment du déroulement d'un évènement.

Si chaque terme référait à un sujet distinct, nous pourrions distinguer facilement les évènements. Cependant, un sujet est souvent représenté par plus d'un terme. Par exemple, la disparition de l'avion *Boeing 777* de *vol MH370*, affilié à *Malaysia airlines* le 8 mars 2014, a provoqué l'apparition de plusieurs hashtags référant à cet évènement : *#PrayForMH370*, *#MH370*, *#MH370Flight*, *#MalaysiaAirlines*... Il ne suffit pas donc de calculer la fréquence de chaque terme séparément, mais de les regrouper quand ils réfèrent au même sujet puis de calculer la fréquence de chaque *cluster*. En supposant que le sujet le plus important est le plus fréquent, l'addition des fréquences des termes représentant l'évènement peut donner plus d'importance à ce dernier. Certains termes (hashtags) pouvant avoir été créés avant ou après d'autres du même *cluster*, le regroupe-

ment des termes peut servir à déterminer la durée de vie exacte de l'évènement.

Notre première tâche dans le processus de détection des sujets qui ont stimulé l'intérêt d'utilisateurs est de regrouper les termes référant à un même sujet. Pour ce faire, nous avons développé trois méthodes :

Normalisation des hashtags ② [1] Les utilisateurs des médias sociaux commettent souvent des fautes d'orthographes. Dans notre cas, ce phénomène est amplifié par le fait que les tunisiens ont tendance à écrire des mots arabes en utilisant des alphabets latins et des chiffres chaque utilisateur translittérant le mot de sa façon. Pour normaliser ces hashtags, nous avons eu recours à des algorithmes phonétiques en adoptant *Soundex* pour supporter le dialecte tunisien, afin d'attribuer le même code *Soundex* (CS) aux hashtags avec une prononciation semblable. Nous proposons, également, un algorithme de translittération qui permet de coder les hashtags écrits en alphabets arabes avec le même CS que leurs semblables écrits en latin.

Regroupement des mots similaires ① :

Nous avons utilisé *LDA* (*Latent Dirichlet Allocation*), une technique statistique qui sert à détecter les sujets abstraits à partir d'une collection de documents. L'idée de base de *LDA* est que chaque document peut être représenté comme un mélange de sujets latents, où un sujet est lui-même représenté comme une distribution de mots ayant tendance à cooccurrer. Les mots fortement liés à un sujet ont des valeurs de probabilité plus grandes. Dans notre travail, nous avons profité de cette distribution pour obtenir les termes qui portent sur un même sujet.

Afin d'améliorer les résultats obtenus par *LDA*, nous avons regroupé dans un seul document les tweets qui partagent les mêmes CS dans le but de donner plus de chance à des mots d'apparaitre ensemble.

Regroupement de hashtags avec *CoDBscan* ③ :

Dans cette méthode, nous proposons des variantes d'un algorithme de *regroupement* (basées sur l'algorithme *DBscan*), qui tient compte de la distance entre les éléments, pour regrouper les hashtags similaires. Pour déterminer la distance entre

les hashtags, nous nous sommes basés sur leurs cooccurrence et nous avons profité des hyperliens trouvés dans les tweets. Dans cette tâche, nous avons profité également des résultats obtenus par la tâche de normalisation pour améliorer le regroupement en changeant les hashtags par leur CS.

Après ces regroupements, nous avons utilisé les *clusters* de hashtags et de mots obtenus, pour étiqueter chaque tweet avec le sujet approprié (⑤ et ⑥). Nous supposons qu'un tweet porte sur un sujet si au moins un de ses termes est présent dans le tweet en question. Cette tâche a permis de calculer la fréquence quotidienne de chaque sujet (*cluster*).

D'un autre côté, nous avons utilisé un algorithme de *regroupement* incrémental ⑦ pour regrouper les tweets similaires en comparant les hashtags trouvés dans les tweets avec la mesure de *Jaccard*. Dans ce processus, nous avons étendu les tweets par d'autres hashtags similaires, en utilisant des *clusters* de hashtags, obtenus par des variantes de *CoDBscan*, afin d'améliorer le *regroupement*. La fréquence quotidienne d'un *cluster* correspond au nombre de ses tweets créés quotidiennement.

La prochaine étape consiste à identifier, parmi les sujets obtenus précédemment, l'ensemble de sujets référant à des évènements, *EV* (⑧, ⑨ et ⑩). Pour ce faire, nous avons évalué les sujets selon trois critères : *fréquence*, *variation* (ou *écart-type*) et *Tf-Idf*. Selon le critère utilisé, les valeurs du critère, des sujets appartenant à *EV*, correspondent à des pics. Nous avons évalué les résultats avec l'aide des personnes familiarisées avec les évènements déroulés en Tunisie. L'évaluation a été faite via une application web que nous avons créée pour permettre à un expert de déterminer parmi *EV* les sujets correspondant à des évènements.

1.1 Organisation de la thèse

Au chapitre 2, nous présentons les différents types de médias sociaux, la plateforme Twitter et les travaux portant sur l'analyse de données provenant de Twitter.

Au chapitre 3, nous présentons le corpus utilisé dans ce travail ainsi que les APIs fournies par Twitter pour extraire les données, notamment la *STREAMING API*.

Au chapitre 4, nous présentons les particularités des tweets écrits par des tunisiens et les algorithmes proposés pour normaliser et regrouper les termes sémantiquement similaires. Le chapitre 5 présente les méthodes développées afin d'identifier les différents sujets discutés dans le corpus.

Au chapitre 6, nous présentons les expérimentations et les résultats obtenus par les méthodes de normalisation et de regroupement de termes définies au chapitre 4. Le chapitre 7 présente les expérimentations pour déterminer EV.

Nous terminons par une conclusion et des extensions pour des travaux futurs.

CHAPITRE 2

MÉDIAS SOCIAUX ET RECHERCHE DANS TWITTER

Dans ce chapitre, nous définissons le phénomène des média sociaux et les caractéristiques de ses différents types. Nous décrivons plus en détail notre source de données dans ce travail, Twitter.

Dans la deuxième partie de ce chapitre, nous faisons un survol sur les différents types de travaux qui utilisent Twitter comme source de données.

2.1 Médias sociaux

Les médias sociaux désignent des données envoyées par leurs utilisateurs, le terme *sociaux* référant au fait que les utilisateurs partagent ces données. Contrairement aux médias traditionnels (télévision, radio, journaux ...) qui fournissent des informations à l'ensemble de la population (*one-to-many*), les médias sociaux ont transformé cette hiérarchie à *many-to-many* où chacun peut devenir un producteur. Un même média peut être accédé via plusieurs moyens des communications : directement à travers un site web, un *sms*, un courriel, une application mobile, etc.

Nous distinguons plusieurs types de médias sociaux :

Réseaux sociaux : ce service permet aux utilisateurs de partager leurs informations, photos, vidéos avec leurs amis. Les informations des utilisateurs dans ces sites (*Facebook*, *Myspace* ...) ne sont pas publiques, seuls les amis de l'abonné peuvent les consulter.

Création collaborative (*Wikis*) : ce sont des plateformes qui permettent aux gens de contribuer ou de modifier du contenu textuel. L'idée derrière ce service est que n'importe qui peut écrire, et que quelqu'un d'autre peut venir plus tard et corriger les erreurs. *wikipédia* est le *wiki* le plus célèbre.

Monde virtuel : ce service offre aux utilisateurs des environnements graphiques et des

outils pour les utilisateurs d'agir dans ces environnements de différentes manières. Les plateformes (*Second Life*, *World of Warcraft* ...) qui offrent ce service, permettent aux utilisateurs de construire de nouveaux espaces, de créer des objets, et d'utiliser des langages de programmation puissants pour automatiser leur comportement.

Partage de contenu multimédia : ces sites sont conçus pour permettre aux utilisateurs un type de donnée particulier : vidéos (*Youtube*), photos (*Flickr*), musique (*Last.fm*), liens utiles et intéressants (*del.icio.us*)...

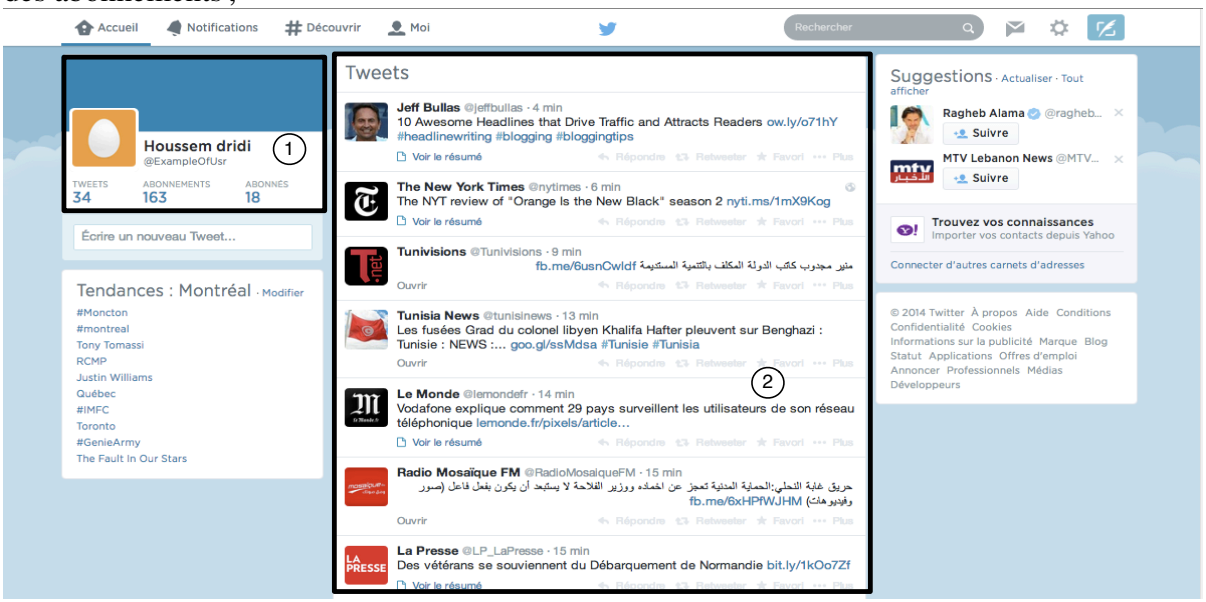
Microblogs : ces plateformes permettent aux utilisateurs de publier des messages courts (entre 140 et 200 caractères au maximum) mais qui peuvent référer à autres documents, possiblement dans un autre média.

Au début, l'idée était de permettre aux internautes d'indiquer aux autres ce qu'ils sont en train de faire. Cependant, les choses se sont développées vite et les internautes ont profité de ce service pour exprimer leurs opinions sur différents sujets, diffuser des informations et faire des discussions. Contrairement aux blogs la plupart des internautes sont actifs, car ils n'ont plus besoin de rédiger de longs textes.

Le style d'écriture, employé dans les microblogs, est parfois incompréhensible par les non-initiés ou par les gens qui ne font pas partie de la conversation. Les utilisateurs commettent fréquemment des fautes d'orthographe et de grammaire, utilisent des abréviations, étirent des mots, et utilisent d'onomatopées (rire = *ha ha*) et des néographies (qui =*ki*).

Plusieurs raisons ont encouragé les internautes à s'inscrire à des *microblogs* : facilité d'utilisation, contact avec les amis, information en temps réel ou échange d'idées avec les autres.

Figure 2.1 – Capture d'écran de la page d'accueil Twitter. ① : information de l'utilisateur (nombre de tweets envoyés, nombre d'abonnement et le nombre d'abonnés) ;② : tweets des abonnements ;



2.2 Twitter

2.2.1 Présentation

Twitter est actuellement la plate-forme de microbloggage la plus populaire. Son premier slogan était `Que faites-vous?` néanmoins l'utilisation a pris une autre piste où les utilisateurs échangent des avis et des informations, le slogan est devenu `Quoi de neuf ?`. Plusieurs célébrités utilisent Twitter, on y trouve même des chefs d'État.

Le site, créé en 2006, a été conçu à l'origine pour téléphones mobiles pour permettre à ses utilisateurs d'envoyer des messages, appelé tweets, à l'aide de service SMS (*Short Message Service*). Comme la taille d'un sms ne dépasse pas 160 caractères, Twitter a limité la taille d'un tweet à 140 caractères, 20 caractères étant réservés au nom de l'expéditeur. Selon les derniers chiffres[1], Twitter a plus de 600 millions utilisateurs inscrits et reçoit plus de 500 millions de tweets par jour. La figure 2.1 montre une capture

[1] `https://about.twitter.com/company`, `http://www.statisticbrain.com/twitter-statistics/`

d'écran de la page d'accueil Twitter.

Contrairement aux autres services de médias sociaux, seulement 22% (Kwak et al. (2010)) des relations entre les utilisateurs sont symétriques. Une relation est symétrique si un utilisateur A suit un utilisateur B et si B suit A ; cela implique que chacun peut consulter (sur son tableau de bord) les messages de l'autre. Par contre une relation asymétrique est le fait qu'un utilisateur A suit un utilisateur B sans que B suive A, cela implique que A peut consulter (sur son tableau de bord) les messages de B. Par contre, B ne peut pas consulter ceux de A. Donc, les amis intimes sont souvent en relations symétrique mais les politiciens ou les stars sont en relation asymétrique. Par exemple, environ 250 000 utilisateurs suivent *Celine Dion*, mais elle ne suit que 2 personnes.

2.2.2 Fonctionnalités et caractéristiques

Les tweets sont, par défaut, publics. Cependant, Twitter offre à ses membres la possibilité de limiter la liste des utilisateurs autorisés à voir leurs tweets. Une fois connecté, un fil de tweets écrits par les utilisateurs qu'on suit s'affiche.

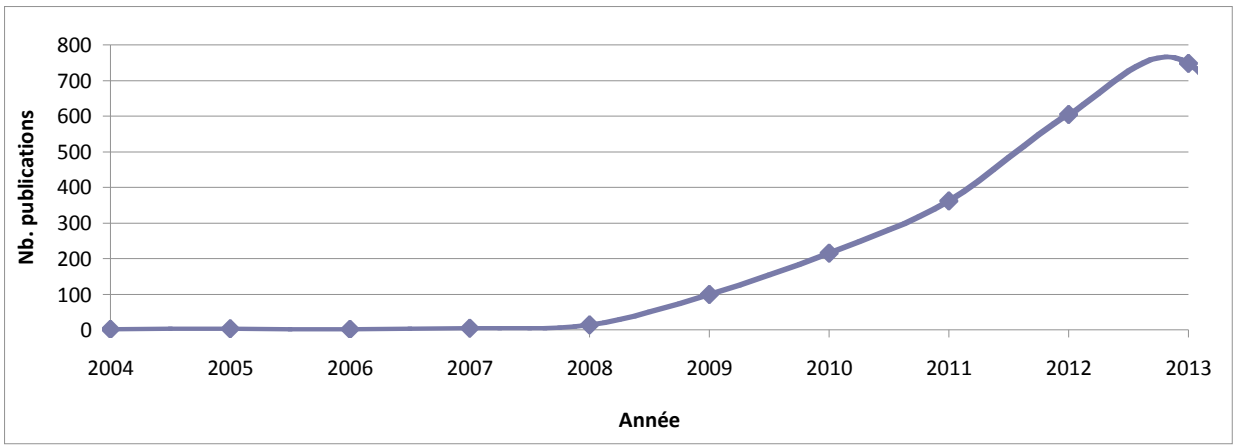
Conventions d'écriture :

- Le nom d'un utilisateur est un identifiant précédé par `@`. Exemple : `@ExampleOfUsr`.
- Dans un tweet on peut étiqueter les sujets dont on parle. Un sujet est précédé par un dièse `#` pour former un hashtag, *mot-clic* en français. Pour le tweet : `les manifestants se sont dispersés. #manifencours #ggi`', les sujets sont `manifencours` et `ggi`. Le but des hashtags est de mettre en surbrillance ou de regrouper les tweets du même sujet pour que les autres les suivent. En cliquant sur `#ggi` la liste des tweets qui contiennent `#ggi` s'affiche.
- Une réponse à un tweet de l'utilisateur `X` commence toujours par `@X`.
- Un retweet (réémission d'un tweet) commence par `RT @Y` où `Y` est l'expéditeur du tweet original.
- Pour mentionner un utilisateur dans un tweet il suffit de taper son nom précédé par `@`.

Certains URLs qu'on veut publier sur Twitter sont trop longues et dépassent la taille permise pour un tweet. Pour cela, Twitter utilise un service de réduction d'URL qui rend la page accessible par l'intermédiaire d'une très courte URL. Il existe plusieurs services de réduction tels que *TinyURL*[2], *bitly*[3] et *t.co* (créé par Twitter et qui est utilisé seulement pour les URLs insérées dans les tweets). Par Exemple l'URL `http://www.iro.umontreal.ca/rubrique.php3?id_rubrique=13` devenue `http://t.co/NrUGjAtx` par le service `t.co`. Un service génère toujours la même URL pour la même entrée.

2.3 Recherches sur Twitter

Figure 2.2 – Nombre de publications indexés par la *base de données bibliographique interdisciplinaire*, *Web science* contenant le terme *'Twitter'* entre 2004 et 2013



Étant donné le nombre d'inscrits dans Twitter, son taux d'utilisation élevé et le fait que les tweets soient publics, plusieurs études récentes l'ont choisi comme source de données.

Nous avons effetué une recherche sur les publications contenant le mot *'Twitter'* dans la base de données bibliographique interdisciplinaire, *Web science*. La figure 2.2 [4]

[2] `tinyurl.com`
[3] `bit.ly`
[4] `http://wokinfo.com/`

présente le nombre annuel de publications contenant le terme 'Twitter'. Les publications effectuées avant 2007 portent principalement sur des études de 'vocalisation' et non pas sur la plateforme Twitter. Les résultats obtenus montrent que la plateforme Twitter est devenue de plus en plus une source intéressante pour les chercheurs vu l'importance de l'information qui peut être extraite à partir de son contenu.

2.3.1 Classification de sentiments

Pour avoir une idée concernant un sujet (produit, service,...) la première question qu'on se pose est : *est-il bon ?*. Pour répondre à cette question à partir des commentaires publiés par les internautes, il est important de les classifier selon le type global de l'opinion exprimée : favorable, défavorable ou neutre.

On parle souvent de polarité de texte dans ces cas : polarité positive correspondant à une opinion favorable et polarité négative lorsque défavorable.

L'objectif de la classification des sentiments est de donner rapidement une impression du ton d'un texte. Celle-ci ressemble à la détermination des sujets dans un texte qui permet d'étiqueter un texte avec des thèmes tel que (*sport*, *politique*, *éducation*...). Pour chaque classe C, on trouve des termes considérés comme des indices pour C. Par exemple, les termes `loi`, `gouvernement`, `président`, `justice`... sont des indicatifs du thème politique. Dans la classification des sentiments les termes qui indiquent un sentiment (négatif ou positif) sont importants : *like*, *dislike*, *hate*,*good*, *excellent*...

Dans Twitter, les utilisateurs publient souvent leurs points de vue sur différents sujets : politique, sports, économie et ainsi de suite. Cela a suscité l'intérêt de la communauté de recherche qui s'intéresse au traitement de la langue naturelle (TALN), qui a commencé à étudier les messages publiés sur Twitter.

Bermingham et Smeaton (2010) ont constaté que la brièveté des messages Twitter donne une classification de sentiment plus fiable.

Go et al. (2009) ont développé une application qui est disponible en ligne intitulée : *twitter sentiment*[5]. Cette application permet de déterminer les polarités des mes-

[5] `http://twittersentiment.appspot.com/`

sages publiés sur Twitter et qui répondent à la requête envoyée par l'utilisateur.

Dans ce travail, les auteurs ont utilisé des techniques classiques de classification pour déterminer la polarité d'un tweet. Ils se sont basés sur des méthodes basées sur l'apprentissage supervisé : *méthode bayésienne naïve*, *méthode d'entropie maximale*, *les machines à vecteurs de support*. Les méthodes par apprentissage supervisé utilisent un ensemble d'apprentissage annoté a priori. Pour le construire, les auteurs se sont basés sur les émoticônes pour déterminer la polarité d'un message. Les émoticônes, appelés aussi *smilies*, sont utilisés souvent dans les messages envoyés sur les microblogs, ils sont employés pour exprimer des émotions (heureux, triste, nerveux...).

Barbosa et Feng (2010) : Deux étapes ont été utilisées pour classifier les tweets : 1) classification de subjectivité : déterminer si le tweet est subjectif (porte une opinion ou un sentiment) ou non. 2) classification de polarité : déterminer, parmi les tweets subjectifs, la polarité de chacun.

Les méthodes utilisées afin de réaliser ces tâches, sont basées également sur l'apprentissage supervisé. Pour collecter les données d'apprentissage, les auteurs ont eu recours à 3 applications qui analysent les sentiments en utilisant *Twitter* :*Twendz*[6], *TweetFeel*[7], *Twitter Sentiment* présenté précédemment. Ces applications retournent des tweets, qui contiennent le mot clé saisi par l'utilisateur, avec leurs classes. Pour recueillir des tweets génériques (qui portent sur différents sujets), les auteurs ont utilisé le mot clé `of`, un mot très fréquent en anglais qui permet de récupérer beaucoup de tweets.

Jiang et al. (2011) : Comme dans le travail précédent, les auteurs ont utilisé deux classifieurs : un classifieur de subjectivité et un classifieur de polarité pour les tweets classifiés comme subjectifs.

Les tweets ne contiennent pas toujours assez d'informations ou sont souvent ambigus, p.e. il est difficile de déterminer la classe du tweet `First game: Lakers!`,

[6] `http://twendz.waggeneredstrom.com/`
[7] `http://www.tweetfeel.com/`

qui contient seulement trois mots. En plus, le tweet `People everywhere love Windows & vista. Bill Gates` qui n'exprime pas aucun sentiment sur `Bill Gates`, mais il peut être classifié comme positif par les méthodes de Go et al. (2009) et Barbosa et Feng (2010) vu la présence du terme `love`. Les auteurs ont constaté qu'il faut sélectionner seulement les termes qui portent sur le sujet cible et qu'il ne suffit pas de considérer que le tweet à classifier, il faut aussi tenir compte d'autres éléments qui sont en relation avec le tweet. Pour résoudre ce problème les auteurs ont décidé de

- Sélectionner les syntagmes nominaux incluant le sujet cible : si le sujet est `Microsoft`, `Microsoft technology` est un syntagme nominal.
- Sélectionner les pronoms et les groupes nominaux qui réfèrent au sujet : dans le tweet `Oh, Jon Stewart. How I love you so.`, `You` sera sélectionné parce qu'il réfère à `Jon Stewart` (le sujet).
- Sélectionner les noms qui sont fortement associés au sujet, c'est-à-dire qui cooccurrent souvent avec le sujet.
- Sélectionner les *traits*, qui ont une relation avec le sujet ou l'un de ses attributs (les termes sélectionnés auparavant), en se basant sur un ensemble de règles.

Les auteurs n'ont pas seulement considéré le tweet à classifier, ils ont aussi étudié les polarités des tweets qui ont une relation avec lui. Trois types de relation ont été étudiés : 1) les retweets 2) les tweets envoyés par le même utilisateur et qui portent sur le sujet et les tweets *répondant à* ou *répondus par* le tweet à classifier.

Cet article a pris en considération la taille (généralement courte) et l'ambiguïté de texte. Les auteurs ont montré que ces facteurs peuvent influencer sur la classification d'un tweet. Ils ont prouvé que la relation entre les tweets peut résoudre ces problèmes et améliorer l'exactitude de la classification.

2.3.2 Prédiction des résultats

Ces travaux traitent de la prédiction des résultats à partir des messages publiés dans Twitter :

Lampos et Cristianini (2010) ont utilisé Twitter pour mesurer la prévalence de la maladie H1N1 pour la population de la Grande-Bretagne. Les auteurs ont collecté des tweets chaque jour pendant 24 semaines : de 22/06/2009 jusqu'à 06/12/2009 qui proviennent de 5 régions différentes dans la Grande-Bretagne. Par la suite, ils ont supprimé les mots vides (*stop words*) et ont appliqué un algorithme de *stemming*.

La méthode proposée dans ce travail consiste à rechercher dans les tweets les symptômes liés à la grippe H1N1 puis à retourner un score intitulé *Flu_score*. Les auteurs ont utilisé un ensemble $\mathscr{M}$ qui contient 41 indices (temperature, headache, sore throat...) pouvant exprimer la grippe. Le *Flu_score* d'un tweet t est calculé comme suit :

$$s(t) = \frac{\sum_i m_i(t)}{k}$$

D'où m_i est $i^{ème}$ indice et k est le nombre des indices, m_i(t) est égale à 1 si m_i apparaît dans le tweet et 0 sinon. Le *Flu_score* des tweets d'un jour j est calculé comme suit :

$$f_j = \frac{\sum_q s(t_q)}{n}$$

D'où t_q est le $q^{ème}$ tweets au jour j et n le nombre de tweets dans j.
Afin d'évaluer les résultats obtenus, les auteurs ont décidé de les comparer avec les données de l'*agence de protection de la santé* (*APS*).

Les auteurs ont essayé d'améliorer les coefficients de corrélation en attribuant un poids à chaque indice. le calcul de *Flu_score* est défini comme suit :

$$s_w(t) = \frac{\sum_i w_i \times m_i(t)}{k}$$

$$f_{w,j} = \frac{\sum_q s_w(t_q)}{n}$$

D'où w_i est le poids de l'indice m_i.

Dans le but d'apprendre les poids de chaque indice, ils ont appliqué la méthode des moindres carrés entre les *Flu_scores* (non pondérés) obtenus et les données de l'APS. Ils ont utilisé comme ensemble d'apprentissage les données qui correspondent à une région, puis ils ont évalué les poids inférés sur les données des autres régions.

Ils ont également essayé de prédire le coefficient de corrélation en utilisant les données qui correspondent à toutes les régions. Les données entre les semaines 28 et 41 sont les données de test et le reste est utilisé pour apprendre les poids.

Les auteurs ont tenté d'extraire automatiquement les indices. Ils ont collecté un ensemble de candidats obtenu à partir des articles dans le web liés à la grippe.

L'ensemble contient 1560 candidats. Par la suite, ils ont appliqué le même principe utilisé précédemment (apprendre les poids). Le nombre de candidats retenus (leurs poids >0) est 73. Une corrélation qui dépasse 95% a été obtenue.

OConnor et al. (2010) ont comparé les résultats provenant de l'opinion publique mesurée à partir des sondages et celle mesurée à partir des tweets. Les élections américaines en 2008 étaient parmi les sujets analysés. Les auteurs ont proposé un score qui reflète l'opinion publique par jour. Le score est le rapport entre les tweets positifs (favorables) et les tweets négatifs (défavorables) qui portent sur le sujet en question. La polarité d'un tweet est considérée positive s'il contient un terme positif (e.g *nice*) et négative s'il contient un terme négatif (e.g *bad*). L'orientation (positive, négative) d'un terme est déterminée par la ressource linguistique *OpinionFinder*. Les résultats obtenus ont montré qu'il y a une forte corrélation entre des données réelles provenant des sondages et l'opinion publique mesurée à partir des tweets.

Bollen et al. (2011) ont utilisé des tweets pour étudier la corrélation de l'évolution de

l'humeur publique et les marchés boursiers. Les auteurs ont analysé le contenu des tweets publiés quotidiennement pour prédire l'humeur publique. Pour déterminer l'humeur exprimée dans un tweet, deux outils ont été utilisés : *OpinionFinder* (utilisé par OConnor et al. (2010)) et *POMS* (*Profile of Mood States*) qui prédit l'humeur parmi 6 états (*Calm*, *Alert*, *Sure*, *Vital*, *Kind*, and *Happy*). Seuls les tweets contennant certaines expressions (*i feel*,*i am feeling*, *i'm feeling*, *i dont feel*, *I'm*, *Im*, *I am* et *makes me*) ont été considérés. Les résultats obtenus ont montré qu'il y a une forte corrélation (86.7%) avec les valeurs de *Dow Jones Industrial Average* (*DJIA*).

Pour les sentiments et prédictions des résultats : la plupart des travaux qui cherchent à déterminer le sentiment et prédire des résultats à partir des tweets se basent sur les termes trouvés dans les tweets pour atteindre leurs objectifs. Les sentiments d'un tweet sont déterminés principalement par certains termes qui représentent des indices pour connaitre la polarité des tweets. Un ensemble de termes (construit, souvent, manuellement) est utilisé pour détecter les tweets discutant le sujet en question.

Certains travaux se basent sur des ressources préexistantes et des règles grammaticales et ignorent les particularités d'écriture employées dans Twitter telles que les fautes d'orthographe et les abréviations…, ce qui n'est pas évident.

2.3.3 Détection des événements

Les microblogs constituent le meilleur moyen pour diffuser des informations et discuter des événements et y donner des avis. Kwak et al. (2010) ont constaté que les utilisateurs de Twitter diffusent parfois des nouvelles avant les médias traditionnels (télévision, radio). Sutton et al. (2008), dans une étude sur les incendies des forêts en Californie en 2007, ont montré que Twitter a représenté une source d'information importante pour les citoyens et ont constaté que les médias traditionnels se sont tournés vers Twitter pour obtenir des informations.

Un événement est défini comme étant *quelque chose* qui arrive à un moment donné. Les recherches qui s'intéressent à la détection des événements à partir des médias tradi-

tionnels comme les journaux se sont basées sur des techniques de traitement automatique de la langue naturelle (Makkonen et al., 2004), (Zhang et al., 2007) telle que l'extraction d'entités nommées. Toutefois, l'application de ces techniques sur les tweets est plus difficile compte tenu la petite quantité d'information contenue dans un tweet.

Plusieurs recherches ont montré que le contenu de ces outils (notamment Twitter) reflète bien l'intérêt et les préoccupations des utilisateurs en temps réel :

Jianshu et Bu-Sung (2011) ont proposé une méthode basée sur la fréquence quotidienne des termes dans le corpus. La fréquence de chaque terme est représentée sous forme d'un signal. Ces signaux obtenus sont analysés par une technique de traitement de signal : les ondelettes pour déterminer quand et comment la fréquence du signal change dans le temps (Kaiser, 2011). Les auteurs ont considéré que les termes avec de signaux similaires, représentent le même événement. La similarité entre deux signaux est effectuée par la mesure *cross correlation*.

Ozdikis et al. (2012a, b) ont effectué une *expansion* sémantique des termes présentés dans les tweets basée sur la cooccurrence des termes afin de regrouper les tweets selon leur similarité. Chaque tweet est représenté sous forme d'un vecteur de termes. La similarité entre deux tweets est calculée par la mesure de *cosinus de similarité*. Les auteurs ont considéré que chaque *cluster* de tweets représente un événement.

Ce travail a montré que les hashtags sont de bons facteurs pour détecter les événements à partir des tweets. En outre, l'expansion sémantique a apporté une amélioration importante. Elle permet d'augmenter le nombre de tweets qui portent sur un même événement ce qui permet de lui donner plus d'importance et d'élargir la durée d'un événement. L'élargissement de l'ensemble de hashtags, qui représente un événement, conduit à hâter sa détection.

Becker et al. (2011) ont proposé une approche pour identifier, parmi tous les tweets, ceux qui décrivent des événements. Plusieurs tweets sont tout simplement une conversation entre des amis ou des opinions. Les auteurs ont implémenté un algo-

rithme de *regroupement* en ligne, qui affecte chaque nouveau tweet à un *cluster* adéquat. Par la suite, ils ont appliqué un algorithme de classification pour déterminer si un *cluster* (selon le contenu de ses tweets asoociés) porte sur un évènement ou non.

Comme les autres tâches, la détection d'évènements s'appuie sur les termes pour déterminer les sujets saillants survenus durant une période donnée. Toutefois, pour la détection d'évènements non connus à priori, il est difficile d'utiliser un ensemble prédéfini pour déterminer des évènements qu'on ne connait pas.

Les travaux qui s'intéressent à cette tâche, cherchent à regrouper les tweets similaires puis à déterminer parmi les *clusters* obtenus ceux qui réfèrent à des évènements. La similarité entre les tweets se base souvent sur leurs contenus textuels.

Dans ce travail, nous avons envisagé la même approche qui consiste à regrouper les tweets similaires. Mais, nous avons suggéré une autre méthode plus simple qui sert à compter la fréquence des *clusters* de termes (voir figure 1.1). Malgré sa simplicité, cette méthode a permis de refléter les sujets saillants au cours d'une période donnée. Notre système détecte non seulement les évènements, mais il permet de détecter les dates saillantes de tous les évènements détectés, ce qui aide à mieux décrire les évènements.

2.4 Conclusion

Dans ce chapitre, nous avons présenté différents types de médias sociaux et des travaux utilisant Twitter comme source de données. La plupart de ces travaux se basent sur les termes trouvés dans les tweets pour atteindre leurs objectifs.

Au chapitre suivant, nous présenterons les APIs proposées par Twitter afin d'extraire les tweets, la démarche envisagée pour recueillir notre corpus et ses caractéristiques.

CHAPITRE 3

EXTRACTION DES DONNÉES

Contrairement aux autres sites de médias sociaux qui appliquent des restrictions sur leurs données, Twitter partage ses données avec des entreprises et des chercheurs. L'accessibilité informatique des tweets a permis de faire de Twitter une source intéressante pour les chercheurs qui s'intéressent à l'analyse des données générées par les utilisateurs. Cela est dû aux interfaces de programmation (*APIs*) mises à la disposition. Plusieurs librairies de programmation permettent l'accès à ces *APIs*. Pour collecter nos corpus, nous avons implémenté un programme *Java* utilisant la librairie *Twitter4j* [1].

Nous présentons d'abord les APIs de Twitter et les caractéristiques de nos corpus, principalement des tweets écrits en dialecte tunisien.

3.1 Twitter API

La collection des données générées par des utilisateurs est la tâche la plus importante dans le processus d'analyse des données. Une première méthode de collection est manuelle, elle consiste à copier manuellement des contenus, généralement textuels, à partir des forums et/ou des médias sociaux, mais il est beaucoup plus pratique d'implémenter des scripts qui permettent de récupérer automatiquement les pages avec des données. Ces méthodes nécessitent toutefois un nettoyage. Généralement, les données manquent certaines informations, telles que le profil de l'utilisateur (identifiant, emplacement, nombre des messages écrits, nombre des amis. . .), le dispositif de communication et l'application utilisés pour envoyer le message . . .

Actuellement, à l'aide des APIs Twitter nous pouvons récupérer facilement des tweets, qui répondent à un ensemble de paramètres (e.g. mots-clés). Ces APIs renvoient des données bien structurées pour faciliter leur analyse et l'accès à l'information désirée. Les tweets retournés contiennent une variété d'informations : le texte, hashtags, nombre

[1] `http://twitter4j.org/en/index.html`

de fois retweetés, les informations de l'utilisateur. Afin d'être autorisé à accéder aux APIs Twitter, il faut obtenir un *consumer key* et *consumer secret key*. Cela ne peut être possible qu'après s'être inscrit sur Twitter et avoir créé une application. Pour créer une application, il suffit de remplir un formulaire (nom de l'application, objectif ...). Trois types d'APIs sont proposés par Twitter : *REST API*, *SEARCH API* et *STREAMING API*.

3.1.1 REST API

La *Representational State Transfer* (*REST*) *API* offre des méthodes, d'écriture et lecture, qui permettent d'accéder à la base de données principale de Twitter qui contient toutes les données. Cela inclut l'extraction des 20 tweets les plus récents envoyés (appelés *timeline*) sur Twitter, des informations sur un utilisateur (ses *timeline*, ses abonnés, ses abonnements, les tweets dans lesquels il est mentionné ...) en indiquant son identifiant, l'envoi d'un tweet ... Néanmoins, Twitter impose certaines limites pour les requêtes envoyées via *REST API* [2] (e.g 350 requêtes autorisées par heure et 180 requêtes par 15 minutes). Ces restrictions compliquent la cueillette d'un grand nombre de tweets. En outre, *REST API* ne permet pas l'extraction des tweets à l'aide d'une requête qui contient des mots-clés. Cette *API* utilisée plutôt par les chercheurs ou les développeurs qui désirent récupérer des informations ou des tweets de certaines personnes.

3.1.2 SEARCH API

Contrairement à la *REST API*, la *SEARCH API* permet d'interroger les données sur Twitter en utilisant une requête q composée des mots-clés. Si q = '`apple store`', *SEARCH API*[3] retourne les tweets qui contiennent les deux termes `apple` et `store`. Dans une requête, on peut utiliser certains opérateurs logiques comme *OR* et *AND* et - . La requête '`apple -store`' retourne les tweets qui contiennent le terme `apple` mais pas le terme `store`. On peut également filtrer les résultats selon plusieurs critères tels que :

[2] `https://dev.twitter.com/docs/rate-limiting/1.1`
[3] `https://dev.twitter.com/docs/using-search`

Langue des tweets spécifier la langue avec laquelle le tweet est écrit.

La période trouver les tweets écrits entre une date `since` et une date `until`.

Type de résultats spécifier le type de tweets retournés les plus populaires (les plus retweetés), les plus récents ou mixtes (mélange entre les plus populaires et les plus récents).

La *SEARCH API* a des limites :

- De même que la *REST API*, le nombre de requêtes autorisées est de 350 requêtes/heure et 180 requêtes/15 minutes.
- La taille d'une requête ne peut pas dépasser 1000 caractères y compris les opérateurs.
- Le nombre de tweets retournés par requête ne peut pas dépasser 1500.
- Il ne peut pas trouver les tweets qui étaient envoyés il y a plus d'une semaine.

La figure 3.1 montre un exemple de tweet retourné par *SEARCH API*,

Figure 3.1 – Tweet retourné par `search API` (format *Json*). Ligne 1 : texte du tweet ; Ligne 4 : langue du texte (identifié par Twitter).

```
01 text='Ok, jeudi il fera 31 degré. #Tunisie #auMax', id=193873044287143936
02 toUserId=-1, toUser='null',
03 fromUser='AnisKhez', fromUserId=121504252,
04 isoLanguageCode='it',
05 source='<a href="http://twitter.com/#!/download/iphone"rel="nofollow">Twitter
   for iPhone</a>',
06 profileImageUrl='http://a0.twimg.com/profile_images
   2008493088/229604_10150324090608888_573338887_7473932_392125_n_normal.jpg',
07 createdAt=Sat Apr 21 21:25:10 EDT 2012,
08 location='null',
09 place=null, geoLocation=null, annotations=null,
10 userMentionEntities=[],
11 urlEntities=[],
12 hashtagEntities=[HashtagEntityJSONImpl{start=28, end=36, text='Tunisie'},
   HashtagEntityJSONImpl{start=37, end=43, text='auMax'}],
13 mediaEntities=null
```

3.1.3 STREAMING API

La *STREAMING API* [4] permet d'obtenir des tweets en temps réel, en lançant une requête d'une durée illimitée tant qu'il n'y a pas de problème de connexion au serveur ou que le programme n'a pas été arrêté. On peut filtrer les tweets à l'aide de plusieurs paramètres tels que :

Mot-clés : un ensemble de mots-clés (jusqu'à 400), séparés par des virgules, qui filtre les tweets qui seront fournis par l'API. Un mot-clé est composé d'un ou plusieurs termes séparés par des espaces. Chaque tweet retourné doit contenir au moins un mot-clé. Plusieurs champs sont considérés pour retourner les tweets adéquats : texte, hashtags, usernames...

User IDs : une liste d'utilisateurs (jusqu'à 5 000), séparés par des virgules. Chaque tweet retourné est : écrit , est *retweeté*, répond à un tweet envoyé, *retweete* un tweet écrit ou répond à un tweet écrit par un utilisateur qui appartient à cette liste.

Langue : depuis mars 2013, *STREAMING API* offre la possibilité de filtrer les tweets par la langue dans laquelle ils sont écrits. Cependant, nous avons constaté que le détecteur de la langue employé par Twitter retourne parfois des mauvais résultats.

Géolocalisation : une liste des emplacements représentés par deux paires (*longitude*, *latitude*) qui limitent les zones désirées. Par exemple : [(-78.925781, 45.671644) ; (-63.720703, 62.367449)] sont deux paires de *longititude* et *latitude* qui limitent la province de Québec (Canada). Ce paramètre permet de retourner les tweets envoyés depuis la région spécifiée (Québec) indépendamment de l'attribut d'emplacement de l'utilisateur : les tweets d'un utilisateur ayant indiqué que son emplacement est le Québec ne seront pas considérés s'ils ne sont pas envoyés à partir du Québec. Les retweets ne sont pas couverts par ce paramètre de géolocalisation. Ce paramètre constitue une autre possibilité pour retourner les tweets mais n'est pas un filtre pour les résultats retournés par les autres paramètres. C'est-à dire,

[4] `https://dev.twitter.com/docs/streaming-api`

si les paramètres **Mots-clés** = ['harper'], **Langue=** `fr` et **Géolocalisation** = [`(-78.925781, 45.671644); (-63.720703, 62.367449)`], les tweets retournés doivent (contenir le terme `harper` ET être écrits en français) OU ils sont envoyés du Québec.

Trois niveaux d'accès sont offerts pour le *STREAMING API* : le *Spritzer*, le *Gardenhose* et le *Firehose*, qui permettent respectivement d'accéder à 1%, 10% et 100% des tweets qui correspondent à une requête donnée. Le seul niveau d'accès disponible au grand public est le *Spritzer*, celui-ci permet de récupérer au plus un échantillon de 1% de tous les tweets envoyés à ce moment. On obtient un échantillon de 60% des tweets qui répondent à la requête, mais cet échantillon correspond à 1% de l'ensemble de tous les tweets. Le *Firehose* permet d'accéder à tous les tweets qui correspondent à la requête donnée. Pour obtenir plus de tweets, il faut contacter *Gnip* [5], la société autorisée par Twitter à vendre les données. *Gnip* annonce que les prix commencent à partir de 500 dollars américains. Pour le quatrième trimestre de l'année 2013, la vente des données représente 9,5 % de chiffres d'affaires de Twitter, soit environ 23 millions de dollars [6].

Morstatter et al. (2013) ont étudié la couverture, de l'échantillon fournit par le *Spritzer*, des activités effectuées sur Twitter. Les auteurs ont comparé des tweets retournés par le *Spritzer* et le *Firehose*, en utilisant exactement les mêmes paramètres. Ils rapportent que plus le nombre de tweets le *Spritzer* augmente plus leur couverture des sujets est précise. Étant donné que l'objectif principal de notre thèse est de détecter les préoccupations d'utilisateurs dans une période donnée, il fallait collecter une grande quantité de données et d'une façon continue. Pour cette raison, nous avons opté pour la *STREAMING API* dont la figure 3.2 montre. Il s'agit d'un retweet envoyé par l'utilisateur *lizpelmeri*. Le résultat retourné contient beaucoup de détails tels que :

- Informations générales sur le tweet : ID, date de création de tweet, s'il est un retweet ou non, une réponse ou non, son emplacement (d'où il est écrit)... S'il est une réponse, on trouve l'IDs du tweet et d'utilisateur à qui il répond. (figure 3.2 ① et ③)

[5] `http://gnip.com/products/pricing/`
[6] `http://mashable.com/2014/04/15/twitter-buys-gnip/`

Figure 3.2 – Extrait d'un tweet retourné par *STEAMING API* (format *Json*). ① : informations sur le tweet (date, identifiant,retweet ou non, réponse ou non . . .) ; ② : informations sur le retweeteur (l'utilisateur qui a retweeté) puisqu'il s'agit d'un retweet ; ③ : informations sur le tweet original (date, identifiant,retweet ou non, réponse ou non . . .) ; ④ : informations sur le propriétaire de tweet ; les entités (hashtags, hyperliens, utilisateurs mentionnés) trouvées dans le tweet original et le retweet (sont toujours les mêmes).

- Les entités trouvées (hashtags, URLs, les utilisateurs mentionnés) avec leurs indices dans le texte.

- Informations détaillées sur l'utilisateur : son ID, nom et prénom, la date d'inscription sur Twitter, son emplacement, sa langue préférée... (figure 3.2 ② et ④)

- S'il est un retweet, on trouve les mêmes informations (informations générales, entités, l'expéditeur ...) sur le tweet en question et sur le tweet original.

3.2 Caractéristiques des données

Tableau 3.I – Ensemble de mots-clés utilisés pour extraire des tweets (qui portent sur la Tunisie entre le 07 février et le 15 avril 2012) à l'aide du *STREAMING API*

Mots-clés	**Définition**
`marzouki`	Président actuel de la Tunisie
`hammadi jebali`	Premier ministre dans cette période
`Tunisie`, `tounes`, `Tunisia`	`tounes` est la prononciation arabe de Tunisie
`tnelec`	Les élections tunisiennes, nous avons utilisé ce terme parce que nous avons trouvé que beaucoup d'utilisateurs en parlent jusqu'à présent
`sebsi`	Ex-premier ministre (après la révolution tunisienne)
`nahdha`, `ennahdha`	Le parti au pouvoir durant cette période
`ghannouchi`	Chef d'`ennahda`
`sidi bouzid`	La région où la révolution tunisienne a commencé
`14jan`	14 janvier est la date de fuite de `Ben ali` (président déchu)

Dans nos expérimentations, nous avons analysé les tweets qui portent sur la Tunisie. Notre but est détecter les événements qui se sont déroulés et qui se rapportent à la Tunisie au cours d'une période donnée. Nous avons extrait les tweets d'une façon continue pendant plusieurs mois à l'aide du *STREAMING API* (section 3.1.3). Pour recueillir des

tweets, nous avons utilisé un ensemble de mots-clés donnés au dans le tableau 3.I, tous fortement liés à la Tunisie.

Nous avons réussi à extraire 276 505 tweets entre le 08 février 2012 et le 15 avril 2012. Les tableaux 3.III et 3.II illustrent des statistiques effectuées sur le corpus obtenu.

Tableau 3.II – Statistiques sur le nombre des mots trouvés dans les tweets qui portent sur la Tunisie (276 505 tweets publiés entre le 08 février et le 15 avril 2012). Le tweet nettoyé ne contient pas les *hashtags*, les *utilisateurs mentionnés* et les *hyperliens*

Nombre minimal de mots	0
Nombre maximal de mots	37
Nombre moyen de mots	12.42

Tableau 3.III – Statistiques sur les tweets qui portent sur la Tunisie publiés entre le 08 février et le 15 avril 2012).

Numbre de tweets	276 505
Nombre des retweets	32 890
Nombre d'utilisateurs distincts	26 093
Nombre de tweets qui contiennent au moins un hashtag	147 395
Nombre de tweets qui contiennent au moins un utilisateur mentionné	88 595
Nombre de tweets qui contiennent au moins un hyperlien	168 309
Nombre de hashtags distincts	12 218

3.3 Conclusion

Dans ce chapitre nous avons présenté les caractéristiques (fonctionnement et limites) des trois APIs fournies afin d'extraire des données à partir de Twitter. Comme notre objectif est de détecter les sujets qui stimulent l'intérêt des internautes dans une période continue, nous avons utilisé la *STREAMING API* pour collecter les tweets portant sur la Tunisie. Cette API permet de récupérer les tweets par un ensemble de mots-clés et par la géolocalisation. Cependant, nous avons essayé de récupérer les tweets en utilisant les informations de géolocalisation de la Tunisie, mais nous n'avons pas réussi à extraire aucun tweet car les coordonnées géographiques de la Tunisie ne sont pas prises en compte par Twitter. Pour cette raison, nous nous sommes plutôt basés sur les mots-clés pour re-

cueillir nos tweets. Dans la deuxième partie, nous avons présenté des statistiques sur le corpus collecté.

Au chapitre suivant, nous décrivons les particularités des tweets écrits, principalement, par des Tunisiens et les algorithmes que nous proposons afin de regrouper les termes similaires.

CHAPITRE 4

REGROUPEMENT DES TERMES

Nos méthodes se basent sur les termes trouvés dans les tweets pour déterminer les sujets saillants, or un sujet est souvent représenté par plus d'un terme. Il est donc important de regrouper les termes référant un même sujet, sinon chaque terme dans le corpus représentera un sujet différent.

Généralement les textes générés par les utilisateurs sur le web, notamment dans les microblogs, contiennent des mots non standards où les utilisateurs commettent souvent des fautes d'orthographe et utilisent des abréviations. Cela conduit à obtenir plusieurs variantes pour un même terme. Plusieurs travaux (Clark et Araki (2011), Sproat et al. (2001)) ont essayé de normaliser automatiquement les termes, c.-à-d de transformer toutes les variantes en un terme unique.

Dans ce chapitre, nous commençons par la présentation des particularités et les caractéristiques des données que nous traitons dans notre travail puis nous décrivons les méthodes que nous avons proposées afin de regrouper les termes identiques et/ou sémantiquement similaires. La deuxième partie est consacrée à la description des tâches de normalisation consistant à attribuer un même code aux hashtags identiques, mais écrits de différentes façons (fautes orthographes, translittération des mots arabes par les utilisateurs). Par la suite, nous présentons les algorithmes proposés permettant de regrouper les termes sémantiquement similaires en nous basant sur des méthodes statistiques.

4.1 Dialecte tunisien

Le dialecte tunisien est la langue employée par tous les Tunisiens, appelé *darija*. Il est différent de l'arabe standard et des autres dialectes arabes. Ce dialecte est très influencé par la langue française, ce qui fait que les Tunisiens utilisent souvent des vocabulaires et des expressions françaises dans leurs communications. D'autres vocabulaires sont aussi utilisés, mais plus rarement, comme l'anglais, le punique, le berbère ou l'italien. De ce

fait, le vocabulaire du dialecte tunisien est composé principalement de :

- mots arabes provenant de l'arabe standard.
- mots tunisiens : utilisés seulement par les tunisiens, la plupart des mots sont empruntés du berbère, du punique, de l'italien, de l'espagnol, du français et du turc.
- mots français

Tableau 4.I – Exemples de mots du vocabulaire tunisien **VT** ; **SA** : signification en arabe ; **SF** : signification en français.

VT (prononciation)	**SA (prononciation)**	**SF**
(chnowa) شنوة	(ma) ما	quoi
(barcha) برشة	(kathir) كثير	beaucoup
(karhaba) كرهبة	(sayara) سيارة	voiture

Le mode d'écriture employé par les Tunisiens dans les SMS et les médias sociaux présente d'autres caractéristiques.

- Un même mot peut être écrit avec l'alphabet latin ou l'alphabet arabe.
- Lorsqu'un mot arabe est écrit en alphabet latin, les lettres arabes ne pouvant être transcrites directement sont remplacées par un chiffre dont la forme rappelle vaguement la lettre en arabe ou avec deux lettres qui rappellent la prononciation de la lettre. Le tableau 4.II montre les équivalents les plus utilisés.

Tableau 4.II – Exemples de lettres arabes et leurs équivalents en alphabet latin

Lettres	ح	ق	ع	خ	ش	غ	ض;ظ; ذ	ث
Équivalents	7 ; *ha*	9 ; *ka*	3 ; *aa*	5 ; *kh*	*ch*	8 ; *gh*	*dh*	*th*

Ces caractéristiques d'écriture amènent une grande variété car le même mot peut être écrit de plusieurs façons, d'où la nécessité d'une certaine normalisation.

Le tableau 4.III présente des tweets écrits par des Tunisiens durant les élections qui ont eu lieu en octobre 2011. Nous remarquons que la façon d'écrire est très variable. Le premier tweet est composé de vocabulaires français et de termes arabes latinisés, le deuxième est écrit uniquement en français avec des abréviations et des fautes d'orthographe, tandis que le troisième est un mélange de texte écrit avec l'alphabet arabe et l'alphabet latin.

Tableau 4.III – Exemples de tweets écrits par des tunisiens

Tweet	**Explications**
`j'ai voté, ta7ya tounes #TnElec #Vote`	L'auteur a informé qu'il a déjà voté. Les mots `ta7ya` et `tounes`, dans le deuxième tweet, sont des néographes des mots arabes qui correspondent respectivement à `vive` et `la Tunisie`.
`@so9rat11 nous sommes ts tunisiens et ns ns devons de respecter la loi q est au dessus de ts! #tnelec #Tunisie`	C'est une réponse au tweet de `@so9rat11`. `ns`=`nous`, `q`=`qui`, `ts`=`tous`.
تصويرة بن علي رجعت في حلق الوادي `http://t.co/2w4Ishy0 excellent !!! #tnelec`	Le texte arabe signifie que la photo de `Ben Ali` (président tunisien déchu) a été republiée dans `La Goulette` (ville tunisienne).

Dans la littérature certains travaux se sont intéressés au dialecte tunisien. mais pour des textes bruts plutôt que pour des tweets, medium qui nous intéresse dans cette thèse. Une autre différence de cette thèse par rapport aux autres travaux, c'est que les tweets sur lesquels on travaille sont écrits en alphabet latin, arabe ou avec un mélange des deux. Tandis que dans la littérature on ne considère pas la mixture d'alphabets différents au sein d'un même texte.

Même s'il existe des ressources linguistiques permettant de déterminer les relations sémantiques entre les termes de dialecte tunisien, nous avons toujours besoin d'autres techniques pour déterminer les termes sémantiquement similaires. Ceci est dû à l'évolution dynamique du vocabulaire dans les médias sociaux. Dans les sections 4.2 et 4.3, nous présentons les différentes techniques proposées afin d'identifier les termes similaires.

4.2 Normalisation des termes

Dans cette section nous présentons les techniques que nous avons appliquées pour normaliser les termes (principalement les hashtags). Afin d'obtenir la bonne normalisation, nous avons profité de certaines informations (e.g hyperliens trouvés dans les tweets) fournies par Twitter.

4.2.1 *Soundex*

Comme mentionné aux chapitres précédents, les utilisateurs commettent souvent des fautes d'orthographe ce qui crée plusieurs variantes pour un même terme. Ce problème a été déjà traité par les systèmes de correction orthographique à l'aide d'algorithmes phonétiques. Ces algorithmes indexent les mots selon leur prononciation. Le principe consiste à utiliser la prononciation d'un mot mal écrit pour prédire le bon mot, avec la même prononciation, qui lui correspond. Dans notre travail, nous avons eu recours à l'algorithme de *Soundex* (Russell (1918)) afin de normaliser les termes qui ont une même prononciation.

Algorithm 1 *Soundex* pour le français. **Texte souligné :** les modifications effectuées pour l'adapter aux particularités de nos données

Entrée : mot *m* à convertir
CodeSoundex ← *m*
CodeSoundex ← première lettre de *m*
Remplacement de *aa*, *â* et *a* (seulement au début) par *3*
Désaccentuer toutes les lettres
Remplacement de certains groupes de lettres dans *m* :
ki ←*gui* ; *ke*←*gue* ; *ka*←*ga* ; *ko*←*go* ; *k*←*gu* ; *ka*←*ca* ; *ko*←*co* ; *ku*←*cu* ; *k*←*q* ;
k←*cc* ; *k*←*ck* ; *k*←*9* ; *kh*←*5* ; *h*←*7*
Remplacer les voyelles (*e*, *i*, *o*, *u*) par *a*
Remplacer les préfixes (*mac*, *asa*, *kn*, *pf*, *sch*, *ph*) respectivement par (*mcc*, *aza*, *nn*, *ff*, *sss*, *ff*).
Enlever les *h* sauf *sh* et *ch* // Garder toujours les *h*
Enlever les *y* sauf précédé par *a*
Supprimer les terminaisons *a*, *d*, *t*, *s* // Garder toutes les terminaisons
Supprimer les *a* sauf au début
Enlever les doublons
if *Longueur(CodeSoundex)* >5 **then**
  Garder les 5 premières lettres
else {*Longueur(CodeSoundex)* <5}
  Ramener *Longueur(CodeSoundex)* à 5, en ajoutant des zéro
end if
Sortie : *CodeSoundex*

Comme les prononciations varient d'une langue à une autre, il existe plusieurs variantes de *Soundex*. Au début, nous avons appliqué un *Soundex* pour le français standard puisque la prononciation des Tunisiens ressemble à celle des Français (voir algorithme 1). Nous avons ainsi réussi à normaliser correctement plusieurs termes. Cependant, cette version *Soundex* a éprouvé des difficultés à normaliser certains termes notamment les termes arabes romanisés. Par exemple le code *Soundex* `rd00` est associé aux chaines : `3ridha`, `9rouda`, `radhia` et `radio`, alors qu'il n'y a aucune relation entre eux. Aussi les termes `e5wan` et `ikhwan` ont deux codes différents `awn0` et `akwn` même s'ils ont une même prononciation étant donné que le chiffre `5` remplace `kh`. Nous avons donc effectué certaines modifications pour qu'il puisse traiter de telles données (voir l'algorithme 1, texte souligné).

Les codes *Soundex* des termes `3ridha`, `9rouda`, `radhia` et `radio` deviennent res-

pectivement `3rdh`, `9rd0`, `rdh0` et `rd00` et les termes `e5wan` et `ikhwan` ont maintenant le même code *Soundex* : `akhw`.

Nous avons appliqué cet algorithme principalement sur les hashtags afin de regrouper les hashtags avec des prononciations similaires. Cependant, les utilisateurs créent souvent des hashtags pour référer à des événements en utilisant des dates. Par exemple : `#9avril`, `#7march`, `#mars2012`, `#3juillet`. Les hashtags `#07march`, `#7mars` et `#mars07` sont similaires tandis qu'ils ne sont pas associés à un même code Soundex (`0hmr`, `hmrs`, `mrs0`) selon l'algorithme 1. Pour cela, nous avons distingué les hashtags qui représentent des dates avant d'appliquer l'algorithme 1. Pour ce faire nous détectons les hashtags qui correspondent à une expression régulière qui représente les formats possibles d'une date. Pour regrouper les hashtags qui se réfèrent à une même date. Par exemple, les hashtags `#07march`, `#7mars` et `#mars07` appartiennent au même *cluster*.

4.2.2 Translittération

L'algorithme *Soundex* défini ci-dessus ne traite que les termes écrits avec l'alphabet latin. Cependant, Twitter permet de créer des hashtags écrits avec d'autres alphabets, dont l'alphabet arabe. Ce phénomène est employé souvent par les Tunisiens. Un même terme peut être écrit parfois en alphabet arabe et parfois en alphabet latin (voir tableau 4.I). Pour regrouper ces termes, nous avons implémenté un algorithme de romanisation basé sur des règles de conversion entre un terme en alphabet arabe et son équivalent en alphabet latin.

La romanisation est un système qui permet la transformation (translittération) d'une écriture non latine vers une écriture latine. Il s'agit de la substitution de chaque graphème, d'un système d'écriture source, par son équivalent dans le système d'écriture cible. Le problème de translittération été abordé par plusieurs chercheurs dont Shao et Ng (2004) principalement dans le domaine de *cross-language information retrieval* (CLIR).

Tableau 4.IV – Exemples d'équivalences graphématiques entre les alphabets arabe et latin

Équivalent latin	**Graphème arabe**
n	ن
t	ت
w, *ou*	و
f, *v*, *ph*	ف
k, *q*, *c*	ك
b, *p*	ب
i, *y*	ي

Un terme, écrit en alphabet arabe, a plusieurs équivalents en alphabet latin. Par exemple le terme غنيم peut être transformé à `ghanim`, `ghaneem`, `ghonim`, `ghoneem`, `ghinim` ... Cela est dû à principalement à l'absence de diacritiques (appelés aussi les voyelles brèves) dans la plupart de textes écrits dans l'alphabet arabe utilisé dans les médias sociaux. Ce qui conduit à une variété de prononciations. En outre, Halpern (2007) a constaté que la différence de prononciation d'un dialecte à un autre et le son, de certains graphèmes arabes, qui n'existe pas dans d'autres langues (e.g français), occasionnent une grande variété dans la translittération. Pour construire notre translittérateur, nous nous sommes basés sur une approche ascendante. Il s'agit de trouver pour chaque graphème, du terme arabe ses équivalents possibles en latin et de concaténer les graphèmes obtenus. Pour ce faire, nous avons créé manuellement un ensemble de règles adapté au dialecte tunisien pour transformer les graphèmes arabes par leurs équivalents latins. Le tableau 4.IV montre un échantillon des règles. L'équivalent d'une voyelle arabe (ي, و et أ) dépend de sa position (début, milieu, fin) et du type (voyelle, consonne) de la lettre qui la précède ou qui la suit. Par exemple la lettre ي est substituée par '*i*' (et n'est pas à côté d'une voyelle dans la translittération) s'il est au milieu du terme et par '*y*' s'il est au début, et la lettre و est substituée par '*w*' et par '*ou*' s'il est au milieu (et n'est pas à côté d'une voyelle dans la translittération) du terme.

Plusieurs translittérations sont possibles pour un terme t_a. Par exemple, le terme فارس, peut être converti à {'*fares*', '*vares*', '*feres*', '*veres*','*phares*', '*pheras*', '*pheres*'}.

Pour obtenir la bonne translittération de t_a, on génère tous les candidats (translittérations) possibles. Par la suite, chaque hashtag sera remplacé par son code *Soundex*. Afin de déterminer la bonne translittération, parmi $\mathscr{C}$ (l'ensemble de différents codes *Soundex* obtenus), nous avons procédé les étapes suivantes :

1. Sélectionner parmi les éléments de $\mathscr{C}$ qui existent déjà dans l'ensemble des codes *Soundex* pour les hashtags écrits initialement en latin $\mathscr{H}_L$.

2. Si aucun élément de $\mathscr{C}$ est trouvé dans l'ensemble $\mathscr{H}_L$, on prend n'importe quel candidat de $\mathscr{C}$.

3. Sinon, s'il y a plus d'une translittération de $\mathscr{C}$ trouvés déjà dans $\mathscr{H}_L$, nous choisissons la bonne translittération en fonction de trois critères dans l'ordre suivant :

   (a) Choisir le code *Soundex* qui partage le plus grand nombre d'hyperliens avec t_a (voir section 4.3.3) ;

   (b) Sinon, choisir le codes *Soundex* qui apparaît le plus au cours d'un même jour avec t_a ;

   (c) Sinon, on garde le hashtag original.

4.3 Regroupement des hashtags

4.3.1 Cooccurrence avec des hashtags

La relation sémantique entre deux termes sert à déterminer leurs degrés d'association. Cette information joue un rôle important dans plusieurs domaines de TALN tels que la construction automatique des thésaurus, la recherche d'informations . . .. Par exemple, il est utile d'utiliser les termes similaires à ceux spécifiés dans la requête d'utilisateur pour récupérer les documents pertinents. Plusieurs travaux se sont basés sur des bases construites manuellement par des linguistes (e.g *Wordnet*) pour déterminer la relation

sémantique entre les termes. Ces bases contiennent des informations indiquant le type de relation (synonyme, antonyme ...) entre les termes. Cependant, elles ne couvrent pas les dialectes ni le mode d'écriture (fautes, abréviations) employée dans les médias sociaux. En outre, le vocabulaire employé dans les médias sociaux est enrichi fréquemment par de nouveaux termes inventés par les utilisateurs ou qui représentent des sujets (e.g produit, personnes, parti politique ...).

Une autre approche, basée sur des techniques statistiques, permet de fournir une information quantitative indiquant le degré de la similarité sémantique entre les termes.Cette information est estimée à partir des données observées, en se basant sur la notion de la cooccurrence. La cooccurrence est l'apparition simultanée de deux ou plusieurs termes dans une même fenêtre. Une fenêtre peut être un paragraphe, une phrase ... Deux termes qui cooccurrent appartiennent souvent à un même contexte. Par exemple, les termes *loi* et *avocat* apparaissent fréquemment ensemble dans un même contexte, *la justice*. Dans notre thèse, nous avons implémenté des méthodes basées sur la cooccurrence afin de regrouper les termes (principalement les hashtags) d'un contexte (sujet) commun. Étant donné que les tweets sont courts et ne contiennent pas assez de mots, nous avons considéré le tweet entier comme fenêtre. Nous avons calculé la cooccurrence de chaque hashtag dans notre corpus et nous avons constaté que les hashtags, qui cooccurrent fréquemment, sont similaires ou réfèrent à un même sujet. Nous avons étudié les hashtags (`#manouba` et `#mannouba`) les plus représentatifs du sujet de la mise en berne de drapeau tunisien, le 07 mars 2012. Il s'agit de la mise en berne du drapeau tunisien par un étudiant "salafiste" dans le bâtiment de la faculté des lettres, des arts et des humanités de *Manouba*. Une étudiante tunisienne a essayé de l'empêcher d'enlever le drapeau. Les salafistes sont des personnes qui sont jugées par la plupart des Tunisiens comme des musulmans extrémistes et radicaux. Cet acte, qui a eu lieu le 07 mars 2012, a entraîné une vague de colère chez plusieurs Tunisiens qui ont considéré qu'un tel acte est un outrage portant atteinte à leur dignité et la souveraineté du pays. Le tableau 4.V montre les hashtags qui ont le plus grand nombre d'occurrences avec les hashtags `#manouba` et `#mannouba`.

Tableau 4.V – Fréquence des hashtags liés au sujet *Manouba*. Certaines personnes demandent au *gouvernement*, *ennahdha* et l'*assemblée constituante* d'intervenir et empêcher ce genre de comportement ; les salafistes ont organisé un sit-in à l'université pour revendiquer le droit des étudiantes de porter le *niqab* ; la mise en berne du *drapeau tunisien* a été faite par un étudiant `salafiste`.

	Hashtags cooccurrant avec le sujet		**nombre de fois**
	Sujet	***Hashtags***	
`Manouba`	Tunisie	`Tunisie`	702
		`Tunisia`	
		`tunisi`	
		`tn`	
	Le gouvernement Tunisien	`tngov`	126
	L'assemblée constituante	`tnac`	108
	Salafistes	`salafistes`	106
		`salafiste`	
		`salafisme`	
		`salafis`	
		`salafist`	
	Ennahdha	`ennahdha`	88
		`nahdha`	
		`ennahda`	
	Drapeau Tunisien	`drapeau`	11
		`touchepasamondrapeau`	
	Niqab	`niqab`	6

Le nombre de cooccurrences d'une paire de termes ne reflète pas toujours leur degré d'association. Par exemple, le degré d'association des hashtags de sujet `drapeau tunisien` et ceux de sujet `mannouba` est plus grand que celui des hashtags de sujet Tunisie et ceux de hashtags `mannouba`, même si le nombre de cooccurrences de ces derniers est plus élevé. C'est que les hashtags de sujet `Tunisie` cooccurrent avec plusieurs, tandis que les hashtags de sujet `drapeau tunisien` apparaissent la plupart de temps avec les hashtags de sujet `mannouba`. Pour cela, plusieurs mesures statistiques sont proposées pour déterminer le degré d'association de deux termes. Ces mesures se basent sur un tableau de contingence. Ce tableau contient des informations sur l'apparition de chaque paire de termes. (voir tableau 4.VI).

Tableau 4.VI – Tableau de contingence d'une paire de hashtags (terme 1 et terme 2). a = nombre de fois où hashtag 1 et hashtag 2 apparaissent ensemble ; b = nombre de fois où hashtag 1 est présent sans la présence de hashtag 2 ; c = nombre de fois où hashtag 1 absent et hashtag 2 présent ; d = nombre de fois où hashtag 1 et hashtag 2 sont absents

	$hashtag2$	$\overline{hashtag2}$
$hashtag1$	a	b
$\overline{hashtag1}$	c	d

Pour mesurer le degré de relation entre deux hashtags nous avons utilisé la mesure *pointwise mutual information* (*PMI*) (Church et Hanks, 1989). La *PMI* mesure la quantité d'information apportée pour la présence simultanée d'une paire de termes (dans notre cas les hashtags)

$$PMI(hashtag_i, hashtag_j) = \log(\frac{P(hashtag_i \& hashtag_j)}{P(hashtag_i)P(hashtag_j)}) = \log(\frac{N * a}{(a+b)*(a+c)})$$

N est le nombre de tweets considérés. *P(*$hashtag_i$ *&* $hashtag_j$*)* est la probabilité que $hashtag_i$ et $hashtag_j$ apparaissent ensemble dans un même document. *P(*$hashtag_i$*)P(*$hashtag_j$*)* est la probabilité que $hashtag_i$ et $hashtag_j$ apparaissent ensemble, s'ils sont statistiquement indépendants. Le ratio P($hashtag_i$ & $hashtag_j$) et P($hashtag_i$)P($hashtag_j$) mesure le degré de dépendance entre $hashtag_i$ et $hashtag_j$. *PMI* est maximisé lorsque $hashtag_i$ et $hashtag_j$ sont parfaitement associés

4.3.2 Regroupement avec *DBscan*

Le *regroupement* est une technique d'apprentissage machine qui permet de regrouper des éléments de fortes ressemblances dans des *clusters*. Les éléments qui n'appartiennent pas aux mêmes *clusters* sont de faible similarité. Le degré de similarité, entre les éléments, est déterminé généralement à l'aide d'une fonction de distance. Plus la distance est faible, plus les éléments sont similaires. Nous avons utilisé cette technique (*regroupement*) pour regrouper les hashtags similaires. Nous avons considéré la *PMI* de deux hashtags comme une mesure de similarité. Plus la *PMI* est grande, plus les hashtags sont similaires :

$$Similarity(hashtag_i, hashtag_j) = PMI(hashtag_i, hashtag_j) \tag{4.1}$$

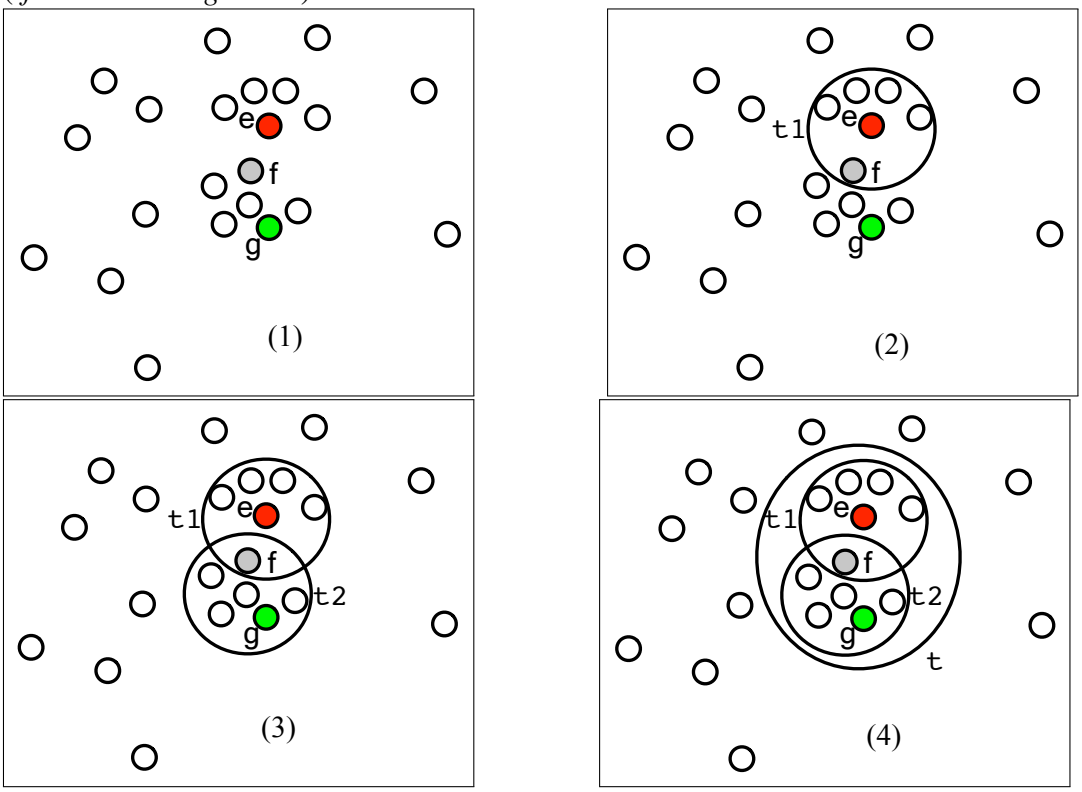
Cependant, pour la plupart des algorithmes de *regroupement* il peut être difficile de savoir quels sont les paramètres d'entrée à utiliser, car l'utilisateur doit disposer de suffisamment de connaissances sur les données surtout le nombre de *clusters* obtenus à la fin. À titre d'exemple, l'algorithme *k-moyenne* nécessite de spécifier à l'avance le nombre *k* de *clusters* à utiliser. Comme il est difficile de trouver la valeur de *k* qui donne le bon résultat, il faut en tester plusieurs ce qui est très coûteux en terme de temps.

Dans notre cas, la spécification du nombre de *clusters* dès le début n'est pas réaliste, puisque chaque *cluster* représente un sujet et nous ne pouvons pas deviner le nombre de sujets discutés dans le corpus. En outre, le *k-moyenne* est incapable de gérer les bruits et les exceptions, car chaque objet doit être associé à un *cluster*. D'autres algorithmes de *regroupement* n'exigent pas de spécifier le nombre de *clusters*, par exemple l'algorithme *Density-Based Spatial Clustering of Applications with Noise* (*DBscan*) qui est basé sur la notion de la densité. L'idée principale de *DBscan* est que le voisinage d'un rayon ε donné doit contenir au moins un nombre minimal d'éléments donné (*MinPts*). Autrement dit, pour chaque élément, le *DBscan* récupère son ε-*voisinage* et vérifie s'il contient bien au moins *MinPts* éléments. Si oui, l'élément en question devait appartenir à un cluster *c* composé de ses ε-*voisinage*, sinon il s'agit d'un bruit. Par la suite, pour chaque élément de *c DBscan* récupère ses ε-*voisinage* (s'ils existent) et les fusionne avec *c*. La figure 4.1 illustre le fonctionnement de *DBscan*. Pour *MinPts* = 5, l'élément *g* ne fait pas partie de *t1* (ε-*voisinage* de *e*), tandis que *g* et *e* appartiennent à un même *cluster t* puisque *g* appartiennent à *t2* (ε-*voisinage* de l'un de ε-*voisinage* de *e* (qui est le *f*))

Les paramètres ε et *MinPts* sont donnés par l'utilisateur, où ε correspond généralement à la distance entre deux points (*éléments*).

Nous avons appliqué *DBscan* pour regrouper les hashtags similaires. Rappelons que nous avons supposé que la similarité entre deux hashtags est déterminée par la mesure *PMI*, pour cela la valeur ε correspond à la valeur de *PMI*. Ainsi, les points (hashtags) qui appartiennent à ε-*voisinage*, d'un hashtag *h*, sont ceux qui ont une *PMI* avec *h* supérieur

Figure 4.1 – Fonctionnement *DBscan* pour *MinPts* = 5. Le point de départ est '*e*'. *DBscan* construit *t1* ('*e*' ∪ ε-*voisinage* de '*e*'). '*f*' fait partie de *t1* et ses ε-*voisinage* $\geq$ *MinPts* ('*f*' ∪ ε-*voisinage* = *t2*) donc on fusionne *t1* et *t2*.



ou égal à ε. L'algorithme 2 (page 45) montre le fonctionnement de *DBscan* pour regrouper les hashtags. Nous avons considéré que les hashtags, qui se trouvent dans un même *cluster*, représentent un même évènement. Cependant, nous avons constaté que *DBscan* regroupe des hashtags qui ne sont pas, vraiment, similaires. Ceci est dû au phénomène de la fusion illustrée dans la figure 4.1. À titre d'exemple, supposons que l'élément *f* correspond au hashtag *#manifestation* qui appartient à deux voisinages qui représentent deux évènements différents, pendant lesquels des manifestations sont organisées . L'application de la fusion conduit au *regroupement* des hashtags, qui portent sur deux sujets différents, dans un même *cluster*. Afin d'assurer plus de cohésion pour un *cluster*, nous avons décidé de ne pas fusionner directement un ε-*voisinage*$'$ à ε-*voisinage*, mais de vérifier pour chaque élément dans ε-*voisinage*$'$ son degré de similarité avec tous les éléments dans ε-*voisinage*. Pour déterminer le degré de similarité, nous utilisons la fonc-

tion $Cohesion(C,e)$ déterminée comme suit :

$$Cohesion(C,e) = \frac{1}{|C|}\sum_{i=1}^{|C|} Similarity(h_i,e) \tag{4.2}$$

e est un hashtag dont on va mesurer le degré de similarité avec les hashtags trouvés dans un *cluster C*. *Cohesion(C,e)* retourne la moyenne des valeurs de *PMI* entre le hashtag e et ceux trouvés dans C. Plus la valeur de *Cohesion(C,e)* est grande plus le hashtag e est similaire aux hashtags trouvés dans C. Pour ajouter e à C la valeur de *Cohesion(C,e)* doit dépasser un certain seuil λ défini manuellement.

Nous présentons ci-dessous un exemple fictif illustrant le fonctionnement de l'algorithme après les modifications présentées ci-dessus. Le tableau 4.VII montre les valeurs de *PMI* entre les points (dans notre cas des hashtags). *L'élément (i,j)* correspond à la valeur de *PMI* de $point_i$ et $point_j$. #سلفي est la translittération arabe du terme *salafi*.

Tableau 4.VII – Exemples des hashtags avec les valeurs de *PMI*. **#7mars** réfère à la date 7 mars 2012 ; **#manif** est une abréviation du terme *manifestation*.

	#7mars	**#drapeau**	**#salafi**	**#manif**	**#ghanim**	**#extremiste**	**#excission**	#سلفي
#7mars	-	10,0	9,0	7,3	-1,0	-2,6	-4,0	6,0
#drapeau	-	-	12,0	7,1	10,4	-1,0	4,0	11,0
#salafi	-	-	-	8,0	-1,0	9,0	-0,2	1,0
#manif	-	-	-	-	10,0	7,0	-3,0	-2,0
#ghanim	-	-	-	-	-	12,0	3,0	-1,0
#extremiste	-	-	-	-	-	-	13,0	4,0
#excission	-	-	-	-	-	-	-	9,0
#سلفي	-	-	-	-	-	-	-	-

Supposons que ε =10, *MinPts* = 3 et λ = 7. Les étapes ci-dessus présentent le processus de *regroupement* des points (hashtags) :

1. Choisir un point au hasard, commençons par `#salafi`.

2. Trouver les $\varepsilon - voisinage$ de `#salafi` = {`#7mars`,`#drapeau`}.
   $|\{$`#salafi`, `#7mars`, `#drapeau`$\}| \geq MinPts$, alors {`#salafi`, `#7mars`, `#drapeau`} forment un nouveau *cluster*.

Algorithm 2 L'algorithme *Dbscan*

Données : ε (valeur minimale de *PMI* entre deux hashatags), *MinHstgs* (le nombre minimal de hashtags dans ε-*voisinage*) et H (l'ensemble de hashtags)
$C \leftarrow 0$ // initialiser le nombre de *clusters* à 0
for chaque hashtag h n'est pas visité **do**
  ε-*voisinage* $\leftarrow$ epsilonVoisinage(h, ε)
  if tailleDe(ε-*voisinage*) $<$*MinHstgs* **then**
    marquer h Comme *NOISE* // h n'appartient à aucun *cluster*.
  else {*il y a au moins* MinHstgs *points voisins à* h}
    $C \leftarrow C + 1$
    ExpandCluster(H, h, ε-*voisinage*, C, ε, *MinHstgs*)
  end if
end for

ExpandCluster(*H*, *h*, ε-*voisinage*, *C*, ε, *MinHstgs*)
begin
ajouter h au *cluster* C
for chaque hashtag *h'* de ε-*voisinage* **do**
  si *h'* n'a pas été visité
  marquer *h'* comme visité
  ε-*voisinage*$'$ $\leftarrow$ epsilonVoisinage(H, *h'*, ε)
  si tailleDe(ε-*voisinage*$'$) $\geq$ *MinHstgs*
  ε-*voisinage* $\leftarrow$ ε-*voisinage* $\cup$ ε-*voisinage*$'$
  si *h'* n'est membre d'aucun *cluster*
  ajouter *h'* au *cluster* C
end for
end ExpandCluster

EpsilonVoisinage(*H*, *h*, ε)
retourner tous les hashtags de H qui ont une valeur *PMI* supérieur ou égale à ε avec h

3. Trouver les $\varepsilon - voisinage$ #7mars et #drapeau. Pour #7mars il n'y a pas de voisinage $\geq MinPts$. La taille de #drapeau et ses $\varepsilon - voisinage$ ($\{$#سلفي et #ghanim$\}$) $\geq MinPts$, Alors on calcule *Cohesion*($\{$#7mars, #drapeau$\}$,#سلفي) et *Cohesion*($\{$#7mars, #drapeau$\}$, #ghanim) :

   - *Cohesion*($\{$#7mars, #drapeau$\}$,#سلفي)
     $= \frac{PMI(\#7mars,\#سلفي)+PMI(\#drapeau,\#سلفي)}{2}$
     $= \frac{6+11}{2} = 8.5$. *Cohesion*($\{$#7mars, #drapeau$\}$,#سلفي) $\geq \lambda$, alors on ajoute #سلفي à $\{$#7mars, #drapeau, #salafi, #سلفي$\}$

   - *Cohesion*($\{$#7mars, #drapeau, #سلفي$\}$, #ghanim)
     $= \frac{PMI(\#7mars,\#ghanim)+PMI(\#drapeau,\#ghanim)+PMI(\#سلفي,\#ghanim)}{3}$
     $= \frac{1+10,4-1}{3} = 3,46$. *Cohesion*($\{$#7mars, #drapeau$\}$, #ghanim) $\leq \lambda$, alors on n'ajoute pas #ghanim à $\{$#7mars, #salafi, #drapeau$\}$

   - Maintenant, on cherche les $\varepsilon - voisinage$ de #سلفي (qui est devenu un nouveau point dans l'ensemble). La taille de $\{\varepsilon - voisinage$ de #سلفي$\} \cup \{$#سلفي$\} \leq MinPts$. Alors rien à faire.

   - Le *cluster* contient, donc, $\{$#7mars, #drapeau, #سلفي, #salafi$\}$

4. Prenons un nouveau point au hasard et qui n'est pas visité, qui est *f*.

5. La taille de $\{\varepsilon - voisinage$ ($\{$#excission, #ghanim$\}$) de #extremiste$\} \cup \{$#extremiste$\} \geq MinPts$, donc un nouveau *cluster* qui contient $\{$#extremiste, #ghanim, #excission$\}$. On vérifie les $\varepsilon - voisinage$ de #excission et #ghanim. Pour #excission il n'y a pas de voisinage (sauf #extremiste), donc rien à faire. Pour le #ghanim, la taille de $\{\varepsilon - voisinage$ ($\{$ #manif, #drapeau, #extremiste $\}$) de #ghanim$\} \cup \{$#ghanim$\} \geq MinPts$. Alors on calcule *Cohesion*($\{$#extremiste, #excission$\}$, #manif) et
   Cohesion($\{$#extremiste, #excission$\}$, #drapeau) :

   - *Cohesion*($\{$#extremiste, #excission$\}$, #manif)
     $= \frac{PMI(\#extremiste,\#manif)+PMI(\#excission,\#manif)}{3}$

$= \frac{7-3}{2} = 2 \leq \lambda$, alors on n'ajoute pas `#manif` à {`#extremiste`, `#ghanim`, `#excission`}

- Idem pour `#drapeau` parce que *Cohesion(*{*#extremiste, #excission, #ghanim*}*, #drapeau)* $= \frac{-1+4+10,4}{3} = 4,46 \leq \lambda$, alors on n'ajoute pas `#drapeau` à {`#extremiste`, `#ghanim`, `#excission`}
- Le *cluster* contient, donc, {`#excission`, `#ghanim`, `#extremiste`}

6. Prenons un nouveau point au hasard et qui n'est pas visité, il reste seulement `#manif`.

7. Trouver les $\varepsilon - voisinage$ de `#manif` = {`#ghanim`}. La taille de {`#manif`} $\cup$ {`#ghanim`} $\leq$ *MinPts*. Donc `#manif` est considéré comme bruit (il reste sans *cluster*)

8. Enfin, les *clusters* obtenus sont : {`#7mars`, `#drapeau`, `#salafi`, `#سلفي`}, {`#extremiste`, `#ghanim`, `#excission`} et {`#manif`} (bruit)

Chaque *cluster* obtenu ci-dessus représente un sujet (évènement). Le *cluster* {`#7mars`, `#drapeau`, `#salafi`, `#سلفي`} réfère à l'évènement de la mise en berne de drapeau tunisien (voir section 4.3.1). Tandis que le *cluster* {`#extremiste`, `#ghanim`, `#excission`} réfère à l'évènement de L'arrivée du prédicateur égyptien *Wajdi Ghonim* en Tunisie le 11 février 2012. Ce prédicateur est considéré par certains comme étant radical. Cela est dû à ses prises de position polémiques concernant des sujets à controverse. Une de ses fameuses prises de position se rapporte au sujet de l'excision des fillettes. Certains accusent le prédicateur d'avoir pris une position favorable envers l'excision.

4.3.3 Cooccurrence avec des hyperliens

Étant donné la taille réduite d'un tweet, les utilisateurs ne peuvent pas décrire des événements, échanger des informations ou exprimer leur avis d'une manière efficace. Pour contourner cette limite, Twitter offre à ses utilisateurs la possibilité d'ajouter des hyperliens vers des pages externes permettant de mieux expliquer et détailler le contenu

d'un tweet. Un hyperlien peut être une page qui contient un texte, des images, de l'audio et/ou vidéo. Les utilisateurs ont profité de cet élément pour enrichir leurs messages. Dans notre corpus, collecté entre le 08 février et 15 avril 2012, 62% des tweets contiennent au moins un hyperlien. Ce dernier constitue une information importante pour déterminer le degré d'association entre les termes, notamment les hashtags. Nous avons supposé que deux hashtags (h_i, h_j) qui apparaissent, presque, avec les mêmes hyperliens, alors h_i et h_j représentent le (s) même (s) sujet (s) détaillé (s) par ces hyperliens. Nous avons utilisé les notations suivantes pour représenter les données :

- $T_k = \{H_k = \{h_j \ldots\}, L_k = \{l_j \ldots\}\}$; T_k est le *k-ème* tweet, H_k et L_k sont respectivement les ensembles de hashtags et d'hyperliens trouvés dans T_k.

- $H = \{h \in H_k, 1 \leq k \leq |T|\}$. H est l'ensemble de différents hashtags trouvés dans T.

- $L = \{l \in L_k, 1 \leq k \leq |T|\}$. L est l'ensemble de différents hyperliens trouvés dans T.

- $L[h_i] = \{l \in L \mid h_i \in H_k \text{ et } l \in L_k,\ 1 \leq i \leq |H|,\ 1 \leq k \leq |T|\}$. $L[h_i]$ est l'ensemble des hyperliens qui apparaissent simultanément avec h_i.

- $LC[h_i, h_j] = L[h_i] \cap L[h_j] = \{l : l \in L[h_i] \wedge l \in L[h_j]\}$

- $\alpha_{i,j} = \frac{|LC_{i,j}|}{|L[h_i]|}$.

$\alpha_{i,j}$ est le rapport entre le nombre d'hyperliens partagés (entre h_i et h_i) et nombre total d'hyperliens qui apparaissent avec h_i. Plus la valeur de $\alpha_{i,j}$ plus grande, plus la possibilité, que h_i et h_j sont similaires et/ou portent sur un même sujet, est grande. Puisque $L[h_i] \neq L[h_j]$, alors $\alpha_{i,j} \neq \alpha_{j,i}$. Le tableau 4.VIII illustre que cette constatation est souvent valide. Le hashtag *#ghanim* se présente, dans un même tweet, avec 70 hyperliens différents.

Tableau 4.VIII – Nombre d'hyperliens partagés entre une paire de hashtags sémantiquement similaires

h_i	$\|L[h_i]\|$	h_j	$\|L[h_j]\|$	$\|LC_{i,j}\|$	$\alpha_{i,j}$	$\alpha_{j,i}$
#boussalem	2	*#innondation*	3	2	$\frac{2}{2}=1$	$\frac{2}{3}\simeq 0,7$
#lomia800	19	*#windows*	22	19	$\frac{19}{19}=1$	$\frac{19}{22}\simeq 0,9$
#ghanim	70	*#sheikhghanim*	2	2	$\frac{2}{70}\simeq 0,03$	$\frac{2}{2}=1$
#excision	22	*#wajdi_ghonim*	2	2	$\frac{2}{22}\simeq 0,1$	$\frac{2}{2}=1$

Des inondations ont eu lieu, durant le mois de février 2012, dans une région tunisienne intitulée *"boussalem"*. Pour cela les deux hashtags qui représentent les inondations (*#inondations*) et boussalem (*#boussalem*) partagent presque les mêmes hyperliens. Également les hashtags *#lomia800* et *#windows* partagent un important nombre d'hyperliens. En fait, *lomia800* est un téléphone intelligent lancé en hiver 2012, équipé du système d'exploitation Windows. Les hashtags *#sheikhghanim*,*#ghanim* et *#wajdi_ghonim* réfèrent à une même personne, *Wajdi Ghonim*.

Pour regrouper les hashtags nous avons utilisé, également, l'algorithme *DBscan*. La similarité entre deux hashtags est définie comme suit :

$$Similarity(hashtag_i, hashtag_j) = \frac{(\alpha_{i,j} + \alpha_{j,i})}{2} \tag{4.3}$$

Cette fois-ci la valeur ε de *DBscan* correspond la valeur de *Similarity* (équation 4.3) entre deux hashtags, définie manuellement. Pour qu'un hashtag h appartienne à $\varepsilon - voisinage$ de e, la valeur de $Similarity(e,h)$ doit être supérieure à ε.

Afin d'éviter le *regroupement* de hashtags qui ne portent pas sur le même sujet à cause de leurs relations avec des hashtags en commun (voir section 4.3.1), nous avons utilisé, également, la mesure de $cohesion(C,e)$ qui permet de déterminer la similarité entre un hashtag e et un *cluster* de hashtags C. La valeur $cohesion(C,e)$ doit dépasser un seuil (déterminé manuellement) pour ajouter e à C.

4.4 Topic Model

Dans la section précédente, nous avons présenté les méthodes proposées pour regrouper les hashtags similaires. Dans cette section, nous nous concentrons sur le *regroupement* des mots trouvés dans les tweets. Pour ce faire, nous avons eu recours à la technique du *Topic Model (TM)*.

TM est une technique statistique qui sert à détecter les sujets abstraits à partir d'une collection de documents. L'idée de base de *TM* est que chaque document peut être représenté comme un mélange de sujets latents, où un sujet est lui-même représenté comme une distribution de mots qui ont tendance à cooccurrer. Les mots fortement liés à un sujet ont des valeurs de probabilité plus grandes. Dans notre travail, nous avons profité de cette distribution pour obtenir les termes qui portent sur un même sujet. Les algorithmes de *TM* utilisent la modélisation de sac de mots qui représentent chaque document par les fréquences des mots qui le composent. Cela permet d'ignorer la syntaxe des phrases pour se concentrer sur les termes trouvés dans le document. Le nombre de sujets est un paramètre qui doit être donné avant l'application de *TM*.

Dans ce travail nous avons appliqué l'algorithme *Latent Dirichlet Allocation* (*LDA*) (Blei et al., 2003), qui est le *TM* le plus utilisé dans la littérature, à l'aide de la librairie *Mallet* (McCallum, 2002). Nous avons considéré chaque tweet comme étant un document différent. Pour utiliser *LDA* sur un corpus, nous devons spécifier le nombre d'itérations (*iter*). Ces itérations servent à raffiner le modèle suggéré par *LDA*. À chaque nouvelle itération, *LDA* met à jour le vocabulaire (les mots) utilisé pour décrire chaque sujet, ainsi que l'ajout d'un nouveau sujet ou encore la suppression d'un sujet déjà identifié dans les itérations précédentes. Il est recommandé de fixer un nombre d'itérations élevé (généralement entre 1000 et 2000) afin de garantir que le modèle final converge (i.e l'ensemble des sujets identifiés reste inchangé et stable) durant plusieurs itérations successives, dû à la stabilité des valeurs des paramètres du modèle

Après la tokenization[1] des tweets, nous avons effectué certaines tâches de prétraitement :

(a) Nous n'avons gardé dans les tweets que les informations textuelles (e.g les mots) :

[1] un processus permet de transformer les tweets en mots

nous avons supprimé les *usersnames* (identifiant d'un utilisateur dans Twitter) et les *hyperliens*.

(b) Nous avons supprimé les mots-clés utilisés pour extraire les tweets.

(c) Il est important de ne pas considérer les mots-vides (e.g *le la de parce que* ...) car ces mots ne sont pas porteurs de sens. Au début, nous avons considéré les mots vides parmi le vocabulaire. Comme les mots vides sont fréquents et très couramment utilisés dans notre corpus, il arrive très souvent que les algorithmes de *TM* construisent des sujets dont la plupart composés de mots vides. De cette façon, les *clusters* des mots, qui représentent les topics, contiennent souvent des termes nuisibles (mots vides) qui détériorent la qualité des résultats retournés par *LDA*. Le tableau 4.IX montre un exemple de sujets que nous avons pu identifier quand on a considéré les mots-vides.

Tableau 4.IX – Ensembles de mots qui représentent 4 sujets différents après l'application de *TM* en considérant les *mots vides* ; le nombre de sujets spécifié = 200.

Sujet 81	***Sujet 52***	***Sujet 74***	***Sujet 71***
est	*tunisie*	*the*	(dans) في
pour	*en*	*of*	(non) لا
en	*la*	*to*	(de, que, ...) أن
et	*je*	*is*	(ne, ne pas) لن
de	*de*	*in*	(ne, ne pas) لن
la	*tu*	*it*	(gouvernement) حكومة
le	*pas*	*we*	(nahdha) النهضة
qui	*est*	*do*	(palestine) فلسطين
ce	*que*	*are*	(ceux) هاذم
il	*et*	*was*	(fin, limite,...)حد

En second lieu, nous avons filtré les mots vides, pour que le *TM* ne les considère pas comme candidats potentiels. Le tableau 4.X montre un exemple de résultats obtenus par *LDA*, en appliquant les mêmes paramètres (nombre d'itérations et nombre de

topics) utilisés dans l'exemple précédent. En comparant ces deux exemples, il est facile de constater que les termes considérés dans le tableau 4.X sont plus significatifs que ceux considérés dans le tableau 4.IX. Basés sur ces observations, nous avons donc décidé d'éliminer les mots vides afin d'obtenir une bonne extraction des sujets par *LDA*.

Tableau 4.X – Ensembles de mots qui représentent 3 sujets différents après l'application de *TM* en éliminant les *mots vides* ; le nombre de sujets spécifié = 200.

Sujet 10	***Sujet 66***	***Sujet 192***
hamma	*manouba*	(ghanim) غنيم
plainte	*faculté*	(wajdi) وجدي
hammami	*salafistes*	(sheikh) شيخ
wajdi	*étudiants*	(mosqué) جامع
ennahdha	*lettres*	(prédicateur) داعية
porter	*drapeau*	(débat) نقاش
excision	*niqab*	(leçon) محاضرة
ghanim	*chaos*	(Abd) عبد
ghonim	*pays*	(moumrou) مورو
visite	*face*	(excision) ختان

Compte tenu du fait que les Tunisiens écrivent en plusieurs langues, nous avons utilisé une liste existante de mots vides en anglais, français et arabe. Nous avons, également, créé manuellement une liste pour le *dialecte tunisien*. Les utilisateurs tunisiens écrivent souvent en français, mais ils font souvent des erreurs d'orthographe et utilisent des abréviations, par exemple *pcq* au lieu de *parce que*. Les listes existantes de mots vides français ne contiennent pas les abréviations et les fautes. Comme le type d'écriture dans les tweets est très similaire à celui utilisé dans les SMS, nous avons utilisé un corpus parallèle d'environ 7 000 SMS (Langlais et al.) et nous avons calculé une *distance Levenshtein* Levenshtein (1966) pour trouver les équivalents (abréviations ou des fautes d'orthographe) pour chaque terme dans

la liste des mots vides (voir tableau 4.XI). Le tableau 4.XII donne les statistiques concernant les mots vides.

Tableau 4.XI – Des exemples de mots vides français et leurs possibles équivalents (les abréviations, les fautes d'orthographe)

Mots vides français	**Équivalents**
puisque	*psk*, *psq*, *puisq*, *puisk*, ...
aussi	*ossi*, *ossi*, *oci*, *o6i*...
comment	*commen*, *cmnt*, *kommen*, *kmnt*, *comon*, *komon*, ...

Tableau 4.XII – statistiques sur les *mots vides* dans de nombreuses langues et dialectes de notre corpus.

Langue	**Nombre**	**proportion**
Anglais	174	17%
Français	283	28%
Français (les abréviations, les fautes d'orthographe)	329	32%
Arabe	159	16%
Dialecte Tunisien	75	7%
Total	1020	100 %

Tweets partageant le même hashtag sont fusionnés dans le même document : Pour obtenir des données plus riches, nous avons regroupé au sein d'un seul document les tweets partageant un même hashtag, même s'ils sont en différentes langues (e.g en arabe ou en français). Chaque *cluster* va représenter un document, sous forme de plusieurs tweets. Avec cette technique, les termes qui sont sémantiquement proches seront regroupés, ce qui permet d'obtenir un vocabulaire beaucoup plus riche qu'en se limitant à un simple tweet. Cette méthode amène *LDA* à retourner des sujets plus significatifs quand il est utilisé sur un long document que sur un court tweet. De plus, notre méthode de *regroupement* de tweets réunit des termes provenant de langues différentes (français, arabe, dialecte tunisien). Le tableau 4.XIII montre des tweets qui ont été regroupés, car ils partagent le même hashtag `#9avril`. À titre d'exemple, les mots `police` et شرطة sont considérés dans le

même sac de mots ; il s'agit de synonyme provenant de deux langues différentes (français, arabe). L'application de cette technique sert à la création de plusieurs sujets composés de termes similaires qui sont écrits dans différentes langues, voici un exemple des sujets obtenus : {*martyrs*, الفاسد (abusif), *police*, بوليس (police), *fête*, *pire*, *avril*, شهيد (martyr)}.

Tableau 4.XIII – Exemple de tweets partageant le même hashtag (#9AVRIL)

Langue	**Tweet**
Français	#ACAB #TUNISIE #LARAYEDH **#9AVRIL** manif dispersée violemment réprimée ! !
Arabic	الشرطة تلاحق المتاظهرين في كل من شارع باريس و مرسيليا و جون جوريس #tunisie #9avril
English	15 :15- police break up small protest on Mohamed V, chasing protesters with sticks **#9AVRIL** #TUNISIE

4.5 Conclusion

Dans ce chapitre nous avons essayé de regrouper les termes qui discutent du même sujet. L'objectif de ce *regroupement* est de déterminer à partir d'un terme, l'ensemble de termes qui lui sont reliés. Pour atteindre cet objectif, nous avons exploité plusieurs techniques : *PMI* (section 4.3.1), l'algorithme de *regroupement DBscan* (section 4.3.2), la translittération (section 4.2.2), *Soundex* (4.2.1). Nous avons réussi à exploiter ces techniques normalement considérées comme des champs de recherche séparés. Nous les avons adaptées pour nos données en profitant des informations (date de tweet, *hyperliens*...) fournies par Twitter.

Le travail présenté dans ce chapitre constitue un prétraitement pour le prochain chapitre dans lequel nous étendrons les tweets par les hashtags regroupés afin d'identifier les tweets similaires discutant du même sujet.

CHAPITRE 5

DÉTECTION DES ÉVÈNEMENTS

Au chapitre précédent, nous avons présenté des méthodes pour regrouper les termes sémantiquement similaires. Afin d'atteindre cet objectif, nous avons exploité des techniques du TALN, et nous avons profité de certaines informations offertes par Twitter.

Dans ce chapitre, nous revenons sur les travaux qui s'intéressent aux évènements dans les médias sociaux, en particulier Twitter, notre source de données dans cette thèse. Nous soulignons l'importance des hashtags pour signaler la présence d'un évènement et l'apport du *regroupement* des termes dans le processus de détection des sujets discutés au cours d'une période donnée. Dans la dernière partie de ce chapitre, nous présentons les méthodes utilisées afin de déterminer les dates saillantes des évènements identifiés.

Allan et al. (1998) ont défini un évènement par quelque chose d'unique qui arrive à un certain point dans le temps. Kleinberg (2003) fait remarquer qu'une tendance (ou un évènement qui stimule l'intérêt) dans un flux de document est signalée par une explosion d'activité via certaines caractéristiques avec une fréquence marquée, les caractéristiques étant les termes associés à l'évènement en question.

5.1 Évènement dans les médias sociaux

La détection des évènements est un champ de recherche étudié depuis des années, il est souvent appelé *Topic Detection and Tracking* (*TDT*). La *TDT* est facilitée par l'existence de flux quotidiens des journaux sur le web. La motivation de cette tâche est l'implémentation d'un système d'alerte qui permet de détecter et d'analyser, à partir d'un flux de documents, les évènements majeurs (Allan, 2002). Les recherches, qui ont abordé ce thème, ont utilisé principalement des techniques de TALN (e.g lemmatisation, détermination des parties du discours,...). Étant donné les tailles, généralement grandes, des documents analysés contenant des informations bien structurées, ces techniques fonctionnent de manière efficace.

Dans la littérature, nous avons distingué deux types de travaux qui s'intéressent à l'identification des évènements :

évènements connus à priori : Ces travaux se concentrent sur les évènements dont les caractéristiques (type, nom, emplacement ...) sont connues au préalable. Certains travaux (Sakaki et al. (2010)) s'intéressent à l'identification de documents (messages) qui discutent d'un événement particulier (e.g tremblement de terre, concert...) en formulant des requêtes contenant ses caractéristiques. D'autres travaux (Chakrabarti et Punera (2011), Shamma et al. (2010)) s'intéressent à la génération des résumés des messages qui discutent d'un évènement particulier. Petrovic et al. (2010) ont essayé de trouver le premier message qui discute d'un évènement précis.

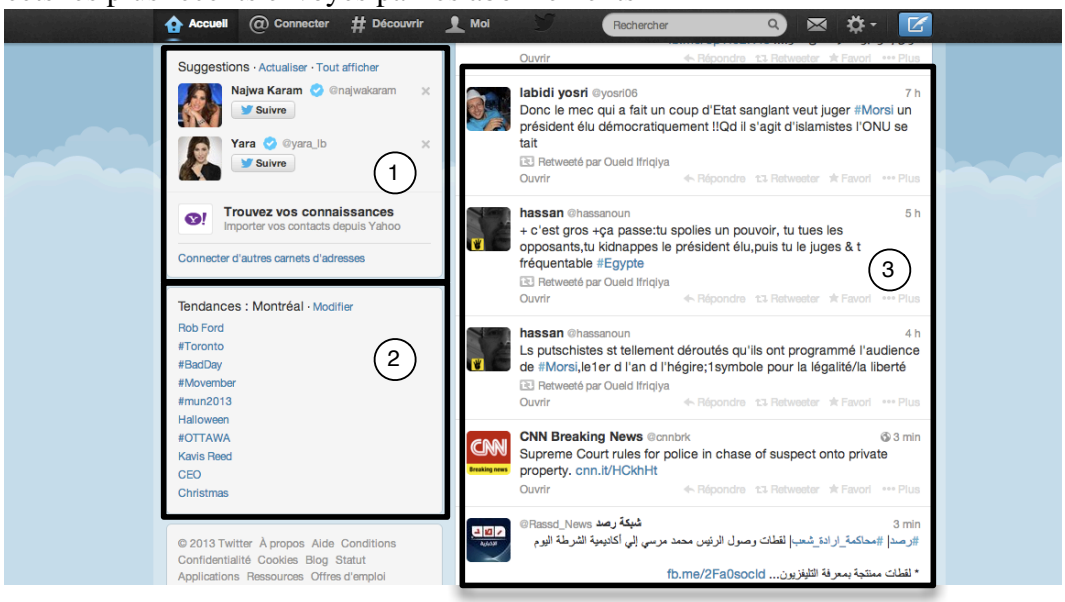
évènements inconnus : Dans ce cas, les recherches s'intéressent à la détecter des tendances en identifiant les sujets inédits ou en croissance rapide au sein d'un flux de documents (Kontostathis et al. (2004)). Comme discuté au chapitre 1, plusieurs travaux ont montré que les utilisateurs de Twitter discutent et partagent des nouvelles à propos d'évènements imprévus (e.g tremblement de terre). Pour cette raison les travaux qui s'intéressent à détection des tendances (ou les évènements inconnus) ont eu recours à d'autres indices permettant de signaler la présence d'un évènement dans une période.

Dans cette thèse, nous nous sommes basé sur ces définitions pour détecter les évènements importants qui stimulent l'intérêt public dans une période donnée. Un évènement e, au cours d'une période T, est représenté par un ensemble de *traits* F_e dont les fréquences augmentent brusquement à un ou plusieurs points t_e inclus dans T. Dans nos expériences, nous nous basons sur des fréquences quotidiennes.

Twitter offre déjà un service qui permet d'afficher, chaque heure, les dix meilleures tendances (sous forme de termes) dans la barre à gauche de la page d'accueil d'un utilisateur. Les tendances sont, par défaut, personnalisées par l'emplacement de l'utilisateur et changent régulièrement à un intervalle de quelques minutes. La figure 5.1 montre une

capture d'écran d'une page d'accueil Twitter qui montre des tendances pour le Québec (le *04 novembre 2013 vers 12h : 7mn PM*). L'algorithme utilisé par ce service est inconnu. Les tendances affichées dépendent de l'emplacement de l'utilisateur et non pas de son profil. Ceci explique la présence des tweets, envoyés par les abonnements de l'utilisateur, qui ne sont pas liés au Québec (voir Figure 5.1, Zone 3) . Nous avons constaté que les tendances affichées réfèrent aux termes les plus fréquents à un moment donné. Les termes peuvent être des hashtags ou des mots dans les tweets. Ce service ne regroupe pas les termes qui représentent la même tendance, par exemple, lors de décès de *Michael Jackson* la plupart des tendances étaient autour de ce sujet : *Michael Jackson*, *MJ*, *King of Pop* ... (Kwak et al. (2010)). Cependant, nous avons remarqué que Twitter n'affiche pas les tendances pour toutes les régions. Par exemple, nous avons essayé d'afficher les tendances pour la Tunisie, mais nous avons remarqué qu'elle ne fait pas partie de la liste des régions.

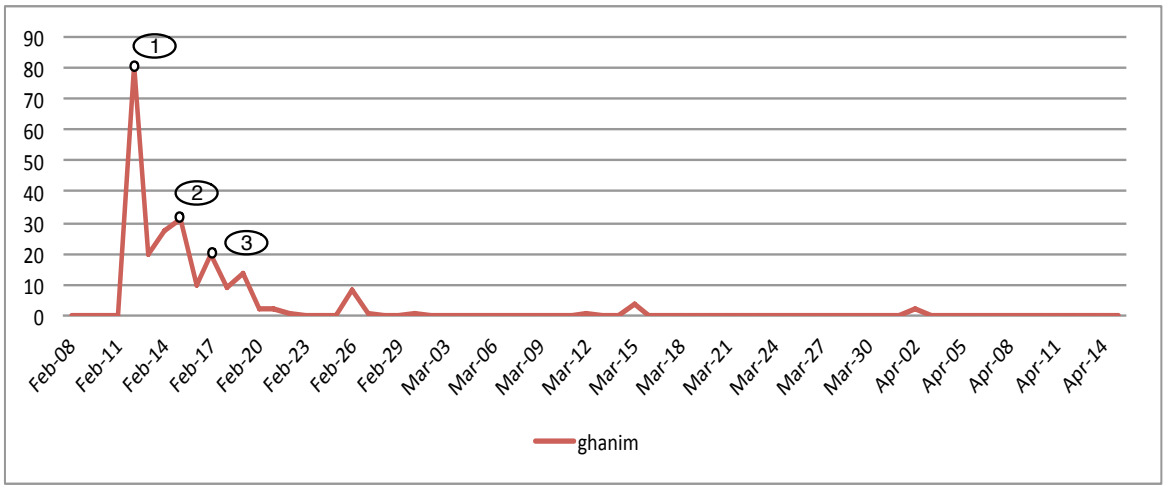
Figure 5.1 – Capture d'écran qui montre les tendances (Zone 2), pour la région du Québec (*04 novembre 2013 vers 12h : 7mn PM*), affichées dans la page d'accueil de Twitter ; Zone 1 : une liste des utilisateurs à suivre suggérée par Twitter ; Zone 3 : les tweets les plus récents envoyés par les abonnements



5.2 Évènement vs. Hashtags

Un hashtag est un mot clé, précédé par le symbole '#', employé par les utilisateurs dans Twitter pour spécifier le(s) sujet(s) de leurs tweets. Étant donné le nombre important de tweets qui contiennent au moins un hashtag (voir chapitre 2) et la difficulté de comprendre les tweets, nous avons essayé, en premier lieu, de nous baser sur les hashtags pour identifier les sujets les plus discutés par les internautes. Nous avons constaté que cet élément joue un rôle très important pour avoir une idée sur les préoccupations des utilisateurs. La figure 5.2 montre la fréquence quotidienne du hashtag, `#ghanim`, relatif à l'arrivée du prédicateur égyptien *Wajdi Ghanim* en Tunisie (discuté au le chapitre 3). Les pics observés dans la figure 5.2 illustrent une forte corrélation entre les occurrences des évènements réels et le hashtag `#ghanim`. Le premier pic correspond à la date de l'annonce de la visite (12 février) de *Wajdi Ghanim* qui est arrivé dans la nuit du 11 février, le deuxième pic (15 février) est produit à cause des deux plaintes qui ont été déposées contre ce prédicateur et le troisième est occasionné par les manifestations organisées contre cette visite.

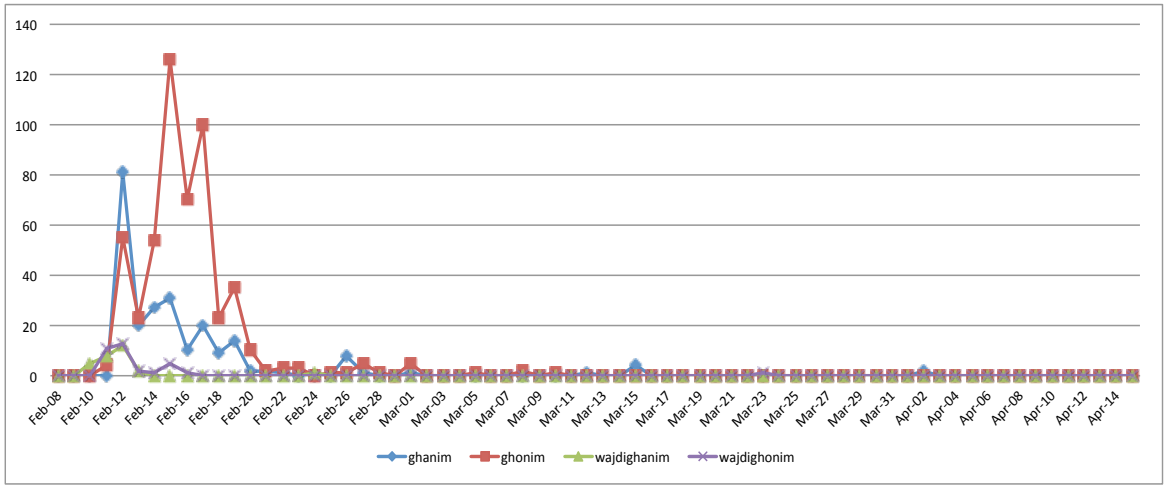
Figure 5.2 – Distribution par jour du hashtag `#ghanim` dans la période entre 08 février et 15 avril 2012. ① : 12 février ; ② : 15 février ; ③ : 17 février



Comme un sujet est souvent représenté par plus d'un hashtag, il est nécessaire d'identifier les hashtags sémantiquement similaires (discutant le même sujet), sinon chaque

hashtag représentera un évènement différent. Concernant le sujet, l'arrivée de *Wajdi Ghanim*, discuté ci-dessus, plusieurs hashtags (36 hashtags identifiés manuellement) réfèrent au prédicateur. Nous présentons dans la figure la distribution de quatre hashtags (`#ghanim`, `#ghonim`, `#wajdighanim` et `#wajdighonim`) relatifs à cet évènement. La figure 5.3 montre que l'apparition du `#ghanim` commence le 12 février, tandis que l'utilisation du `#ghonim` a commencé le 11 février (la date réelle de l'arrivée du prédicateur) et le hashtag `#wajdighanim` a même débuté la veille (10 février) de l'arrivée.

Figure 5.3 – Distribution par jours des hashtags (`#ghanim`, `#ghonim`, `#wajdighanim` et `#wajdighonim`) relatifs à l'évènement de la visite de *Wajdi Ghanim* dans la période 08 février et 15 avril 2012



Le même hashtag (`#wajdighanim`), qui a signalé le début de l'évènement, n'est plus utilisé dès le 14 février même les utilisateurs n'avaient pas cessé d'en discuter. Cela est probablement justifié par la création des nouveaux hashtags relatifs à l'évènement et qui sont devenus plus populaires au sein de la communauté. La figure 5.4 montre que le *regroupement* des hashtags (relatifs à un même évènement) donne un impact positif pour une détection précoce de l'évènement, aussi pour la durée de vie de l'évènement. Cette figure montre que cet évènement a été discuté entre les 10 et 23 février.

Ce type de *regroupement* permet de donner plus d'importance à l'évènement en sommant les fréquences des hashtags reliés. Les fréquences du hashtag `#ghanim` aux dates de 12 février, 15 février et 17 février sont respectivement 81, 31 et 20. Le *regroupement*

des quatre hashtags (`#ghanim`, `#ghonim`, `#wajdighanim` et `#wajdighonim`) accroît les fréquences de l'évènement dans cette période, où elles sont devenues 161, 162 et 120 respectivement aux dates des 12, 15 et 17 février (voir figure 5.5). Ces dernières fréquences permettent de rendre l'évènement plus marquant et qui frappe plus l'attention.

5.3 Fréquences quotidiennes des *clusters*

L'apparition d'une tendance ou d'un évènement au cours d'une période donnée T, est signalée par l'explosion des fréquences à un moment M. Dans ce travail, nous nous sommes basés sur cette définition pour identifier les évènements à partir des tweets. Au début, nous avons commencé par calculer les fréquences de différents termes (hashtags et mots) trouvés dans notre corpus. Cependant, nous avons constaté que cette manière n'est pas très efficace étant donné qu'un évènement peut être référé par plus d'un terme. Pour cette raison, au lieu de calculer la fréquence quotidienne de chaque terme, il est préférable de calculer la fréquence quotidienne d'un *cluster* des termes représentant un évènement. Pour ce faire, nous avons utilisé une méthode simple qui incrémente la fréquence quotidienne F_{gj} d'un *cluster* g au jour j, si au moins l'un des termes de g se trouve dans TW_{ij}, où TW_{ij} est le tweet i au jour j. Nous obtenons ainsi pour chaque évènement (représenté par son *cluster* des termes) sa fréquence quotidienne. Étant donné qu'un tweet n'est composé que de quelques termes (hashtags et/ ou mots) alors si l'un de ses termes est relatif à un évènement e, nous pouvons déduire que ce tweet porte probablement sur e. Avec cette méthode, nous avons réussi à dégager les intentions d'utilisateurs à partir des tweets. Le tableau 5.I montre les fréquences quotidiennes de *clusters* de hashtags relatifs à l'évènement de la *visite de prédicateur ' Wajdi Ghonim'*. Le *regroupement* des hashtags obtenu est obtenu par la technique de *Soundex* présentée au chapitre 3.

Figure 5.4 – Durée de vie de chaque hashtag relatif à l'évènement de la visite de *Wajdi Ghanim*. *La D.E* : la durée de vie de l'évènement.

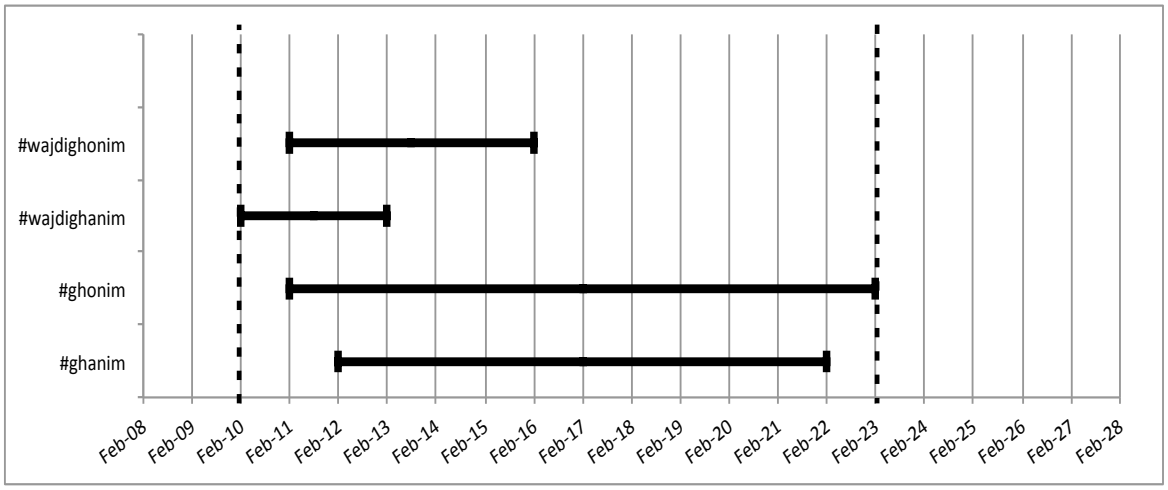


Figure 5.5 – La somme de fréquence quotidienne des hashtags (`#ghanim`, `#ghonim`, `#wajdighanim` et `#wajdighonim`) relatifs à l'évènement de Wajdi Ghanim

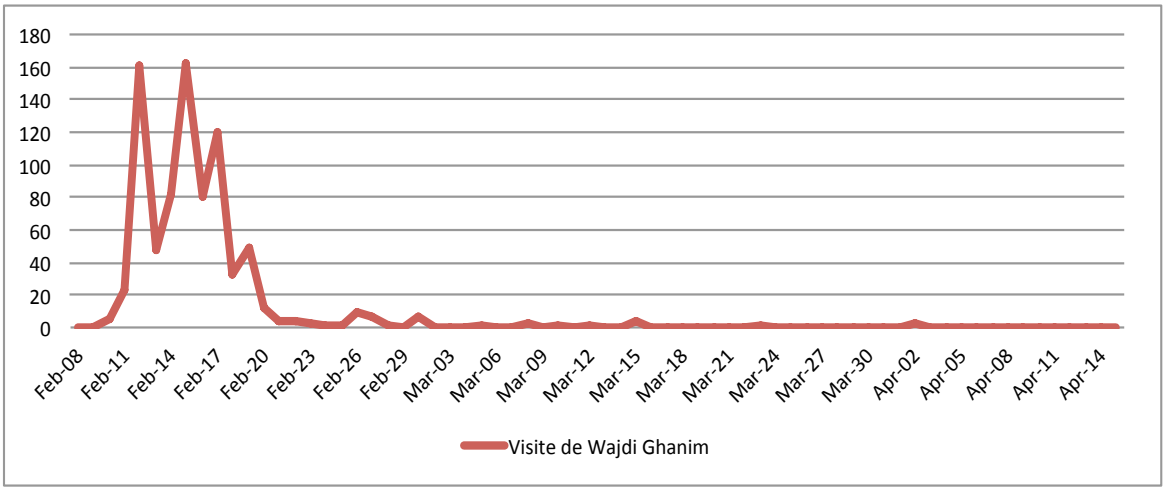


Tableau 5.I – Fréquence quotidienne de *cluster* de hashtags liés à *Wajdi Ghonim*. GHNM est le code *Soundex* des hashtags relatifs à *Wajdi Ghonim* : {#ghanim, #ghenim, #ghnaim, #ghnim, #ghoneim, #ghonem, #ghonim, #ghonnim, #ghounim, #ghénim}. Toutes les dates sont en 2012

	08-02	*09-02*	*10-02*	*11-02*	*12-02*	*13-02*	*14-02*	*15-02*	*16-02*
GHNM	3	4	4	12	35	8	8	56	1

Les dates avec les plus grandes valeurs de fréquences font référence aux dates importantes des évènements.

5.4 Regroupement des tweets

Les travaux sur la détection des évènements utilisent des techniques de *regroupement* afin de partitionner le corpus en *clusters* dont les documents sont supposés discuter sur le même sujet (évènement). Les méthodes de *regroupement* existantes représentent un document sous forme d'un vecteur de *traits* ainsi que l'importance de chaque *trait* dans le document. Les *traits* sont généralement les termes du vocabulaire utilisé dans le corpus. Dans cette représentation vectorielle, l'importance d'un terme dans un document est donnée par un score tels qu'un score binaire, le produit *Term Frequency-Inverse Document Frequency*.

La représentation booléenne est la première représentation adoptée dans la recherche d'informations. Un document est caractérisé par la présence (valeur 1) ou l'absence (valeur 0) de chaque terme du vocabulaire dans son contenu. Soit les deux textes suivants :

***d1* :** *Anonymous de la tunisie s'attaque à Ennahdha*

***d2* :** *c à la tunisie non ?*

La représentation booléenne de ces textes est illustrée par le tableau 5.II

Tableau 5.II – Exemple d'une représentation booléenne

	Termes								
	à	*Anonymous*	*attaque*	*c*	*Ennahdha*	*la*	*non*	*s'*	*tunisie*
d1	1	1	1	0	1	1	0	1	1
d2	1	0	0	1	0	1	1	0	1

Cette présentation n'évalue pas l'importance des *traits*. Contrairement à la méthode de *Tf-Idf* qui attribue une pondération à chaque terme trouvé dans le document. Le poids de chaque terme dépend de sa fréquence dans le document en question ainsi que de sa présence dans le corpus.

Tf_i ***(Term frequency)*** **:** le nombre d'occurrences du terme t_i dans le document d.

Idf ***(Inverse Document Frequency)*** **:** sert à évaluer la pertinence du terme dans le corpus. Cette mesure pénalise les termes les plus fréquents dans le corpus, plus le terme est présent dans le corpus, plus la valeur *Idf* diminue. La mesure *Idf* est calculée de la manière suivante :

$$Idf(t_i) = \log \frac{|D|}{|d_j : t_i \in d_j|}$$

où $|D|$ le nombre de documents dans le corpus ; $|d_j : t_i \in d_j|$ est le nombre de documents où t_i apparait.

Le tableau 5.III montre les valeurs *Idf* pour certains hashtags trouvés dans notre corpus. On constate que les *IDFs* des hashtags `#Tunisie` et `#Tunisia` sont moins élevés par rapport aux autres `#securité` et `#Cinema`. Cela est expliqué par la présence fréquente de `#Tunisie` et `#Tunisia` dans notre corpus.

Tableau 5.III – La valeur *Idf* de certains hashtags dans le corpus

Hashtags	***Idf***
`#Tunisie`	0,08
`#Tunisia`	1,07
`#securité`	4,48
`#Cinema`	3,84

Comme tous les algorithmes de *regroupement*, les algorithmes de *regroupement* de documents adoptent des mesures de similarité afin de comparer chaque paire de documents. Les mesures de similarité les plus utilisées sont : le *cosinus de similarité* et l'*indice de Jaccard*.

Cosinus de similarité. est la mesure la plus populaire pour la comparaison des documents. Le cosinus de similarité entre deux vecteurs de documents est calculé de la manière suivante :

$$Cosinus(\vec{d}_a, \vec{d}_b) = \frac{\vec{d}_a.\vec{d}_b}{|\vec{d}_a|.|\vec{d}_b|} = \frac{\sum_{i=1}^{n} d_{ai} * d_{bi}}{\sqrt{\sum_{i=1}^{n} d_{ai}^2} * \sqrt{\sum_{i=1}^{n} d_{bi}^2}}$$

où $\vec{d}_a.\vec{d}_b$ est le produit scalaire des vecteurs relatifs aux document $\vec{d}_a$ et $\vec{d}_b$; $|\vec{d}_a|$ et $|\vec{d}_b|$ sont respectivement les normes des vecteurs $\vec{d}_a$ et $\vec{d}_b$.

La valeur retournée par $Cosinus(\vec{d}_a, \vec{d}_b)$ est entre 0 (si $\vec{d}_a$ et $\vec{d}_b$ sont totalement différents) et 1 (si $\vec{d}_a$ et $\vec{d}_b$ sont identiques).

Cette mesure est appliquée seulement si les vecteurs sont composés des valeurs numériques. Dans le *regroupement* de documents, les vecteurs sont construits par des valeurs *Tf-Idf* des termes.

L'indice de Jaccard est une mesure qui permet de comparer deux ensembles. Dans le cas de *regroupement* de documents, elle est plus utile dans le cas de vecteurs binaires. l'indice de Jaccard de $\vec{d}_a$ et $\vec{d}_b$ est obtenu par le rapport de nombre de termes en commun et le nombre de différents termes dans d_a et d_b. Soit :

- A : Le nombre de termes qui sont présents (leurs valeurs vaut 1 dans $\vec{d}_a$ et $\vec{d}_b$) dans les deux documents.
- B : Le nombre de termes qui sont présents (leurs valeurs vaut 1 dans $\vec{d}_a$) dans d_a et qui sont absents (leurs valeurs vaut 0 dans $\vec{d}_b$) dans d_b.
- C : Le nombre de termes qui sont absents (leurs valeurs vaut 0 dans $\vec{d}_a$) dans d_a et qui sont présents (leurs valeurs vaut 1 dans $\vec{d}_b$) dans d_b.

$$Jaccard(\vec{d}_a, \vec{d}_b) = \frac{A}{B+C+A}$$

Étant donné la richesse du vocabulaire de notre corpus (dû principalement à la variété d'écriture de plusieurs termes et les différentes langues utilisées), le nombre de *traits*

utilisé pour représenter un document (tweet) est immense.

D'une autre part, un tweet ne contient pas beaucoup de termes[1] ce qui occasionne l'obtention de la valeur 0 pour la plupart des *traits* pour chaque tweet.

Aussi, en analysant les tweets nous avons constaté que la plupart des tweets ne contiennent pas de termes qui se répètent. En appliquant *Tf-Idf*, la valeur *Tf* vaut souvent 1 pour les termes présents dans le tweets. Étant donné ces deux caractéristiques, nous avons décidé de représenter les tweets par des vecteurs binaires en considérant seulement les termes dont la valeur égale à 1. Par conséquent, nous utilisons la mesure *Jaccard* pour comparer une paire de tweets.

5.5 Expansion des tweets

Les mesures de similarité utilisées se basent sur les termes en commun dans deux documents. Cependant, les termes, sémantiquement similaires ou écrits dans une forme différente (fautes d'orthographe, abréviation, ...), ne sont pas considérés comme des termes en commun. Par exemple, si un tweet tw_1 contient le terme `Ghnem` et un deuxième tweet tw_2 contient `Ghanim`, ces deux termes ne sont pas considérés des termes en commun même s'ils sont similaires.

Pour améliorer les résultats obtenus par les mesures de similarité afin de détecter les termes en commun entre deux documents même s'ils sont écrits d'une manière différente, nous avons décidé d'étendre les tweets par des termes sémantiquement similaires. Cette technique est couramment utilisée dans la recherche d'informations afin d'améliorer les résultats retournés par un système de recherche d'informations. Elle consiste à étendre la requête, utilisée pour trouver les documents, par d'autres termes reliés. Certaines méthodes utilisent des ressources externes telles que des thésaurus (p.ex. *WordNet*) pour enrichir la requête.

Dans notre travail, nous adoptons le même principe en enrichissant les tweets par d'autres termes. Cependant, le recours à des ressources externes prédéfinies est difficile étant donné le peu de ressources pour le dialecte. Un thésaurus contient un ensemble

[1] selon les statistiques de notre corpus, la taille moyenne d'un tweet $\cong$ 7 termes

fini de termes, mais le vocabulaire utilisé dans les médias sociaux est évolutif par le fait que les internautes produisent souvent des nouveaux termes (abréviation, convention, etc.) et parce que des nouvelles personnes et technologies apparaissent. Pour étendre les tweets, nous avons exploité les *clusters* de termes, sémantiquement similaires, obtenus par les méthodes de *regroupement* présentées dans le chapitre 4. Chaque terme (mot ou hashtags) est remplacé, le cas échéant, par le *cluster* de termes qui lui appartient. Il est possible qu'un terme n'appartienne à aucun *cluster*. Dans ce cas, aucune modification n'est effectuée (nous réécrivons le même terme).

Ci-dessous des échantillons de trois *clusters* de hashtags similaires :

GroupeAttounisia
#attounisia
#attounisiya
#attounisiyya
#attouniss
#ettounssia
#ettounseya
#attounsia
#ettounssiya
...

GroupeIslami
#islami
#islamic
#islamists
#islamophobie
#islamism
#islamisation
#islamiste
#islamist
...

GroupeGhanim
#ghanim
#ghenim
#islamists
#ghnaim
#ghnim
#ghoneim
#ghonem
#ghounim
...

Soit les tweets suivants représentés par les hashtags qui les contiennent :

Original1 : `#tunisie #attounisya #ttn`

Étendu1 : `#tunisie GroupeAttounsia #ttn`

Original2 : `#tunisie #islamisme #ghenim`

Étendu2 : `#tunisie GroupeIslami GroupeGhanim`

À la fin, nous obtiendrons des tweets qui contiennent plus d'informations ce qui permet aux mesures de similarités de détecter mieux les termes en commun entre une paire de tweets

5.5.1 Regroupement par algorithme incrémental

Dans cette section nous présentons l'algorithme de clustering appliqué pour regrouper les tweets similaires. Étant donné que le nombre d'évènements au cours d'une période

est inconnu, il est difficile de prévoir le nombre de clusters de tweets qui portent sur le même sujet (évènement).

Rappelons que dans cette tâche nous supposons que chaque cluster de tweets porte sur un évènement. Pour cette raison, nous avons utilisé un algorithme de clustering incrémental, qui n'impose pas la prédétermination, du nombre de clusters à obtenir. L'algorithme 3 montre comment nous avons procédé pour regrouper les tweets similaires.

Algorithm 3 Regroupement des tweets par un algorithme incrémental

Entrées : TW (l'ensemble de tweets à regrouper)
tw_0 forme le premier *cluster*
for chaque $tw_i \in TW$ **do**
  Déterminer le cluster C^* le plus approprié à tw_i
  if $C^* \neq$ nil **then**
    ajouter tw_i à C^*
  else {il n'y a aucun cluster adéquat pour tw_i}
    créer un nouveau *cluster* qui contient initialement tw_i
    $|C| \leftarrow |C| + 1$
  end if
end for

Pour de déterminer le cluster C^* adéquat pour tw_i, nous avons appliqué la formule suivante :

$$C^* = \arg\max_{C_j \in C} \frac{1}{|C_j|} \sum_{tw \in C_j} Similarity(tw_i, tw)$$

Où :

- C_j est le $j^{\text{ème}}$ cluster déjà créé.
- $Similarity(tw_i, tw)$ calcule la similarité, entre tw_i et tous les tweets de C_j, en utilisant l'indice de Jaccard, défini dans la section 5.I.
- $|C_j|$ est le nombre de tweets dans C_j.

Pour chaque *cluster* C_j existant, nous comparons tous ses tweets avec tw_i. C_j est

considéré comme candidat pour tw_i si la moyenne de la similarité ($\frac{1}{|C_j|}\sum_{tw\in C_j} Similarity(tw_i,tw)$) de tw_i et les tweets de C_j dépasse un seuil ε. ε est un seuil prédéfini manuellement.

Le candidat, qui a la valeur $\frac{1}{|C_j|}\sum_{tw\in C_j} Similarity(tw_i,tw)$ la plus élevée, est le *cluster* auquel tw_i va appartenir. Si l'ensemble de candidats est vide, nous créons un nouveau *cluster* contenant tw_i.

Avec cet algorithme, il n'est pas indispensable que tous les tweets appartiennent à des *clusters*. Il est entièrement possible que certains tweets soient distincts. Ces tweets devraient se trouver seuls dans leurs *clusters*.

Chaque *cluster* obtenu devrait être composé des tweets discutant le même sujet. Cependant, les *clusters* contiennent des tweets écrits à des dates différentes. Soit le *cluster* $C_{VisiteGhanim}$ qui contient des tweets, portant sur l'évènement de la visite de `Wajdi Ghonim`, écrits tout au long de la période.

La fréquence quotidienne d'un sujet correspond au nombre de ses tweets créés quotidiennement.

5.6 Détection des dates saillantes

À l'étape précédente, nous avons regroupé les termes similaires. Nous avons considéré que chaque *cluster* obtenu représente un évènement. Par la suite, nous avons calculé la fréquence quotidienne de chaque *cluster*. Le nombre d'occurrences d'un *cluster* g (dans un jour j) est égal au nombre de tweets (dans un jour j) qui contiennent au moins un élément de g. Nous avons représenté chaque évènement (*cluster*) s par une suite de nombres :

$$f(s) = f_1(s), f_2(s) \dots f_t(s)$$

où $f_i(s)$ est le nombre quotidien d'occurrences d'un évènement dans la journée i dans une période τ.

Notre objectif dans ce travail est de détecter les évènements majeurs, et aussi de déterminer les dates où ces évènements ont eu lieu. Pour répondre à cette tâche, nous avons utilisé la méthode proposée par Palshikar (2009) permettant de détecter les dates saillantes. Cette méthode permet de détecter les pics dans une série temporelle. Elle

prend comme entrée une série $f(s)$ et retourne les indices I qui correspondent aux pics. Dans notre cas les indices sont les jours.

Soit S la fonction qui permet de détecter les pics dans une période donnée :

$$S_i(k, f(s)) = \frac{\max_{1 \leq j \leq k}(f_i(s) - f_{i-j}(s)) + \max_{1 \leq j \leq k}(f_i(s) - f_{i+j}(s))}{2}$$

où k est un entier positif indiquant le nombre de voisins à considérer autour de chaque point $f_i(s)$ dans τ, les valeurs les plus appropriées de k étant compris entre 3 et 5. Pour $f_i(s) \in f(s)$, S_i calcule la moyenne de la différence maximale entre les valeurs de k voisins à gauche et à droite de $i^{\text{ème}}$ élément. $f_i(s)$ est considéré comme un pic si :

$$S_i > 0 \textbf{ et } (S_i - mean) > (h * stdv)$$

où *mean* et *STDV* sont respectivement la moyenne et l'écart type des valeurs positives de S_i (ignorant donc la valeur 0). h est une constante prédéfinie par l'utilisateur. Dans nos expériences, h a été fixée à 1 et k à 3.

5.7 Conclusion

Dans ce chapitre, nous avons présenté nos méthodes permettant de déterminer les différents sujets présentés dans notre corpus. Au début, nous avons considéré que chaque code *Soundex* représente un sujet. Par la suite, nous avons regroupé les tweets similaires en utilisant un algorithme incrémental où chaque *cluster* réfère à un sujet.

À la fin, nous avons présenté la méthode que nous avons utilisé pour déterminer les dates saillantes de chaque sujet.

Au chapitre suivant, nous présentons les résultats obtenus par nos méthodes de regroupement de termes. Ces résultats constituent un préparatif pour appliquer les méthodes présentées dans le présent chapitre.

CHAPITRE 6

EXPÉRIMENTATIONS : REGROUPEMENT DES HASHTAGS

Dans les chapitres précédents, nous avons présenté les méthodes que nous avons appliquées afin de regrouper respectivement les termes (hashtags, mots) et les tweets similaires. Certaines d'entre elles ont été modifiées pour qu'elles puissent supporter les données utilisées dans ce travail.

Dans ce chapitre, nous présentons les expérimentations effectuées pour regrouper les hashtags similaires. Tout d'abord, nous allons présenter les résultats obtenus par les méthodes de normalisation, notamment le *Soundex* et la translittération. Par la suite, nous montrons les résultats obtenus des variantes de notre algorithme de *regroupement CoDBscan* : $CoDBscan_{npmi}$, $CoDBscan_{npmiWithSndx}$, $CoDBscan_{hyper}$ et $CoDBscan_{hyperWithSndx}$.

6.1 Expérimentations : regroupement des termes similaires

6.1.1 *Soundex*

Bien que cette technique soit simple, elle nous a permis de regrouper un nombre important des hashtags semblables, écrits de différentes façons à cause de fautes d'orthographe commises par les utilisateurs.

Nous n'avons appliqué cet algorithme que sur les hashtags écrits en alphabet latin. Le nombre de hashtags différents trouvés dans notre corpus est 12 218 (11 693 écrits en alphabet latin et 525 en alphabet arabe). En appliquant le *Soundex* sur les hashtags écrits en alphabet latin, nous avons obtenu 7 810 *clusters*. Un *cluster* contient les hashtags qui ont le même code *Soundex*.

Le *Soundex* a réussi à regrouper correctement des dizaines de hashtags. Étant donné l'absence de données de référence, il est difficile de calculer le rappel des résultats retournés par notre *Soundex*. Il est toutefois possible de calculer la précision. Les formules de rappel et précision sont définies comme suit :

$$Précision = \frac{\text{Nombre d'éléments correctement attribués au } \textit{cluster } g}{\text{Nombre total d'éléments sélectionnés}}$$

$$Rappel = \frac{\text{Nombre d'éléments correctement attribués au } \textit{cluster } g}{\text{Nombre total d'éléments pertinents, pour le } \textit{cluster } g\text{, dans le corpus}}$$

Nous avons calculé, manuellement, la précision pour les 15 plus importants *clusters* (ceux qui contiennent plus de hashtags). Le tableau 6.I montre les résultats obtenus. Le *Soundex* a réalisé une précision de 100 % pour plusieurs *clusters*. La moyenne de précision, pour ces 15 *clusters* est de 96 %.

Comme nous l'avons indiqué au chapitre 4, les utilisateurs inventent souvent des hashtags sous forme de dates. Ce type de hashtags n'est pas considéré par notre *Soundex*. Cependant, nous avons utilisé une expression régulière permettant de détecter ces hashtags. En effet, les hashtags indiquant des dates ne sont pas du même format (p.ex JJMMAAAA, JJMAA,...), en outre il n'est pas rare de trouver des fautes d'orthographe dans les dates, notamment dans les noms des mois. Nous avons regroupé 99 hashtags qui réfèrent à des dates en 58 *clusters* avec une précision égale à 100 %. Le tableau 6.III montre les 10 *clusters* les plus importants de hashtags réfèrant à des dates.

Le 9 avril réfère à la Journée des Martyrs en Tunisie. Le 9 avril 2012 des milliers de Tunisiens sont manifestés dans plusieurs régions en Tunisie et où il y a eu des affrontements violents entre manifestants et forces de l'ordre. Cet évènement a stimulé l'intérêt de Tunisiens, nous avons constaté que plusieurs tweets en parlent. Le 25 février 2012, une immense manifestation est organisée en Tunisie par l'Union générale des travailleurs tunisiens (UGTT). Le 20 Mars est la fête d'indépendance de la Tunisie.

Tableau 6.I – Les 15 *clusters* contenant le plus grand nombre de hashtags, après l'application de *Soundex*. **NE** : Nombre d'éléments ; **EH** : Exemple de hashtags ; **NHMR** : Nombre de hashtags mal regroupés ; **EHMR** : Exemple de hashtags mal regroupé

Soundex	**NE**	**EH**	**NHMR**	**EHMR**	**Précision**
RVLTN	34	*rvolution ; revoliton2 ; revoltion ; revolution ; rrevolution ; révolution ; revolution2*	1	*reveiltunisien*	98%
TNSY0	25	*etounsya ; ettounisia ; ettounisiya ; ettounisseya ; ettounissiya ; ettounissya ; ettounisya ; ettounissia*	0		100%
TFH00	24	*tfih ; tfiih ; tfiiiiiiiiiiih ; tfooh ; tfooooouh tfooouuuhhh ; tfooouuuuuuh ; tfouh ; tfouhh ; tfuuuuh*	1	*tafé́ha*	96%
SLFST	23	*salafist ; salfistes ; salifistes ; sallafiste ; slafiste ; slafistes*	0		100%
TNS00	21	*ttunisie ; tunesie ; tuniise ; tunis tunisa ; tunise ; tunisi ; tunisiie ; tunisis ; tunisisa ; tunisusa ; tunsie ; tunsiie*	8	*eautunisie ; ettounisa ; tennis ; touensa ;*	62%
NHDH0	19	*enahdah ; enahdha ; enanahdha ; enanhdha ; enhahdha ; nahadha ; nahddha ; nhadha ; nnehdha*	0		100%
3TNSY	18	*attouneseya ; attounessia ; attouniseya ; attounisia ; attounsiya ; attounsseya ; attounssia ; attounsya*	0		100%
MNSTR	18	*ministere ; ministre ; ministère ; ministry ;*	9	*monsatir ; monstre ;*	100%
GHNM0	15	*ghanaim ; ghanem ; ghanim ; ghanmi ; ghenim ; ghinim ; ghnaim ; ghnim*	0		100%
TWTPR	15	*tweetprecedent ; tweeteprécédent twittprecedent ; tweetprécédents*	1	*tweetprémonitoire*	93%
THN00	14	*et7ine ; ettahana ; t7ine ; t7inn*	0		100%
TNSYT	14	*ettouniseyatv ; ettounissiyatv ; ettounseyyatv ; ettounsiatv ; ettounsieyatv*	1	*tunisiatheplacetobe*	93%
TKBR0	13	*takbiiiiiiiiiiiiir ; takeabeer ; takebeeeer ; takbire*	0		100%
BSHLK	12	*bouchelka ; bouchlaka ; bouchléka ; boushlek*	0		100%
GHNSH	12	*ghannoucha ; ghannouchi ; ghannoushi ; ghannushi ; ghanouchi*	0		100%
Moyenne					96 %

Tableau 6.III – Des dates utilisées sous forme de hashtags

Date	**Nb. Hashtags**	***Hashtags***
9 avril	9	*#9avril*, *#9april*, *#9avil*, *#9avirl*, *#9avr* *#9avri*, *#9avril2012*, *#9avrils*, *#april9*
25 février	6	*#25feb*, *#25fev*, *#25fevrier201*, *#25fevrier2012*, *#feb25*, *#fev25*
20 mars	4	*#20mars*, *#20mars1956*, *#20mars2012*, *#march20*
13 mars	3	*#13march*, *#13mars*, *#mar13*
20 fevrier	3	*#20feb*, *#20fev*, *#feb20*
25 mars	3	*#25mars*, *#25mars2012*, *#mar25*
8 avril	3	*#8april, #8avril, #8avril2010*
8 mars	3	*#8march, #8mars,#8mars2012*
14 fevrier	2	*#14feb, #feb14*

Discussion :

Le *Soundex* a regroupé efficacement les hashtags. Cependant, La présence de certains hashtags (*#eautunisie*) contenant des mots référant à la Tunisie (tunisie, tounes, etc.) et d'autres hashtags liés à des sujets, dont les noms sont dérivés de terme '*tunisie*' (la chaine de télévision *Ettounisia*), ont mené à la même prononciation ce qui explique la faible précision obtenu dans le *cluster* 5.

Aussi, *Soundex* éprouve des difficultés avec les hashtags de petite taille et ceux dont les codes contiennent plus de deux zéros (voir tableau 6.IV). Ceci est expliqué principalement par l'absence d'un nombre suffisant de caractères permettant de distinguer les chaines de caractères. Nous appelons ces codes *Noisy Soundex*. Le tableau 6.IV montre un échantillon de *Noisy Soundex*. Nous avons décidé de ne pas appliquer le *Soundex* pour ces hashtags, le code de chacun de ces hashtags est le hashtag lui-même.

Tableau 6.IV – Type de hashtags qui rencontrent des problèmes*Soundex*

Soundex	**hashtags**
L0000	*#el*, *#l*, *#lolilol*, *#lol* *#looooooooooooool*, *#lulu*, ...
K0000	*#ca*, *#cc*, *#coca*, *#uk* *#ce*, *#q*, *#ge*, *#ku* ...
S0000	*#es*, *#esa*, *#ouais*, *#sousse*, *#suisse*, ...

Dans le tableau 6.I, *TNSY0* et *3TNSY* semblent former un même *cluster*. Nous avons vérifié dans notre corpus et nous avons constaté que les hashtags trouvés dans ces deux *clusters* portent sur deux sujets différents. Le *cluster TNSY0* contient des hashtags référant à une chaine de télévision intitulée à un magazine intitulé '*ettounsia TV*' tandis que le *cluster 3TNSY* contient des hashtags référant à un magazine intitulé '*attounissia*' [1].

Toutefois, les hashtags trouvés dans les *clusters TNSY0* et *TNSY0* devraient être dans un même *cluster* puisque *TNSYT* et *TNSY0* contiennent des hashtags discutant du même sujet, la chaine de télévision *ettounsia TV* [2]. Cette erreur est expliquée par l'ajout de la chaine de caractères '*TV*' aux hashtags trouvés dans *TNSYT*. Bien que *Soundex* montre une haute précision, nous avons constaté qu'il trouve parfois des difficultés pour reconnaitre certains *clusters* similaires tels que*TNSY0* et *TNSYT*.

6.1.2 Translittération

Le *soundex* que nous avons implémenté pour regrouper les hashtags qui ont une prononciation similaire fonctionne avec les hashtags écrits en alphabet latin. Cependant, notre corpus contient également des hashtags écrits en alphabet arabe. Afin de tenir compte de tels hashtags, nous avons appliqué un algorithme de translittération qui sert à trouver pour chaque hashtag écrit en alphabet arabe son équivalent latin (voir chapitre 4). La translittération n'est pas une tâche facile car certaines lettres ont plus qu'un seul équivalent (voir tableau 4.IV) et les textes dans les médias sociaux ne contiennent pas de signes diacritiques qui aident généralement à déterminer les voyelles latines correspon-

[1] http://www.attounissia.com.tn/
[2] http://www.ettounsiya.tv/

dantes. Par exemple, le mot غنيم (ghanim) qui n'a pas de signes diacritiques peut avoir plusieurs translittérations. Le tableau 6.VI montre des exemples de translittérations qui varient selon les signes diacritiques utilisés.

Tableau 6.V – Exemples de signes diacritiques et leurs équivalents

Équivalents possibles	**signe diacritique**	**Type de signe**	**Exemple**
a	ـَ	*fatha*	بَ
ou, o, au	ـُ	*dhama*	بُ
i, y	ـِ	*kasra*	بِ

Tableau 6.VI – Exemples de translittérations pour un terme en alphabet arabe avec signes diacritiques

Translittérations possible	**Terme arabe**
ghnim (sans signes diacritiques)	غنيم
ghanim, *ghaneem*	غَنِيمْ
ghonim, *ghounim*, *ghoneem*, *ghouneem*	غُنِيمْ
ghanaym, *ghanayem*	غَنَيمْ
ghonaym, *ghonayem*	غُنَيمْ

Au début, nous avons généré pour chaque hashtag h_a toutes les translittérations possibles et nous gardons seulement les translittérations identiques à des hashtags trouvés dans H_L. H_L est l'ensemble de hashtags écrits initialement en latin. Ensuite nous choisissons parmi les translittérations retenues (*TR*) la bonne selon certains critères : le nombre d'hyperliens partagés entre h_a et les hashtags trouvés dans *TR* et le nombre de fois où h_a

et les hashtags de *TR* apparaissent dans le même jour.

Cependant, l'ajout de toutes les voyelles possibles entraine l'obtention d'un grand nombre de translittérations possibles. Par exemple pour le terme غنيم il y a 252 translittérations possibles = {*ghanim*, *ghounem*, *ghenim*, *ghonim*, *ghunim*, *ghanym* ... }. Puisque nous allons appliquer le *Soundex* sur la translittération obtenue afin de l'affecter au *cluster* des hashtags qui lui sont similaires et que le *Soundex* ne considère pas les voyelles, il est inutile d'ajouter les voyelles qui provoquent un grand nombre des translittérations possibles.

Pour cette raison nous avons décidé de ne pas ajouter les voyelles dans les combinaisons possibles et de considérer les codes *Soundex*, des nouvelles translittérations possibles, comme candidats.

En appliquant ces changements, le nombre de translittérations pour le terme غنيم se limite à deux : {*ghnim*, *ghnym*} avec le même code *Soundex* = *GHNM0* qui existe déjà dans la liste des codes *Soundex* obtenue avec H_L. Comme il n'y a qu'un seul candidat (GHNM0), nous supposons que غنيم fait partie du *cluster GHNM0*, ce qui est correct (voir tableau 6.I).

Le tableau 6.VII montre des exemples de termes écrits en alphabet arabe et les codes *Soundex* associés à ses translittérations possibles :

- Le hashtag منوبة n'a qu'un seul code Soundex comme candidat qui est *MNB0*. Donc منوبة $\in$ *MNB0*. Le hashtag منوبة est le mot arabe de la ville *mannouba* qui appartient déjà à *MNB0*.

- La liste des candidats pour le hashtag مرزوقي contient trois codes *Soundex* {*MRSK0*, *MRZ90*, *MRZK0*}. Cependant, مرزوقي ne partage aucun hyperlien avec les trois codes *Soundex*, par contre il apparait plus avec *MRZK0* dans le même jour. Alors مرزوقي $\in$ *MRZK0*. مرزوقي est le nom du président de la Tunisie, *MRZK0* regroupe les hashtags qui réfèrent au nom du président en français (*Marzouki*, *Mrzouki* ...)

- Le hashtag سلفي a également comme candidats = {*SLFY0*, *SLF00*, *CLV00*}. سلفي partage 2 hyperliens avec les hashtags trouvés dans *SLFY0*, tandis qu'il ne partage

aucun hyperlien avec *SLF00* et *CLV00*. سلفي est le mot arabe de terme *salafy* qui a le code *Soundex* = *SLFY0*.

Tableau 6.VII – Exemples de hashtags écrits en alphabet arabe avec les codes *Soundex* correspondants aux translittérations possibles . **HAA** : Hashtags en alphabet arabe ; **NTP** : nombre de translittérations possibles ; **CS dans le corpus** :codes*Soundex* correspondant aux translittérations possibles présentes dans le corpus ;

HAA	NTP	CS dans le corpus
منوبة	8	*MNB00*
مرزوقي	16	*MRSK0*, *MRZ90*, *MRZK0*
بحرين	8	*BHRN0*, *BHRYN*
سلفي	24	*SLFY0*, *SLF00*, *CLV00*
طحين	4	*THYN0*, *THN00*

Pour les hashtags qui réfèrent à des dates et qui sont écrits en alphabet arabe, nous avons utilisé également une expression régulière permettant de les détecter. Ensuite, nous déterminons leurs équivalents en latin. Par exemple, 9افريل et 20مارس vont être regroupés avec les hashtags référant, respectivement, aux dates 9 avril et 20 mars.

La translittération nous a permis d'enrichir les *clusters* (contiennent des hashtags écrits initialement en alphabets latins) obtenus précédemment à l'aide de *Soundex*. L'utilisation des informations (hyperliens, date de création) trouvées dans les tweets nous a permis d'affecter d'une manière efficace un hashtag écrit en alphabet arabe au bon *cluster*. Le tableau 6.VIII présente les statistiques après les tâches de normalisation effectuées ci-dessus. À la fin, nous avons obtenu 9033 différents codes *Soundex*.

6.1.3 Clustering *DBscan* & *PMI*

Dans les sections précédentes, nous avons présenté des méthodes déterministes afin de regrouper les termes avec une prononciation similaire. Bien que ces algorithmes aient

Tableau 6.VIII – Statistiques après les tâches de normalisations

Nbr. de hashtags	12 218
Nbr. de hashtags écrits en latins	11 693
Nbr. de *clusters Soundex* initiaux	7 810
Nbr. de *clusters Soundex* après la normalisation des dates	7 781
Nbr. de *clusters Soundex* en disjoignant les *Noisy Soundex*	8 750
Nbr. de *clusters Soundex* après la romanisation	9 033

une bonne précision, ils sont incapables de regrouper les termes qui n'ont pas une prononciation similaire. Rappelons qu'un évènement est représenté par plusieurs termes. Par exemple, l'évènement de la mise en berne du drapeau tunisien (discuté dans les chapitres précédents) peut être représenté par ses termes : drapeau, manouba (la place où l'évènement a eu lieu), salafiste (étudiant qui a fait l'acte), étudiante (qui a essayé d'empêcher l'étudiant salafiste). *Soundex* regroupe les hashtags référant au nom de famille de '*Wajdi Ghonim*' mais il n'ajoute pas à ce *cluster* les hashtags, qui commencent et/ou contiennent le prénom du prédicateur, tels que *#Wajdi, #Wagdi* ...

Dans les sections suivantes, nous montrons quelques résultats que nous avons obtenus en appliquant des méthodes statistiques afin de regrouper les termes similaires.

6.1.3.1 *Pointwise mutual information (*PMI*)*

Pour regrouper les termes d'une manière plus efficace, nous avons eu recours à des algorithmes de *regroupement*. Cependant, le fonctionnement de ces algorithmes nécessite une distance pour déterminer la similarité entre les éléments (termes) à regrouper. Cette distance devrait refléter la relation entre deux termes. Pour ce faire, nous avons utilisé la mesure de *PMI* basée sur la cooccurrence entre termes. Nous avons constaté que la valeur de *PMI* reflète bien le degré d'association entre les termes. Le tableau 6.IX montre un exemple de hashtags avec ceux qui ont une grande valeur de *PMI* :

- Le hashtag `#abbasi` réfère à *Houcine Abbasi* est le Secrétaire général de l'UGTT (*#ugtt*), qui est le successeur de *Abdessalem Jrad* (*#jrad*). *Abid Briki* (*#biki*) ex-secrétaire général adjoint de l'*UGTT*[3].

[3] `http://fr.wikipedia.org/wiki/Houcine_Abassi`

Tableau 6.IX – Exemples de valeurs de *PMI* entre des hashtags représentant un même sujet (évènement)

Hashtag 1	**Hashtag 2**	**valeur *PMI***
#abbasi	*#biki*	15.46
	#jrad	11.14
	#ugtt	5.71
#9avril2012	*#poltique*	12.22
	#blessée	12.22
	#agressions	11.22
	#milices	8.22
	#acab	4.16
#9avil	*#al9assas*	14.88
	#dictature	6.95
#khaola	*#manouba*	7.33
#khaoula_rachidi	*#drapeau*	9.63
#khaoularachidi	*#lesfemmes2monpays*	14.88
	#salafistes	5.64
#attounssia	*#gercke*	12.29
	#nasreddinebensaida	11.29
	#khdira	10.97
	#free	8.90
	#khedira	8.34
	#censorship	8.34

- Le 9 avril 2012 (*#9avril*), une grande manifestation a eu lieu devant le ministère de l'Intérieur, la police (*#acab*) et les milices (*#milices*) du parti au pouvoir l'ont dispersé violemment et ils ont agressé (*#agressions*) les manifestants. Plusieurs personnes ont été blessées (*#blessée*) dont des politiciens (*#politique*) qui ont participé à cette manifestation. L'un de politicien agressé est *Ibrahom el Gassas* (*#al9assas*)[4].

- La mise en berne du drapeau (*#drapeau* tunisien par un étudiant *salafiste* (*#salafiste*) dans le bâtiment de la faculté des lettres, des arts et des humanités de *Manouba* (*#manouba*). Une étudiante tunisienne *Khaoula Rachdi* (*#khaoula, #khaoularachidi, #khaoula_rachidi*), remet le drapeau tunisien à sa place. Plusieurs Tuni-

[4] http://tempsreel.nouvelobs.com/monde/20120409.OBS5778/tunis-la-police-diperse-violemment-une-manifestation-interdite.html

siens ont apprécié (*#lesfemmes2monpays*) l'acte de l'étudiante[5].

- *Nasreddine Ben Saïda* (*#nasreddinebensaida*), directeur du journal Attounsia (*#attounssia*) a été arrêté et emprisonné, le 15 février 2012, suite à une publication dans son journal d'une photographie du footballeur *Sami Khedira* (#khedira) avec son épouse *Lena Gercke* (*#gercke*) qui a été dénudée. Plusieurs ont trouvé que cette arrestation est une censure et ils ont demandé la libération immédiate du journaliste[6].

L'application de *Soundex* a montré une excellente précision (voir tableau 6.I) pour regrouper les hashtags avec la même prononciation ou une variation d'écriture. Pour cette raison, nous avons décidé de remplacer les hashtags par leurs codes *Soundex* ce qui permet d'augmenter la probabilité de cooccurrence entre les hashtags. Par exemple, le hashtag *#khaoula_rachidi* cooccurre maintenant avec les hashtags avec codes *Soundex SLFST*, puisqu'il a le même code *Soundex* (*KHLRS*) que *#khaoularachidi* et que ce dernier a cooccurré avec le hashtag *#salafistes*. Le tableau 6.X montre des exemples les valeurs de *PMI* obtenues entre le code *Soundex GHNM0* (*#ghnim, #ghoneim, #ghanim* ...), référant au prédicateur *Wajdi Ghonim*, et d'autres hashtags. Le prénom du prédicateur se prononce en arabe standard '*Wajdi*' mais en dialecte égyptien '*Wagdi*'.

Grâce à *Soundex* toutes les variations du nom de prédicateur *Ghonim* ont une relation avec les hashtags qui appartiennent aux codes *Soundex* mentionnés dans le tableau 6.X. À titre d'exemple, les hashtags de *cluster SHKGH* cooccurrent seulement avec le hashtags *#ghanim* de *cluster GHNM0* qui contient 15 variations du nom de prédicateur. Aussi, seulement le hashtag *ghoneim* $\in$ *GHNM0* cooccurre avec les hashtags appartenant à *ATFD0* [7]

6.1.3.2 *Normalized Pointwise Mutual Information (*NPMI*)*

Dans ce travail, nous considérons la valeur de *PMI* comme distance utilisée par notre algorithme de *regroupement*, *DBscan* qui nécessite une distance seuil ε. Cepen-

[5] `http://www.lapresse.tn/02032013/61113/lannee-2012.html`
[6] `http://oua.be/1e5u`
[7] `http://femmesdemocrates.org/`

Tableau 6.X – Exemples de valeurs de *PMI* entre le code*Soundex GHNM0* (*#ghnim, #ghoneim, #ghanim* ...) et d'autres.

Code ***Soundex***	**Explications**	**valeur *PMI***
SHKGH	Hashtags qui commencent par *'sheikh'* et qui réfèrent à *ghanim*. Exemple (*#sheikhghanim, #sheikh_ghnim*).	6.98
WGD00	Hashtags qui réfèrent à son prénom. Exemple (*#wagdi*).	6.98
WJD00	Hashtags qui réfèrent à son prénom. Exemple (*#wajdi*)	6.98
ATFD0	Hashtags réfère à l'***A**ssociation **f**emmes **d**émocrates **t**unisiennes* qui a dénoncé la visite de *Wajdi Ghonim*	6.40
Kobba	Hashtag réfère à la place où *Wajdi Ghonim* a fait son premier séminaire	5.11
FGM00	Hashtags qui réfèrent à l'abréviation *Female Genital Mutilation*	5.66
XSN00	Hashtags qui réfèrent à l'excision. Exemple : *#excision, #exision*	4.83

dant, l'intervalle des valeurs de *PMI* est préalablement inconnu puisqu'il dépend de caractéristiques du corpus. Ce qui complique le choix de ε. Pour cela, nous avons normalisé les valeurs de *PMI*, obtenues antérieurement, en utilisant la méthode proposée par Bouma (2009) :

$$NPMI(e_i, e_j) = \log(\frac{P(e_i \& e_j)}{P(e_i)P(e_j)}) / -\log P(e_i \& e_j) = PMI / -\log P(e_i \& e_j)$$

P(e_i & e_j) est la probabilité que e_i et e_j apparaissent ensemble. La valeur de *NPMI* est limitée par [-1,1]. $NPMI(e_i, e_j)$ égale à 1 lorsque e_i et e_j sont entièrement dépendants, -1 sinon. Par conséquent, il est plus facile de déterminer ε. Le tableau 6.XI montre les valeurs de *NPMI* entre les hashtags présentés dans le tableau 6.IX.

Parmi les 12 218 hashtags, il y n'a que 9 793 hashtags qui ont des valeurs numériques définies de *NPMI*. Les autres hashtags non considérés (2 425) correspondent à des valeurs *NPMI* infinies, car ils ne cooccurrent avec aucun autre hashtag ou cooccurrent avec des hashtags que nous avons déjà supprimés (p.ex Tunisie). Les hashtags non considérés

correspondent à des hashtags moins fréquents ou rares (on a trouvé que environ 96 % de ces hashtags ont été observés au plus 3 fois).

Tableau 6.XI – Exemples de valeurs de *NPMI* entre des hashtags représentant un même sujet (évènement)

Hashtag 1	**Hashtag 2**	**valeur *PMI***	**valeur *NPMI***
#abbasi	*#biki*	15.46	0.93
	#jrad	11.14	0.67
	#ugtt	5.71	0.34
#9avril2012	*#poltique*	12.22	0.84
	#blessée	12.22	0.74
	#agressions	11.22	0.68
	#milices	8.22	0.49
	#acab	4.16	0.27
#9avil	*#al9assas*	14.88	0.96
	#dictature	6.95	0.42
#khaola	*#manouba*	7.33	0.44
#khaoula_rachidi	*#drapeau*	9.63	0.58
#khaoularachidi	*#lesfemmes2monpays*	14.88	0.90
	#salafistes	5.64	0.34
#attounssia	*#gercke*	12.29	0.74
	#nasreddinebensaida	11.29	0.68
	#khdira	10.97	0.66
	#free	8.90	0.54
	#khedira	8.34	0.50
	#censorship	8.34	0.50

6.1.3.3 Application de $DBscan_{NPMI}$

Comme mentionné au chapitre 3, nous avons appliqué l'algorithme *DBscan* pour regrouper les hashtags sémantiquement similaires en nous basant sur leurs valeurs *NPMI*.

Au début, nous avons appliqué la version standard de *DBscan* (voir section 1.3.2). Cette version ne tient pas compte de la fonction $Cohesion(C,e)$ qui vérifie la cohérence de l'élément *e* avec ceux trouvés dans le *cluster C*. Cette méthode conduit à l'apparition d'un *cluster* immense composé de 6 639 hashtags. La majorité des hashtags dans ce *cluster* ne sont pas sémantiquement similaires dû au phénomène *voisins-des-voisins* qui fusionne, d'une manière récursive, le *cluster* (formé initialement) avec voisins de ces

éléments.

Le tableau 6.XII montre les plus importants *clusters* obtenus en appliquant la version standard de *DBscan* avec $\varepsilon = 0.5$ et *ptsMin* = 2, nous avons obtenu 213 *clusters*.

Tableau 6.XII – Les *clusters* les plus importants obtenus par $DBscan_{npmi}$ standard ; $\varepsilon = 0.5$ *ptsMin*=2 ; ***ID* cluster** est l'identifiant du *cluster*

***ID* cluster**	***Taille***
C0	6639
C2	193
C51	13
C67	9
C33	8

Par la suite nous avons appliqué notre algorithme *CoDBscan* qui prend en considération la cohésion entre le hashtag et un *cluster* avant de l'ajouter. Avec les mêmes valeurs de ε =0.5 et *ptsMin* =2, nous avons ajouté une valeur $\lambda = 0.1$. *CoDBscan* a réussi à construire 1 483 *clusters*. Le tableau 6.XIII montre les plus importants *clusters* obtenus par *CoDBscan*.

Tableau 6.XIII – Les *clusters* les plus importants obtenus par $CoDBscan_{npmi}$; $\varepsilon = 0.5$; *MinPts* = 2 ; λ = 0.1 ; ***ID* cluster** est l'identifiant du *cluster*

***ID* cluster**	***Taille***
C1	67
C9	57
C19	50
C5	38
C49	36

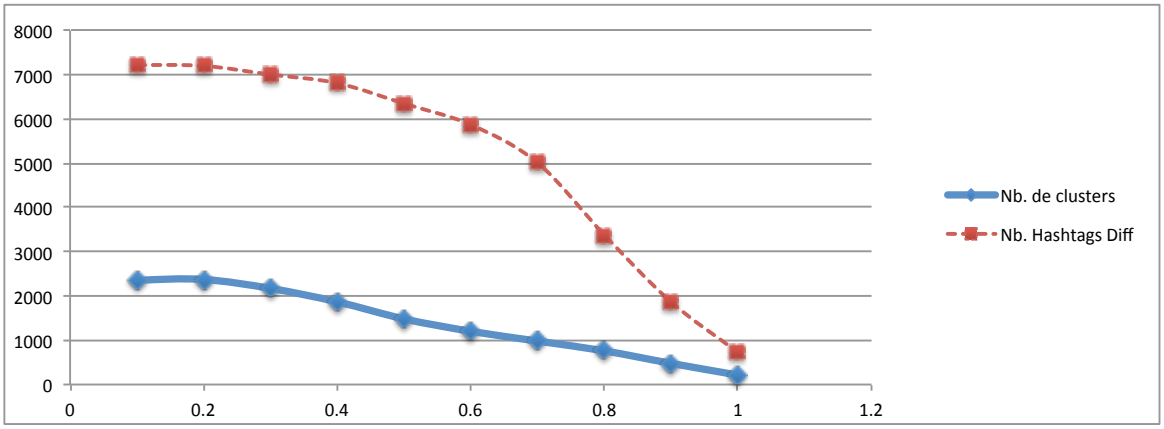
Contrairement au *Soundex*, notre $CoDBscan_{npmi}$[8] était capable de regrouper des hashtags de même contexte. À titre d'exemple, $CoDBscan_{npmi}$ a réussi à regrouper les hashtags {*#khaoula_rachidi, #drapeau, #flag, #femmelibre, #winhomrejelek* [9]*, #salafdemerde, #rachidi*}, ce qui était impossible avec *Soundex*. Même à l'aide des ressources préexistantes, on est incapable de déterminer la similarité sémantique entre *khaoula Rachidi* (*#khaoula_rachidi, #rachidi*) et le drapeau (*#drapeau, #flag*)

[8] l'application de *CoDBscan* en utilisant *NPMI* comme mesure de similarité

[9] *winhomrejelek* est la translittération de mots tunisiens qui signifient *'où sont les homme'*

La figure 6.1 montre la variation du nombre de hashtags différents regroupés ainsi que le nombre *clusters* obtenus en fonction de ε. Le nombre de *clusters* diminue quand la valeur de ε augmente. Il n'est pas évident que tous les hashtags seront affectés à des *clusters*. Si la valeur de ε est petite, il y aura plus de chance d'avoir des hashtags dont le *NPMI* dépasse ε. Ceci explique un plus grand nombre de *clusters* avec une petite valeur ε qu'avec une grande. Avec $\varepsilon < 0.9$ *CoDBscan* ne regroupe que quelques centaines de hashtags parmi les 9 793. Les autres hashtags sont restés seuls puisqu'ils n'ont aucune relation avec $NPMI > 0.9$

Figure 6.1 – La variation du nombre hashtags différents regroupés et le nombre *clusters* en fonction de ε



Le tableau 6.XVI montre, pour différentes combinaisons de ε et λ, le nombre de hashtags différents qui appartiennent à des *clusters* parmi les 9 793 hashtags ainsi que le nombre de *clusters*. Un *cluster* doit contenir au moins deux hashtags.

Tableau 6.XIV – Nombre de différents hashtags et *clusters* obtenus à l'aide de $CoDBscan_{npmi}$. **NH** : le nombre de hashtags distincts regroupés ; **NG** : le nombre de *clusters* obtenus ;

		Valeur λ																			
		0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
		NH	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**
Valeur ε	0.1	7225	2375	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.2	7203	2375	7083	2673	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.3	7007	2173	6898	2467	6900	2644	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.4	6813	1869	6627	2184	6614	2297	6584	2411	-	-	-	-	-	-	-	-	-	-	-	-
	0.5	6351	1483	6203	1787	6205	1852	6134	1997	6120	2039	-	-	-	-	-	-	-	-	-	-
	0.6	5871	1203	5740	1469	5650	1607	5614	1688	5653	1724	5623	1739	-	-	-	-	-	-	-	-
	0.7	5011	982	4856	1172	4753	1291	4743	1364	4756	1407	4705	1388	4726	1411	-	-	-	-	-	-
	0.8	3381	761	3323	816	3242	861	3217	885	3206	926	3186	917	3198	949	3201	949	-	-	-	-
	0.9	1865	478	1854	493	1844	503	1826	519	1811	520	1808	526	1801	522	1815	528	1792	528	-	-
	1	726	213	726	213	726	213	726	213	726	213	726	213	726	213	726	213	726	213	726	213

Étant donné l'absence de données de référence, il est donc impossible d'évaluer automatiquement les résultats obtenus par *CoDbScan*. Pour cela, nous les avons évalués manuellement.

Comme ε est le paramètre principal qui sert à créer les *clusters*, nous avons décidé de calculer la précision en ne tenant compte que de la valeur de ε. Ceci simplifie notre évaluation manuelle car nous avons ainsi moins de résultats à évaluer. Le tableau 6.XV montre les précisions pour les différentes valeurs de ε avec une valeur fixe de λ.

L'évaluation de ces résultats a été faite par deux experts. Nous avons donné à chaque expert une moitié de résultats à évaluer. Nous leur avons demandé de reconnaître le sujet principal de chaque *cluster* puis de déterminer les mauvais hashtags dans le *cluster*. Selon les évaluations obtenues, nous avons calculé la précision du regroupement. D'abord, nous avons calculé la précision de chaque *cluster* puis la moyenne pour les différentes valeurs de ε.

Bien que nos experts soient des Tunisiens et au courant des évènements survenus en Tunisie. Ils ont été, parfois, obligés de se rafraichir la mémoire pour se rappeler certains détails sur un sujet en effectuant des recherches dans le web. Par exemple, l'un des experts avait reconnu l'évènement de drapeau à l'*Université de manouba*, mais il avait oublié le nom de l'étudiante (*#khaoula_rachidi, #rachidi*) qui a empêché l'étudiant salafiste à enlever le drapeau (*#drapeau, #flag*). Nous avons été obligé d'aider les experts à comprendre certains hashtags qui sont généralement des créations (p.ex abréviations) employés par les internautes. Pour ce faire, nous avons consulté les tweets où le hashtag est présent afin d'obtenir leur explication. Les résultats présentés dans le tableau 6.XV montrent que plus la valeur de ε est grande plus les résultats retournés par $CoDBscan_{npmi}$ sont précis. Une précision de 100 % est obtenue pour ε entre 0.9 et 1. Avec ces valeurs, les *clusters* contiennent que les hashtags qui cooccurrent toujours ensemble. Toutefois, $CoDBscan_{npmi}$ n'a réussi à regrouper que 726 hashtags ($\approx$ 5 % de nombre total de hashtags) soit une très faible valeur de rappel. Avec $\varepsilon \leq 0.5$, $CoDBscan_{npmi}$ a réussi à regrouper plus de hashtags différents tandis que la précision obtenue est plus faible qu'avec $\varepsilon \geq 0.6$.

Tableau 6.XV – Évaluation manuelle des résultats obtenus avec $CoDBscan_{npmi}$. **NH** : le nombre de hashtags dans le *cluster* ; **Pr** : la précision

	Valeur ε																			
	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
	NH	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**
1	74	0.88	77	0.77	76	0.69	73	0.79	67	0.79	55	0.80	40	0.90	30	1.00	14	1.00	7	1.00
2	70	0.84	75	0.73	64	0.76	71	0.85	57	0.91	45	0.82	33	1.00	29	1.00	13	1.00	7	1.00
3	63	0.76	74	0.78	62	0.82	65	0.85	50	0.88	36	0.89	30	0.93	23	1.00	12	1.00	7	1.00
4	62	0.71	65	0.69	57	0.87	41	0.85	38	0.76	33	1.00	28	1.00	21	1.00	11	1.00	6	1.00
5	60	0.68	58	0.81	52	0.60	41	0.73	36	0.89	33	0.94	27	0.89	20	0.75	11	1.00	6	1.00
6	54	0.72	51	0.65	43	0.81	41	0.76	35	0.91	30	1.00	26	0.77	18	1.00	11	1.00	6	1.00
7	50	0.72	51	0.92	42	0.68	40	0.75	33	1.00	30	1.00	26	0.96	18	1.00	11	1.00	5	1.00
8	45	0.64	46	0.80	40	0.95	40	0.73	33	0.85	29	0.76	25	0.84	18	0.78	10	1.00	5	1.00
9	43	0.70	42	0.71	38	0.83	39	0.92	33	0.97	28	0.79	25	0.92	18	1.00	10	1.00	5	1.00
10	42	0.62	41	0.71	36	0.94	37	0.70	32	0.66	27	0.89	24	1.00	15	0.87	9	1.00	5	1.00
11	41	0.88	37	0.78	36	0.86	36	0.69	32	1.00	25	0.88	21	0.86	14	0.86	9	1.00	5	1.00
12	39	0.72	37	0.68	35	0.69	34	0.91	32	0.97	25	0.92	21	1.00	14	1.00	9	1.00	5	1.00
13	37	0.84	36	0.78	32	0.81	33	1.00	31	1.00	24	1.00	21	1.00	13	0.85	9	1.00	5	1.00
14	36	0.69	35	0.69	31	0.74	33	0.97	30	0.77	24	1.00	20	0.75	13	1.00	8	1.00	5	1.00
15	35	0.86	33	1.00	31	0.80	30	0.83	29	1.00	24	1.00	20	0.95	12	1.00	8	1.00	5	1.00
16	35	0.77	33	0.76	30	1.00	29	1.00	29	0.66	24	0.96	20	1.00	12	1.00	8	1.00	5	1.00
17	35	0.54	33	0.91	30	1.00	29	1.00	29	0.66	23	0.91	20	1.00	12	1.00	8	1.00	4	1.00
18	34	1.00	32	1.00	29	0.90	29	0.69	29	0.72	22	0.95	20	1.00	12	1.00	8	1.00	4	1.00
19	33	0.91	32	1.00	29	0.90	28	1.00	28	0.82	20	0.85	19	1.00	12	1.00	8	1.00	4	1.00
20	32	0.84	31	1.00	29	1.00	28	0.86	28	0.89	20	1.00	19	1.00	11	1.00	8	1.00	4	1.00
Moyenne		0.77		0.81		0.83		0.84		0.86		**0.92**		**0.94**		**0.95**		1.00		1.00

Bien que le calcul du rappel soit difficile, nous avons observé qu'avec $\varepsilon \in [0.6, 0.8]$, on a pu créer des *clusters* avec une bonne précision. Tandis que avec $\varepsilon \in [0.9, 1]$ beaucoup de hashtags ont été rejetés.

Amélioration avec *Soundex* :

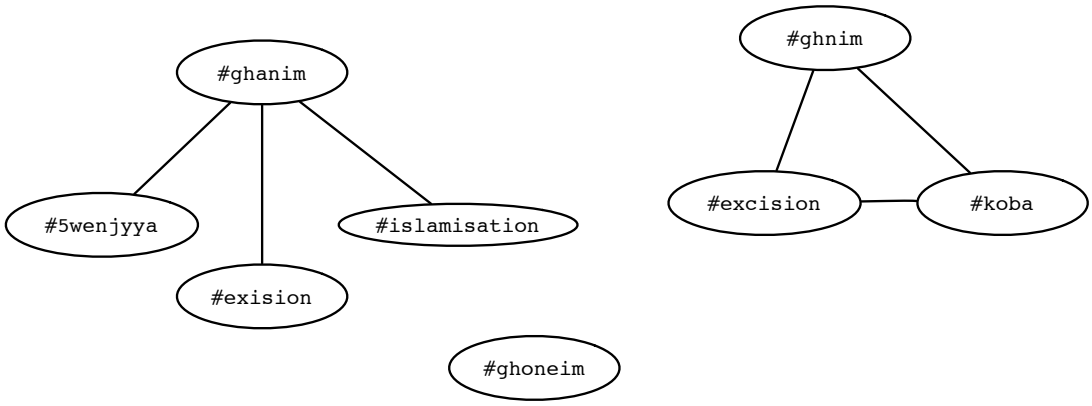
$CoDBscan_{npmi}$ était capable de regrouper des hashtags d'un même contexte. Toutefois, plusieurs hashtags ont été rejetés puisqu'ils n'ont pas de valeurs définies de *NPMI*. En outre, nous avons constaté que, pour toutes les valeurs de ε, il existe des hashtags qui devraient être ensemble dans un *cluster* mais qui se sont trouvés dans des *clusters* différents. Une des principales raisons derrière ce problème est la variation orthographique de la plupart des hashtags. La figure 6.2 montre un exemple de sous-graphes représentant la relation entre des hashtags. Deux hashtags h_i et h_j reliés cela signifie qu'ils cooccurrent. *#ghanim* cooccurre avec *#exision* [10], *#islamisation* et *#5wenjyya* [11] et n'a été jamais présent avec *#excision* et *#koba* qui cooccurrent avec *#ghnim* tandis que *#ghoneim* ne cooccurre avec aucun autre hashtag.

Par conséquent, les valeurs *NPMI* pour (*#ghanim*, *#koba*), (*#koba*, *#excision*), (*#ghnim*, *#5wenjyya*), (*#ghnim*, *#islamisation*), (*#ghnim*, *#exision*) et (*#ghnim*, *#ghanim*) sont indéfinies puisqu'ils ne cooccurrent pas. D'autre part *#ghoneim* n'a aucune valeur *NPMI* car il n'apparait avec aucun hashtag. Il est donc difficile d'avoir tous les hashtags présentés dans la figure 6.2 dans un même *cluster*.

[10] faute d'orthographe de terme *'excision'*

[11] réfère aux islamistes extrémistes

Figure 6.2 – Graphes de cooccurrence pour des hashtags référant à l'arrivée de *Wajdi Ghanim*



Comme *Soundex* a pu regrouper efficacement les hashtags avec variation orthographique. Nous avons décidé de l'utiliser afin d'améliorer les résultats obtenus par $CoDBscan_{npmi}$. Nous avons remplacé tous les hashtags par le code *Soundex* correspondant. Ensuite, nous avons recalculé les valeurs de *NPMI* entre les codes *Soundex*. Cette méthode a favorisé la relation de cooccurrence pour plusieurs hashtags, notamment ceux qui ont une grande variété orthographique. La figure 6.3 montre le nouveau graphe généré pour les hashtags présentés dans la figure 6.2. Contrairement à la figure 6.2, la figure 6.3 montre qu'il y une relation entre *#ghoneim* et les autres hashtags. Aussi, nous avons constaté qu'il y a une relation entre *SLMST* (code *Soundex* de *#islamisation*) et *KHWNJ* (code *Soundex* de *#5wenjyya*), ainsi que *#koba* et *SLMST*. Ceci est expliqué par la cooccurrence des hashtags *#5wenjya* et *#islamistes* qui ont respectivement les mêmes code *Soundex* que *#5wenjyya* et *#islamisation*. De même pour *#koba* et *#islamistes* (même code *Soundex* que *#islamisation*) qui cooccurrent.

Figure 6.3 – Graphes de cooccurrences pour des codes *Soundex* référant à l'arrivée de *Wajdi Ghanim* ; **GHNM0** : code *Soundex* des hashtags référant au nom de prédicateur (*#ghoneim*, *#ghanim*, *#ghnim*, *#ghonim* ...) ; **KHWNJ** : code *Soundex* de hashtags référant aux islamistes extrémistes (*#5wenjya, #5wenjyya, #khawanj, #khwanjia, #khwenjia, #khwenjiya*) ; **SLMST** : code *Soundex* référant aux variations de hashtags islamistes et islamisation (*#islamisation ; #islamist ; #islamiste ; #islamistes ; #islamists*) ; **XSN00** : code *Soundex* de hashtags référant à l'excision (*#excision ; #exision*)

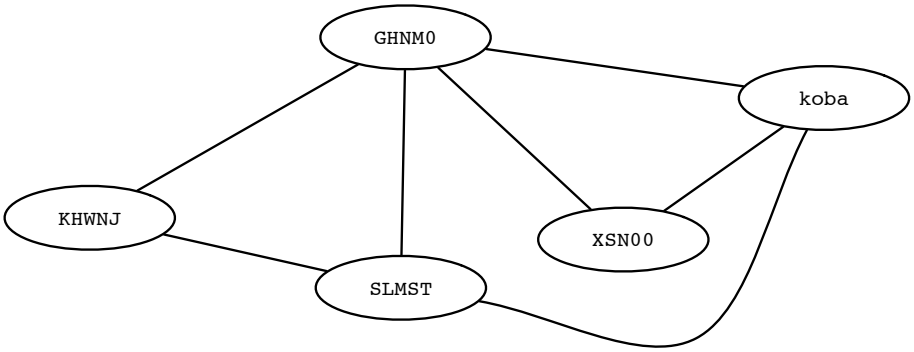


Tableau 6.XVI – Nombre de différents hashtags et *clusters* obtenus à l'aide de $CoDBscan_{npmiWithSndx}$. **NH** : le nombre de hashtags distincts regroupés ; **NG** : le nombre de *clusters* obtenus ;

		Valeur λ																			
		0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
		NH	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**
Valeur ε	0.1	8965	2358	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.2	8676	2144	8617	2401	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.3	8291	1709	8193	2035	8167	2150	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.4	7874	1451	7822	1666	7794	1800	7745	1868	-	-	-	-	-	-	-	-	-	-	-	-
	0.5	7394	1123	7260	1416	7168	1512	7197	1556	7216	1565	-	-	-	-	-	-	-	-	-	-
	0.6	6633	908	6490	1139	6424	1236	6397	1286	6374	1360	6353	1361	-	-	-	-	-	-	-	-
	0.7	5070	718	4928	863	4861	951	4785	985	4816	1036	4809	1046	4806	1054	-	-	-	-	-	-
	0.8	2996	526	2958	567	2876	597	2858	609	2834	644	2815	645	2812	651	2798	650	-	-	-	-
	0.9	1235	277	1226	282	1216	282	1203	291	1192	293	1189	296	1191	299	1188	298	1186	303	-	-
	1	313	94	313	94	313	94	313	94	313	94	313	94	313	94	313	94	313	94	312	94

Le tableau 6.XVI montre le nombre de hashtags regroupés et le nombre de *clusters* obtenus pour différentes valeurs de (ε et λ). Pour $\varepsilon \in [0.1, 0.7]$, le nombre de hashtags couverts par $CoDBscan_{npmiWithSndx}$ est toujours plus grand que celui de $CoDBscan_{npmi}$. Tandis qu'avec $\varepsilon \geq 0.8$,$CoDBscan_{npmi}$ couvre plus de hashtags que $CoDBscan_{npmiWithSndx}$. Lorsque $\varepsilon \geq 0.8$, $CoDBscan_{npmi}$ regroupe que les hashtags fortement dépendants. Toutefois, il y a moins de codes *Soundex* avec *NPMI* supérieur à 0.8. Le regroupement des hashtags avec $CoDBscan_{npmiWithSndx}$ affecte la valeur de *NPMI* pour les hashtags fortement dépendants (avec $NPMI \geq 0.8$). Par exemple, *NPMI* (*#labess* [12], *#ettouniseyatv* [13]) = 1, cependant *NPMI* (LBS00 [14] = 0.58 puisque les hashtags (p.ex *#labes, #labès, #labass* ...) qui ont le même code *Soundex* que *#labess* cooccurrent avec d'autres hashtags que *#ettouniseyatv*. Quand on compare deux codes *Soundex*, plusieurs hashtags entrent en jeu. À titre d'exemple, le hashtag *#labass* cooccurre avec *#anonymoustn* qui ne fait pas partie de *cluster TNSYT*. En calculant le *NPMI* entre les codes *Soundex LBS00* et *TNSYT*, le score d'information mutuelle obtenu descend.

Le nombre de codes *Soundex* considérés par $CoDBscan_{npmiWithSndx}$ est 7855 parmi 9033. Les 1178 codes *Soundex*, non considérés, correspondent à des valeurs de *NPMI* non définies, car le (s) hashtag (s) associé (s) à ces codes *Soundex* ne cooccurrent avec aucun autre hashtag. Cette démarche a augmenté le nombre de hashtags considérés de 9 973 à 10 929.

[12] hashtag réfère à une émission sur la chaine *'ettounsiya TV'*
[13] réfère à la chaine de télévision *'ettounissia'*
[14] code de *Soundex* de *#labess*, TNSYT [15])

Tableau 6.XVII – Évaluation manuelle des résultats obtenus avec $CoDBscan_{npmiWithSndx}$. **NH** : le nombre de hashtags dans le *cluster* ; **Pr** : la précision

	Valeur ε																			
	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
	NH	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**
1	165	0.58	280	0.39	108	0.73	98	0.83	80	0.88	43	0.98	33	0.96	25	1.00	14	1.00	6	1.00
2	149	0.35	181	0.80	105	0.84	72	0.79	38	0.87	36	0.88	25	1.00	13	0.79	11	1.00	6	1.00
3	149	0.42	151	0.51	99	0.62	66	0.77	37	0.85	32	0.79	24	0.96	13	0.77	10	1.00	5	1.00
4	130	0.56	134	0.49	83	0.75	60	0.82	36	0.97	28	0.82	21	0.88	12	0.85	10	1.00	5	1.00
5	125	0.59	123	0.87	74	0.47	59	0.77	36	0.67	27	0.88	20	0.95	12	0.85	9	1.00	5	1.00
6	123	0.64	120	0.85	71	0.66	58	0.87	35	0.69	26	0.88	20	0.90	12	1.00	9	1.00	5	1.00
7	117	0.64	109	0.73	70	0.74	57	0.73	34	0.65	25	0.91	19	0.86	11	0.83	9	1.00	5	1.00
8	116	0.66	106	0.83	70	0.53	56	0.63	33	0.73	25	0.87	18	0.95	11	0.83	9	1.00	5	1.00
9	115	0.75	98	0.58	69	0.84	54	0.87	33	0.70	24	0.90	17	0.89	11	0.83	8	1.00	5	1.00
10	114	0.73	95	0.51	66	0.79	52	0.82	32	0.73	24	0.93	17	0.94	11	0.91	8	1.00	4	1.00
11	110	0.74	94	0.58	65	0.75	51	0.84	32	0.97	23	0.93	17	0.89	11	1.00	8	1.00	4	1.00
12	105	0.62	91	0.77	64	0.73	50	0.72	32	0.70	23	0.88	17	0.94	11	1.00	8	1.00	4	1.00
13	105	0.61	90	0.52	62	0.57	46	0.72	31	0.94	22	0.88	16	0.94	11	1.00	8	1.00	4	1.00
14	103	0.82	88	0.70	61	0.77	46	0.69	31	0.69	22	1.00	16	1.00	11	1.00	7	1.00	4	1.00
15	102	0.78	80	0.63	61	0.86	45	0.76	30	0.68	22	0.91	15	0.94	11	1.00	7	1.00	4	1.00
16	101	0.74	80	0.88	60	0.82	45	0.71	30	0.57	22	0.91	15	0.94	11	1.00	7	1.00	5	1.00
17	100	0.53	80	0.78	59	0.70	45	0.75	30	0.63	22	0.87	15	0.88	11	1.00	7	1.00	4	1.00
18	97	0.91	79	0.69	58	0.57	45	0.70	30	0.53	21	0.91	15	0.94	10	1.00	7	1.00	4	1.00
19	95	0.81	79	0.75	58	0.56	44	0.67	30	0.87	21	0.95	15	0.87	10	1.00	7	1.00	4	1.00
20	94	0.74	76	0.87	58	0.83	44	0.55	30	0.63	21	0.86	14	0.93	10	1.00	7	1.00	4	1.00
Moyenne		0.66		0.69		0.71		0.72		0.75		**0.90**		**0.93**		**0.93**		1.00		1.00

Le tableau 6.XVII montre les valeurs de précision obtenues par $CoDBscan_{npmiWithSndx}$ pour différents ε. Comme $CoDBscan_{npmi}$, plus ε est grand, plus les résultats retournés sont précis, tandis que le nombre de hashtags regroupés n'a pas dépassé 303 pour $\varepsilon = 0.9$ et 313 pour $\varepsilon = 1$. Toutefois, avec $\varepsilon \leq 0.4$, $CoDBscan_{npmiWithSndx}$ a regroupé plus de hashtags mais la précision obtenue n'a pas dépassé 72 %. Nous avons constaté qu'avec $\varepsilon \in [0.6, 0.7]$, on a pu regrouper plusieurs hashtags avec une bonne précision.

6.1.4 Regroupement avec $DBscan_{Hyper}$

Dans la section précédente, nous avons utilisé l'information mutuelle pour déterminer la similarité entre deux hashtags. Dans cette section, la similarité sémantique entre deux hashtags est déterminée par le nombre d'hyperliens partagés (voir section 4.3.3). Plus les deux hashtags partagent d'hyperliens plus qu'ils sont sémantiquement similaires. Pour regrouper les hashtags similaires, nous avons appliqué aussi l'algorithme de *DBscan*. La valeur ε correspond à la mesure de similarité définie dans la section 4.3.3.

Le tableau 6.XVIII montre la distribution des plus importants *clusters* obtenus en appliquant la version standard (sans *Cohesion*) de *DBscan*. Parmi les 12 218 hashtags, il y a que 6 008 considérés par $DBscan_{Hyper}$, les autres hashtags non considérés ne partagent pas d'hyperliens avec d'autres hashtags. Pour les hashtags considérés, il y a 24 303 pairs avec une valeur définie de $Similarity_{Hyper}$. Nous avons remarqué que comme pour à $DBscan_{NPMI}$, un immense *cluster* a été créé à cause du phénomène *voisins-des-voisins* (voir section 6.1.3.3). Ainsi, nous avons appliqué notre algorithme $CoDBscan_{hyper}$ qui vérifie la cohérence d'un hashtag avant de l'ajouter à un *cluster*.

Tableau 6.XVIII – Les *clusters* les plus importants obtenus par $DBscan_{Hyper}$ standard ; $\varepsilon = 0.4$ *ptsMin*=2 ; ***ID* cluster** est l'identifiant du *cluster*

***ID* cluster**	***Taille***
C2	849
C5	110
C126	80
C70	66
C21	57

Le tableau 6.XIX montre le nombre de hashtags regroupés ainsi que le nombre de *clusters* obtenus pour différentes valeurs de ε et λ. Nous avons observé que le nombre de hashtags regroupés chute quand ε dépasse 0.6, ceci parce qu'il y a une faible proportion (14 %) de pairs de hashtags qui ont une valeur de $Similarity_{Hyper} \geq 0.6$.

Tout comme nous avons fait précédemment, nous avons demandé à nos experts d'évaluer la précision des 20 plus importants *clusters* obtenus à l'aide de $Similarity_{Hyper}$. Avec $\varepsilon = 0.5$, $CoDBscan_{hyper}$ a regroupé 3 556 hashtags différents avec une précision = 92% (voir tableau 6.XX). Tandis que avec $\varepsilon > 0.5$, $CoDBscan_{hyper}$ obtient une précision $\in$ [0.99, 1], cependant le nombre de hashtags regroupés est petit.

Tableau 6.XIX – Nombre de différents hashtags et *clusters* obtenus à l'aide de $CoDBscan_{hyper}$. **NH** : le nombre de hashtags distincts regroupés ; **NG** : le nombre de *clusters* obtenus ;

		Valeur λ																			
		0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
		NH	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**
Valeur ε	0.1	4373	1544	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.2	4168	1471	4065	1518	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.3	3856	1323	3721	1346	3692	1356	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.4	3654	1234	3598	1276	3504	1278	3465	1289	-	-	-	-	-	-	-	-	-	-	-	-
	0.5	3556	1213	3477	1253	3452	1241	3455	1212	3421	1249	-	-	-	-	-	-	-	-	-	-
	0.6	1626	542	1620	539	1606	550	1608	547	1609	564	1602	546	-	-	-	-	-	-	-	-
	0.7	1038	345	1037	349	1038	352	1032	345	1032	347	1032	347	1032	365	-	-	-	-	-	-
	0.8	676	225	676	230	675	227	675	235	675	232	674	231	675	234	674	236	-	-	-	-
	0.9	608	208	608	211	608	201	608	213	608	211	608	206	608	208	608	214	608	210	-	-
	1	589	204	589	205	589	201	589	205	589	202	589	205	589	205	589	207	589	209	589	200

Tableau 6.XX – Évaluation manuelle des résultats obtenus avec $CoDBscan_{hyper}$. **NH** : le nombre de hashtags dans le *cluster* ; **Pr** : la précision

	Valeur ε																			
	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
	NH	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**
1	74	0.86	58	0.86	51	0.90	68	0.85	38	0.89	20	0.95	16	1.00	11	1.00	9	1.00	9	1.00
2	62	0.84	46	0.89	49	0.82	38	0.82	34	0.88	19	0.89	12	1.00	9	1.00	9	1.00	9	1.00
3	51	0.76	43	0.58	35	0.71	34	0.65	34	0.76	17	1.00	12	1.00	9	1.00	9	1.00	8	1.00
4	51	0.80	41	0.66	32	0.81	32	0.72	27	0.96	16	1.00	12	1.00	8	1.00	8	1.00	8	1.00
5	50	0.82	40	0.75	32	0.72	31	0.87	26	0.92	16	0.88	11	1.00	8	1.00	8	1.00	8	1.00
6	45	0.76	37	0.70	31	0.77	29	0.86	24	0.96	16	1.00	11	1.00	8	1.00	7	1.00	7	1.00
7	42	0.76	34	0.71	31	0.87	27	0.85	24	0.88	15	1.00	11	1.00	8	1.00	7	1.00	7	1.00
8	40	0.65	31	0.81	30	0.83	24	0.92	24	1.00	14	1.00	10	1.00	7	1.00	7	1.00	7	1.00
9	38	0.68	29	0.86	29	0.86	24	0.75	23	0.83	14	1.00	10	1.00	7	1.00	7	1.00	7	1.00
10	38	0.55	27	1.00	29	0.86	23	1.00	22	0.86	13	1.00	9	1.00	7	1.00	7	1.00	7	1.00
11	37	0.73	26	0.88	28	0.86	22	0.82	22	0.95	12	1.00	9	1.00	7	1.00	7	1.00	6	1.00
12	35	0.63	26	0.73	28	0.82	22	1.00	20	0.80	12	1.00	9	1.00	7	1.00	7	1.00	6	1.00
13	33	0.82	26	0.96	26	0.88	21	1.00	18	1.00	11	1.00	8	1.00	7	1.00	6	1.00	6	1.00
14	32	0.72	25	0.72	26	0.92	21	1.00	18	0.94	11	1.00	8	1.00	6	1.00	6	1.00	6	1.00
15	30	0.53	25	0.80	26	0.77	20	0.70	18	1.00	11	1.00	7	1.00	6	1.00	6	1.00	6	1.00
16	28	0.86	25	0.76	25	1.00	20	0.90	18	1.00	11	1.00	7	1.00	6	1.00	6	1.00	5	1.00
17	27	1.00	23	0.87	24	0.75	19	0.84	17	0.94	10	1.00	7	1.00	6	1.00	6	1.00	5	1.00
18	25	0.68	23	0.74	24	0.92	19	0.79	17	0.94	10	1.00	6	1.00	5	1.00	6	1.00	5	1.00
19	25	0.92	21	0.81	23	0.65	19	0.84	17	0.94	10	1.00	6	1.00	5	1.00	5	1.00	5	1.00
20	24	1.00	21	0.86	21	0.71	18	0.72	16	0.94	9	1.00	6	1.00	5	1.00	5	1.00	5	1.00
Moyenne		0.77		0.80		0.82		**0.85**		**0.92**		**0.99**		1.00		1.00		1.00		1.00

Amélioration avec *Soundex* :

Comme effectué dans la section 6.1.3, nous avons essayé d'améliorer la performance de $CoDBscan_{hyper}$ en l'appliquant sur les codes *Soundex*. Parmi 9033 codes *Soundex*, 4 708 partagent des hyperliens avec d'autres. Cette démarche a augmenté le nombre de hashtags considérés de 6 003 à 7 431. Le tableau 6.XXI montre le nombre de hashtags ainsi que le nombre de *clusters* obtenus par $CoDBscan_{hyperWithSndx}$ pour différentes valeurs de (ε et λ). De même que le nombre de hashtags regroupés a diminué d'une façon remarquable, car seulement 18 % de pairs de codes *Soundex* ont une valeur de $Similarity_{Hyper} >= 0.6$.

Le tableau 6.XXII montre les précisions de 20 *clusters* les plus importants.

Tableau 6.XXI – Nombre de différents hashtags et *clusters* obtenus à l'aide de$CoDBscan_{hyperWithSndx}$. **NH** : le nombre de hashtags distincts regroupés ; **NG** : le nombre de *clusters* obtenus ;

		Valeur λ																			
		0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
		NH	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**	**NH**	**NG**
Valeur ε	0.1	5739	1288	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.2	5377	1143	5269	1250	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.3	4867	982	4800	1095	4769	1153	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.4	4660	925	4569	1016	4542	1059	4570	1094	-	-	-	-	-	-	-	-	-	-	-	-
	0.5	4597	897	4545	1022	4545	1043	4483	1074	4491	1047	-	-	-	-	-	-	-	-	-	-
	0.6	1500	284	1493	298	1483	314	1456	319	1434	323	1452	322	-	-	-	-	-	-	-	-
	0.7	775	164	770	165	772	176	763	172	763	180	752	177	759	180	-	-	-	-	-	-
	0.8	398	94	398	95	399	97	398	96	398	96	397	98	397	96	398	99	-	-	-	-
	0.9	313	78	313	79	313	78	313	79	313	80	313	78	313	82	313	78	312	79	-	-
	1	292	76	292	76	292	76	292	76	292	76	292	76	292	76	292	76	292	76	292	76

Tableau 6.XXII – Évaluation manuelle des résultats obtenus avec $CoDBscan_{hyperWithSndx}$. **NH** : le nombre de hashtags dans le *cluster* ; **Pr** : la précision

	Valeur ε																			
	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1	
	NH	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**	**NH**	**Pr**
1	118	0.75	124	0.77	123	0.79	103	0.86	126	0.87	35	1.00	32	1.00	14	1.00	12	1.00	9	1.00
2	94	0.73	76	0.80	69	0.83	86	0.79	84	0.82	25	1.00	16	1.00	11	1.00	9	1.00	9	1.00
3	85	0.79	70	0.71	66	0.77	67	0.90	65	0.88	23	1.00	12	1.00	9	1.00	9	1.00	8	1.00
4	81	0.65	65	0.77	65	0.88	64	0.94	60	0.93	20	1.00	12	1.00	9	1.00	8	1.00	8	1.00
5	77	0.74	61	0.92	65	0.77	60	0.88	60	0.97	16	1.00	11	1.00	9	1.00	8	1.00	7	1.00
6	76	0.72	59	0.90	56	0.96	56	0.91	56	0.98	15	1.00	11	1.00	9	1.00	7	1.00	7	1.00
7	70	0.73	57	0.68	51	0.96	56	0.95	56	0.96	15	1.00	11	1.00	9	1.00	7	1.00	7	1.00
8	69	0.77	55	0.71	51	0.90	53	0.87	53	0.91	15	1.00	10	1.00	8	1.00	7	1.00	6	1.00
9	67	0.78	54	0.93	51	0.82	51	0.92	52	0.94	14	1.00	10	1.00	8	1.00	7	1.00	6	1.00
10	66	0.80	52	0.94	50	0.80	51	0.92	52	0.98	14	1.00	9	1.00	7	1.00	6	1.00	5	1.00
11	66	0.82	51	0.90	50	0.82	51	0.82	51	0.88	14	1.00	9	1.00	7	1.00	6	1.00	5	1.00
12	65	0.92	51	0.90	49	0.90	48	0.75	50	0.84	13	1.00	9	1.00	7	1.00	5	1.00	5	1.00
13	65	0.62	51	0.96	47	0.85	47	0.83	50	0.86	13	1.00	8	1.00	7	1.00	5	1.00	5	1.00
14	64	0.73	47	0.94	47	0.87	46	0.98	47	0.89	13	1.00	8	1.00	7	1.00	5	1.00	5	1.00
15	61	0.87	47	0.89	46	0.96	45	0.98	46	0.91	13	1.00	8	1.00	6	1.00	5	1.00	5	1.00
16	60	0.83	47	0.89	45	0.89	45	0.96	46	0.93	13	1.00	7	1.00	6	1.00	5	1.00	5	1.00
17	59	0.80	47	0.83	45	0.96	42	0.98	45	0.98	12	1.00	7	1.00	6	1.00	5	1.00	4	1.00
18	59	0.85	45	0.96	45	0.96	41	1.00	42	0.95	12	1.00	7	1.00	5	1.00	5	1.00	4	1.00
19	59	0.76	45	0.93	42	0.93	41	0.98	42	1.00	11	1.00	7	1.00	5	1.00	5	1.00	4	1.00
20	55	0.75	44	0.89	42	0.98	40	1.00	41	0.88	11	1.00	7	1.00	5	1.00	4	1.00	4	1.00
Moyenne		0.77		0.86		0.88		**0.91**		**0.92**		1.00		1.00		1.00		1.00		1.00

6.2 Conclusion

Dans ce chapitre, nous avons effectué des expérimentations pour les méthodes de normalisation et de regroupement des hashtags sémantiquement similaires, présentées au chapitre 3.

Les algorithmes de *Soundex* et de translittération ont réussi à regrouper efficacement les hashtags (écrits en alphabets latins et arabes) de même prononciation. Cependant, nous n'avons pas effectué de changement pour les hashtags qui ont un *Noisy Soundex* car ces derniers ont montré une faible précision. Nous avons, également, utilisé des expressions régulières afin de normaliser les hashtags correspondant à des dates.

En appliquant toutes ces méthodes, nous avons transformé les 12 218 hashtags à 9 033 *clusters*, soit une réduction de ≈ 26 %.

Avec *DBscan*, nous avons réussi à regrouper des hashtags sémantiquement similaires même s'ils n'ont pas une prononciation semblable, ce qui était impossible avec *Soundex*. Nous avons proposé deux variations de *DBscan* : la première s'appuie sur la mesure de *NPMI* pour déterminer le degré de similarité entre deux hashtags (ou codes *Soundex*) tandis que la deuxième utilise sur les hyperliens partagés par une paire de hashtags (ou *Soundex*) comme mesure de similarité. Les résultats obtenus ont montré que ces deux mesures (*NPMI* et les hyperliens partagés) sont de bons indices pour déterminer la similarité entre les hashtags. Dans notre contexte, la version standard de *DBscan* n'était pas efficace, pour cette raison nous l'avons modifiée afin de bien regrouper un nombre important de hashtags de notre corpus.

L'évaluation des résultats était coûteuse en temps, il nous fallait des heures ou même des jours pour calculer les précisions de toutes les variantes de *CoDbScan*. Nos variantes de *CoDbScan* utilisent trois paramètres : *MinPts*, ε, λ. Pour les variantes $CoDBscan_{npmi}$ et $CoDBscan_{npmiWithSndx}$, il est recommandé d'utiliser une valeur $\varepsilon \in [0.6, 0.8]$ car avec une telle valeur nous avons réussi à regrouper le plus grand nombre de hashtags avec une forte valeur de précision. Tandis que pour les variantes de $CoDBscan_{hyper}$ et $CoDBscan_{hyperWithSndx}$, il recommandé d'utiliser une valeur $\varepsilon \in [0.4, 0.6]$.

En général, les annotations obtenues manuellement par les experts peuvent être ex-

ploitées par d'autres travaux dans le futur sous forme de données de référence afin de déterminer la similarité entre certains hashtags. Cependant, certains termes peuvent toujours être regroupés car ils n'ont pas de dépendance temporelle. D'autre part, certains sont regroupés seulement pour une certaine période. Par exemple, une relation sémantique entre *'université de manouba'* et *'drapeau tunisien'* ne pourrait être valide qu'au cours de la période analysée dans ce travail. En dehors de cette période, il n'y a aucune garantie de considérer ces termes similaires.

Le tableau 6.XXIII récapitule les principaux résultats obtenus dans ce chapitre. Pour la tâche de normalisation le nombre de hashtags regroupés est le même que celui de nombre de hashtags considérés vu que chaque hashtag possède un code Soundex (*cluster*). Par contre pour les variantes de *CoDbScan*, il est possible qu'un hashtag n'appartienne à aucun *cluster*.

Tableau 6.XXIII – Récapitulation de certains résultats obtenus par les méthodes de regroupement de hashtags

Tâches	**NHC**	**NGobt**	**NHreg**	**Précision**
Normalisation par Soundex , normalisation des dates)	12 218	9 033	12 218	96 % (pour les 15 *clusters* les plus importants obtenus par *Soundex*)
$CoDBscan_{npmi}$ ($\varepsilon = 0.7$)	9 973	928	5 011	94 % (pour les 20 *clusters* les plus importants)
$CoDBscan_{npmiWithSndx}$ ($\varepsilon = 0.6$)	10 929	908	6 633	90 % (pour les 20 *clusters* les plus importants)
$CoDBscan_{hyper}$ ($\varepsilon = 0.5$)	6 008	1 213	3 556	92 % (pour les 20 *clusters* les plus importants)
$CoDBscan_{hyperWithSndx}$ ($\varepsilon = 0.4$)	7 431	925	4 660	91 % (pour les 20 *clusters* les plus importants)

CHAPITRE 7

EXPÉRIMENTATIONS : DÉTECTION D'ÉVÈNEMENTS

Les résultats, présentés dans le chapitre précédent, ont montré l'efficacité de nos méthodes de *regroupement* de hashtags sémantiquement similaires. Rappelons que chaque groupe contient des hashtags qui portent sur un même sujet.

Dans ce chapitre, nous utilisons ces résultats afin de déterminer les évènements qui se sont déroulés sur une période donnée. Pour réaliser cette tâche, nous nous basons d'abord sur les groupes obtenus par la tâche de normalisation (*Soundex*, normalisation des dates, translittération). Par la suite, nous montrons les résultats obtenus par la méthode de *Topic Model (*LDA*)*. Enfin, nous présentons les résultats de l'algorithme de clustering incrémental (*ACI*) (chapitre 4) qui sert à regrouper les tweets qui portent sur un même sujet. Nous effectuons finalement une expansion des tweets en remplaçant chaque hashtag par d'autres qui lui sont similaires en nous basant sur les résultats obtenus par *CoDBscan*.

7.1 Détection des évènements en utilisant *Soundex*, la translittération et la normalisation des dates

Dans cette section, nous utilisons les *clusters* obtenus après la tâche de normalisation (voir les sections 6.1.1 et 6.1.2) pour déterminer les évènements discutés par les utilisateurs de Twitter durant une période donnée. Cette méthode se base sur la fréquence de chaque *cluster* pendant toute la période pour déterminer l'ensemble *EV* des sujets qui réfèrent à des évènements. Ensuite, nous en déterminons les dates saillantes en utilisant la méthode de Palshikar (2009) (voir section 5.6). Pour sélectionner les éléments de *EV*, nous avons utilisé trois critères : *fréquence*, *variation* et *Tf-Idf*.

7.1.1 Fréquence :

Le tableau 7.I montre les 10 sujets les plus fréquents entre le 08 février et le 15 avril 2012. Cependant, nous avons constaté que le critère de fréquence n'est pas fiable pour distinguer entre les sujets dans EV et les autres notés $\overline{EV}$. Parmi les 10 sujets présentés dans le tableau 7.I, seul *9avril* réfère réellement à un évènement significatif. Les autres sujets réfèrent plutôt à des annonces d'emploi, des prévisions météo ou des sujets qui sont toujours présents. Aussi, nous avons remarqué que la variation des sujets dans $\overline{EV}$ est faible par rapport à ceux de EV. Ce qui indique que la présence de ces sujets est presque uniforme durant la période. La variation des sujets est définie par l'écart-type de sa fréquence quotidienne. En outre, les sujets de $\overline{EV}$ sont toujours présents, contrairement à ceux de EV qui n'apparaissent fréquemment qu'au cours d'une certaine période.

Tableau 7.I – Les 10 sujets les plus fréquents avec leurs dates saillantes. Toutes les dates sont en 2012. **DF** : nombre de jours où le sujet est présent. **E-t** : écart-type

code ***Soundex***	**Hashtags**	**Dates**	**DF**	**Fréquence**	**E-t**	***Tf-Idf***
9avril	#9avril #9april #9avirl #9avr	09/04	14	6070	154.81	4127,27
MPL00	#emplo #emploi	03/16	67	3025	22,12	0,00
RCRTM	#recrutement	03/16	67	2854	22,63	0,00
JBS00	#jobs	03/16	67	2812	22,08	0,00
NTRM0	#interim	03/16	67	2804	22,26	0,00
KNKRS	#concours	03/16	67	2787	22,21	0,00
KTR00	#qatar #qatar	02/21	67	2193	27,94	0,00
GYPT0	#egyipt #égypte	03/02 04/09	67	2098	43,92	0,00
WTHR0	#weather	02/15 02/16	62	2043	10,25	68,81
NHD00	#enahda #ennahada	02/21	66	1872	66,41	12,23

7.1.2 Variation

Nous avons supposé que les sujets dont la variation de fréquence quotidienne est grande font partie de *EV*. L'intuition derrière cette supposition est les sujets importants sont discutés fréquemment pendant une précise, suivra ensuite une baisse de fréquence (voir figure 5.5). Tandis que les sujets de $\overline{EV}$ sont discutés quotidiennement d'une façon quasi uniforme. La variation de la fréquence quotidienne des sujets est déterminée par l'écart-type de leur fréquence quotidienne. Le tableau 7.II montre les 10 sujets avec les plus grandes valeurs d'écart-type. Plus la valeur de l'écart-type de la fréquence quotidienne du sujet est grande plus il est important. En effet, le critère de la variation pénalise les sujets de $\overline{EV}$ tel que : *WTHR0* qui a une variation faible par rapport à ceux de *EV*.

Tableau 7.II – Les 10 sujets qui ont les plus grandes valeurs d'écart-type. Toutes les dates sont en 2012. Toutes les dates sont en 2012. **DF** : nombre de jours où le sujet est présent. **E-t** : écart-type

code *Soundex*	**Hashtags**	**Dates**	**DF**	**Fréquence**	**E-t**	***Tf-Idf***
9avril	#9avril #9april #9avirl #9avr	09/04	14	6070	154.81	4127,27
20mars	#20mars, #20mars2012	03/20	15	1422	124,59	924,27
ugtt	#ugtt	02/25	53	1797	97,76	182,93
NHD00	#enahda #ennahada	02/21	66	1872	66,41	12,23
3KB00	#acab	04/09	34	789	60,15	232,43
MNB00	#mannoub, #mannouba	03/07	42	812	53,91	164,69
TNPHR	#tunpharma	04/04 04/05	15	1710	52,79	1111,47
PHRMC	#pharmacie #pharmacies	04/04 04/05	19	1608	49,32	880,09
3LGRY	#algeria #algria	02/16 02/21	61	1712	46,27	69,75
GYPT0	#egyipt #égypte	03/02 04/09	67	2098	43,92	0,00

7.1.3 *Tf-Idf*

Nous avons utilisé une mesure inspirée de *Tf-Idf* pour trier les sujets.

- ***Tf*** : correspond à la fréquence quotidienne du sujet S_i durant la période.
- $Idf(S_i) = \log \frac{Nombre\ de\ jours\ total}{Nombre\ de\ jours\ où\ S_i\ est\ présent}$

Nous avons supposé que les sujets avec les plus grandes valeurs *Tf-Idf* correspondent à des évènements (voir tableau 7.III). L'intuition derrière ce critère est de donner plus d'importance aux sujets qui ne sont fréquemment discutés que dans une période précise.

Cette mesure pénalise les sujets qui apparaissent sur plusieurs jours même s'ils sont fréquents. Les sujets *MPL00*, *RCRTM*, *JBS00*, *NTRM0* et *KNKRS* ont une valeur *Tf-Idf* égale à 0 car ils sont toujours présents même s'ils sont fréquents. (voir tableau 7.I)

Tableau 7.III – Les 10 sujets avec les plus grandes valeurs de *Tf-Idf*. Toutes les dates sont en 2012. Toutes les dates sont en 2012. **DF** : nombre de jours où le sujet est présent. **E-t** : écart-type

code *Soundex*	**Hashtags**	**Dates**	**DF**	**Fréquence**	**E-t**	***Tf-Idf***
9avril	#9avril #9april #9avirl #9avr	09/04	14	6070	154.81	4127,27
TNPHR	#tunpharma	04/04 04/05	15	1710	52,79	1111,47
20mars	#20mars, #20mars2012	03/20	15	1422	124,59	924,27
3PLKT	#application	04/04 04/05	15	1355	41,89	880,72
PHRMC	#pharmacie #pharmacies	04/04 04/05	19	1608	49,32	880,09
MNF20	#manif20mars	03/18 03/20	5	413	29,6	465,49
MBL00	#mobile	04/04 04/05	28	1154	34,72	437,26
TSHDY	#techdays, #techdaystunisia,	02/13	7	416	33,72	408,08
JBLKS	#jbalileaks #jbeleaks	04/08	6	336	33,98	352,10
GHNM0	#ghanim #ghenim	02/15 02/17	25	756	30,74	323,66

7.1.4 Identification des pics

Rappelons que notre objectif est de détecter, parmi tous les sujets, l'ensemble *EV*. Nous allons considérer que les éléments de *EV* sont les sujets avec les plus grandes valeurs pour les critères présentés (*fréquence*, *variation* ou *Tf-Idf*). Nous ne retenons que les sujets, dont la valeur (correspondant au critère considéré) dépasse un certain seuil.

Pour les trois critères, nous considérons que les sujets avec les plus grandes valeurs (correspondant au critère utilisé) appartiennent à *EV*. Par conséquent, nous constatons que les sujets de *EV* correspondent à des pics, ce qui n'est pas le cas pour ceux de $\overline{EV}$. À cet effet, nous avons réutilisé la fonction de détection de pics (section 5.6) avec x_i ($1 \leq i \leq N$), $N = 9\,033$.

Nous n'avons pas utilisé la variable k parce que nous ne travaillons pas sur une série temporelle.

En appliquant cette méthode, nous avons retenu, parmi 9 033 sujets :

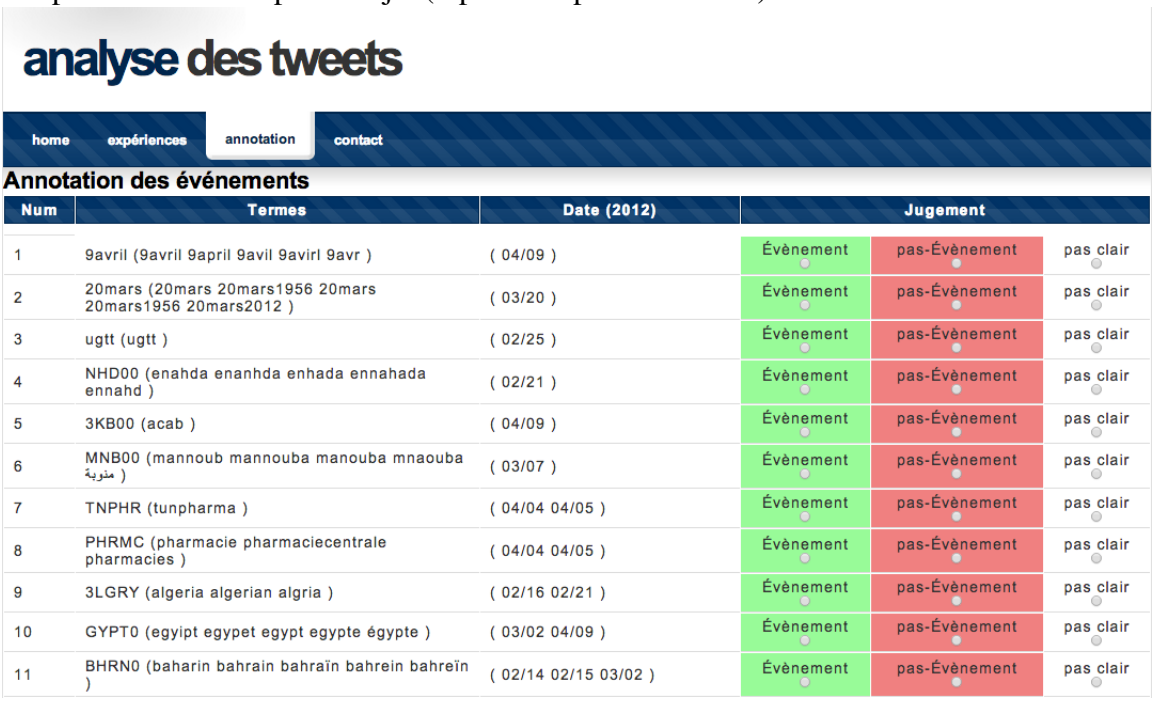
- critère de *fréquence* : 123 sujets sont dans *EV* (c.-à-d. 123 sujets dont la fréquence correspond à un pic par rapport aux fréquences des autres sujets).
- critère de *variation* : 88 sujets sont *EV* (c.-à-d. 88 sujets dont l'écart-type correspond à un pic par rapport aux écarts-types des autres sujets).
- critère de *Tf-Idf* : 81 sujets sont *EV* (c.-à-d. 81 sujets, dont leur valeur de *Tf-Idf* correspond à un pic par rapport aux valeurs de *Tf-Idf* des autres sujets).

7.1.5 Évaluation

Dans ce type de problème, il est difficile de déterminer la pertinence des résultats obtenus, car il n'y a pas de données de référence.

Pour évaluer la robustesse de chaque critère, nous avons vérifié si les sujets que nous avions détectés devraient appartenir à *EV* ou non. À cette fin, nous avons demandé à 10 experts tunisiens familiers avec les évènements qui ont eu lieu en Tunisie afin de déterminer l'ensemble (*EV* ou $\overline{EV}$) de chaque sujet. La majorité de nos experts sont

Figure 7.1 – Capture d'écran de la page dédiée à l'annotation de l'ensemble *EV* détecté. *Évènement* : l'expert reconnaît le sujet (représenté par des termes) et le considère comme un évènement (un élément de *EV*) ; *pas Évènement* : l'expert reconnaît le sujet (représenté par des termes) et ne le considère pas comme un évènement ; *pas clair* : l'expert ne reconnaît pas le sujet (représenté par des termes)



des étudiants tunisiens au Québec. L'annotation des évènements a été faite par l'intermédiaire d'un site web [1] que nous avons créé. Une fois connecté, l'expert reçoit la liste des sujets sélectionnés et les dates saillantes de chaque sujet. La figure 7.1 montre une capture d'écran de la page d'annotation. L'expert doit reconnaître le sujet référé par des hashtags et choisir l'ensemble (*EV*, $\overline{EV}$ ou non reconnu [2]) approprié de chaque sujet.

Le tableau 7.VI présente les résultats d'annotations des experts pour les différents critères. Nous présentons à chaque expert le nombre de sujets de chaque catégorie (*EV*, $\overline{EV}$, non reconnu). Afin d'aider les experts à reconnaitre les sujets, nous avons mis à leur disposition une liste des sites des médias traditionnels fiables qui publient quotidienne-

[1] `http://rali.iro.umontreal.ca:8080/dridihou/`

[2] l'expert n'est pas en mesure de reconnaître le sujet à partir des hashtags présentés

ment les évènements déroulés en Tunisie tels que : *shemsfm.net* [3], *mosaiquefm.net* [4], *Association de la presse francophone* [5], etc.

Tableau 7.IV – L'annotation des experts d'évènements détectés (par les codes *Soundex*) entre le 08 février et le 15 avril 2012 ($|EV| = 123$ (pour le critère fréquence), $|EV| = 88$ (pour le critère variation), $|EV| = 81$ (pour le critère *Tf-Idf*)). *EV* : *ensemble de sujets qui réfèrent à des évènements* ; $\overline{EV}$:*ensemble de sujets qui ne réfèrent pas à des évènements* ; **Nr** : *ensemble de sujets non reconnus*.

	Fréquence			**Écart-type**			***Tf-Idf***		
	EV	$\overline{EV}$	**Nr**	*EV*	$\overline{EV}$	**Nr**	*EV*	$\overline{EV}$	**Nr**
expert1	76	45	2	72	15	1	77	4	0
expert2	74	47	2	73	14	1	77	4	0
expert3	79	42	2	69	18	1	76	5	0
expert4	81	41	1	73	15	0	78	3	0
expert5	76	46	1	73	14	1	78	2	1
expert6	82	40	1	70	18	0	76	5	0
expert7	80	41	2	69	19	0	78	3	0
expert8	73	48	2	74	14	0	78	2	1
expert9	79	42	2	75	12	1	76	5	0
expert10	82	40	1	76	12	0	77	4	0
Moyenne	78,2	43,2	1,6	72,4	15,1	0,5	77	3,7	0,2
Précision	**0,64**			**0,82**			**0,95**		

Les résultats obtenus montre que le critère de *Tf-Idf* est le plus efficace (précision = 95%) pour déterminer l'ensemble *EV*.

7.2 Détection des évènements en utilisant *Topic Model*

Dans cette section, nous utilisons la technique *LDA* pour déterminer les sujets importants dans une période donnée. Rappelons que cette technique permet de produire à partir d'un corpus donné, des *clusters* de termes sous forme de distribution. Chaque *cluster* est censé représenter un sujet. Les termes qui ont les plus grandes valeurs de probabilités sont les plus représentatifs du sujet. Dans ce travail nous supposons qu'un sujet *s* est représenté par les 10 termes les plus importants (avec les plus grandes valeurs

[3] `http://www.shemsfm.net`
[4] `http://www.mosaiquefm.net/`
[5] `http://www.afp.com/`

de probabilités), nous notons ces termes : $TopTerms^s$.

Afin d'attribuer plus de chance aux termes à cooccurer, nous avons regroupé les tweets partageant le même *code Soundex* dans un même document. À titre d'exemple, les tweets contenant les hashtags *'#ghanim', '#ghonim', '#ghoneim'* sont regroupés dans un même document. Ainsi, le corpus d'entrée est composé de 9 033 documents, soient le nombre de *code Soundex* obtenus après la tâche de normalisation.

7.2.1 Normalisation des termes

LDA est incapable d'identifier les sens des termes. Cependant, des termes comme *'parlent'*, *'parle'* sont considérés différents même s'ils correspondent au même verbe. La normalisation de ces termes peut améliorer les résultats retournés par *LDA*. Étant donné que nos tweets sont écrits principalement en français et en arabe (incluant le dialecte Tunisien), nous avons utilisé le lemmatiseur *BDF* pour remplacer un mot en français par son lemme. Par exemple, les mots *'répond'* et *'répondu'* sont remplacés par le mot *'répondre'*. Tandis que pour les mots écrits en alphabets arabes nous avons appliqué le stemmer de la librairie *Safar* [6]. Le tableau 7.V montre le nombre de termes différents trouvés dans le corpus après chaque tâche de normalisation.

Tableau 7.V – Nombre de termes différents trouvés dans le corpus après chaque tâche de normalisation

	Nombre de termes
Corpus original	173 135
Après l'application de *BDF*	159 651
Après l'application de *SAFAR*	100 411

7.2.2 Étiquetage des tweets

Nous avons parcouru tous les tweets de notre corpus et nous avons considéré qu'un tweet tw_i porte sur un sujet s si au moins un terme de $TopTerms^s$ est présent dans tw_i. Par la suite, nous avons calculé la fréquence quotidienne de chaque sujet.

[6] `http://sibawayh.emi.ac.ma/safar/index.php`

7.2.3 Expérimentation

Nous avons appliqué *LDA* avec les paramètres suivants :

- Nombre d'itérations : 1500.
- Nombre de sujets : nous avons supposé qu'il y a 500 sujets dans notre corpus.

Comme dans la section 7.1, nous avons évalué les résultats selon les trois critères : *fréquence*, *variation* et *Tf-Idf*. Ensuite, nous n'avons gardé, en utilisant la méthode de Palshikar (2009) que les éléments de *EV* sur les 500.

- critère de *fréquence* : 53 sujets sont dans *EV* (c.-à-d. 53 sujets dont la fréquence correspond à un pic par rapport aux fréquences des autres sujets).
- critère de *variation* : 67 sujets sont dans *EV* (c.-à-d. 67 sujets dont l'écart-type correspond à un pic par rapport aux écarts-types des autres sujets).
- critère de *Tf-Idf* : 74 sujets sont dans *EV* (c.-à-d. 74 sujets, dont la valeur de *Tf-Idf* correspond à un pic par rapport aux valeurs de *Tf-Idf* des autres sujets).

Tableau 7.VI – L'annotation des experts d'évènements détectés (en utilisant *LDA*) entre le 08 février et le 15 avril 2012 ($|EV| = 53$ (pour le critère fréquence), $|EV| = 67$ (pour le critère variation), $|EV| = 74$ (pour le critère *Tf-Idf*)). *EV* : *ensemble de sujets qui réfèrent à des évènements* ; $\overline{EV}$:*ensemble de sujets qui ne réfèrent pas à des évènements* ; **Nr** : *ensemble de sujets non reconnus*.

	Fréquence			**Écart-type**			***Tf-Idf***		
	EV	$\overline{EV}$	**Nr**	*EV*	$\overline{EV}$	**Nr**	*EV*	$\overline{EV}$	**Nr**
expert1	36	15	2	58	8	1	69	5	0
expert2	38	13	2	58	7	2	69	5	0
expert3	40	10	3	57	9	1	67	7	0
expert4	40	11	2	58	8	1	69	5	0
expert5	38	13	2	60	6	1	66	7	1
expert6	39	12	2	56	9	2	67	7	0
expert7	38	13	2	59	6	2	69	5	0
expert8	37	14	2	60	6	1	68	5	1
expert9	41	10	2	58	7	2	68	6	0
expert10	39	12	2	58	8	1	68.0	6	0
Moyenne	38.6	12.3	2.1	58.2	7.4	1.4	68	5.8	0.2
Précision	**0.73**			**0.87**			**0.92**		

Les résultats obtenus sont également évalués par 10 experts selon la même méthode expliquée dans la section 7.1.5. Selon les évaluations effectuées, le critère *Tf-Idf* est toujours le plus efficace pour déterminer les *EV* dans une période donnée.

7.3 Détection des évènements en regroupant les tweets similaires

Les méthodes présentées dans les deux sections précédentes se basent sur la présence des termes dans les tweets afin de déterminer les sujets de *EV*. Dans cette section, nous proposons une autre méthode qui consiste à regrouper les tweets similaires. Chaque *cluster* devrait contenir des tweets discutant le même sujet. Puisque le nombre de sujets n'est pas connu préalablement, nous avons utilisé un *ACI* qui précise automatiquement le nombre de clusters (voir section 5.4). Pour déterminer la similarité entre deux tweets nous avons utilisé la distance de Jaccard. Un tweet est représenté sous forme d'un vecteur de hashtags présents.

Nous testons deux variantes de *ACI* :

- ***tweet_one_cluster*** : Affecter un tweet *tw* au cluster C^* le plus similaire. ($C^* = \arg\max_{C_j \in C} \frac{1}{|C_j|} \sum_{tw \in C_j} Similarity(tw_i, tw)$) **et** qui dépasse un seuil défini manuellement.

- ***tweet_many_clusters*** : Un tweet peut appartenir à plus d'un cluster : nous affectons le tweet *tw* à tous les clusters C_i dont la similarité $\frac{1}{|C_i|} \sum_{tw \in C_i} Similarity(tw_j, tw)$ dépasse un seuil défini manuellement.

7.3.1 Pré-traitement

Dans la tâche de clustering, nous ne considérons que les tweets qui contiennent des hashtags. Cependant, les hashtags fréquemment utilisés peuvent provoquer du bruit car ils augmentent la similarité entre deux tweets même s'ils ne constituent pas de bons indices pour les comparer. Pour cela, nous avons supprimé les hashtags qui ont servi à extraire les *tweets* (p.ex *#tunisie*).

Après cette tâche, 82 400 (parmi les 147 397 tweets dans le corpus avec hashtags) *tweets* seront considérés dans le clustering. Les autres *tweets*, non considérés, ne contiennent plus de hashtags. À titre d'exemple, 43 546 (parmi les 64 997 tweets supprimés) des *tweets* contenaient uniquement le hashtag *#tunisie*. Tous ces *tweets* ont été supprimés après la tâche de pré-traitement.

7.3.2 Expansion des *tweets*

La mesure de *Jaccard* comptabilise les hashtags en commun entre deux tweets. Or, comme nous l'avons montré précédemment, un hashtag pourrait être écrit sous plusieurs formes et/ou reliés à d'autres hashtags. Ces hashtags similaires ne seront pas détectés par la mesure de *Jaccard*. Il serait, probablement, intéressant d'enrichir le tweet original avec d'autres hashtags similaires.

Pour étendre les *tweets*, nous nous sommes basés sur les résultats de regroupement obtenus dans le chapitre précédent, notamment ceux de *CoDBscan*. Chaque hashtag est remplacé, le cas échéant, par l'identifiant du *cluster* où il a été affecté. L'algorithme d'expansion proposé est décrit par l'algorithme 4.

Algorithm 4 L'algorithme d'expansion de tweet

Données : tweet tw_i à étendre et *Cluster_hashtags* (les *clusters* de hashtags obtenus par *CoDBscan*)
$tweet\_resultat \leftarrow \emptyset$ **// initialiser à l'ensemble vide**
for chaque hashtag h dans tw_i **do**
  $nombre\_cluster \leftarrow |Cluster\_hashtags_j : h \in Cluster\_hashtags_j|$ **// nombre de *cluster* de hashtag ou *h* est présent.**
  if $nombre\_cluster \neq 1$ **then**
    // *h* appartient à plus qu'un *cluster* où il n'appartient à aucun *cluster*.
    $tweet\_resultat \leftarrow tweet\_resultat \cup CodeSoundex(h)$
  else
    $tweet\_resultat \leftarrow tweet\_resultat \cup Cluster\_hashtags^*$ **//** $Cluster\_hashtags^*$ **est le seul *cluster* où *h* est présent**
  end if
end for

CoDBscan peut affecter certains hashtags à plus d'un seul *cluster*, dans ce cas on

parle d'un hashtag ambigu. Toutefois, l'expansion du tweet par tous les clusters concernés pourrait occasionner du bruit, car les *clusters* (de hashtags) portent généralement sur des sujets différents. Pour éviter ce problème, nous avons décidé de remplacer les hashtags portant une telle ambiguïté, seulement, par leurs *codes Soundex*. Cela, nous permet, au moins, de détecter ceux qui partagent des hashtags avec une faible variation orthographique et de même prononciation.

Comme ils couvrent un bon nombre de hashtags et qu'ils montrent une bonne précision, nous avons utilisé dans nos expérimentations les résultats obtenus par $CoDBscan_{npmiWithSndx}$ ($\lambda = 0.1$ et $\varepsilon = 0.6$) et $CoDBscan_{hyperWithSndx}$ ($\lambda = 0.1$ et $\varepsilon = 0.4$).

L'impact de l'expansion des tweets est évalué en testant *ACI* : sans expansion et avec expansion en utilisant les clusters de hashtags obtenus par $CoDBscan_{npmiWithSndx}$ et par $CoDBscan_{hyperWithSndx}$. Le tableau 7.VII montre le nombre de clusters de tweets obtenus sans et avec expansion. Un cluster doit contenir au moins deux tweets. Nous constatons que le nombre de clusters obtenus par *ACI* sans expansion est toujours plus grand que celui obtenu par les variantes avec expansion, de même concernant le nombre de tweets rejetés ou qui se trouvent seuls. Ceci est expliqué par le fait que *ACI* sans expansion est incapable de regrouper certains tweets similaires car ils ne partagent pas les mêmes hashtags.

Tableau 7.VII – Impact de l'expansion sur les résultats de *ACI*. **NC** : nombre de clusters ; **NTNR** : nombre de *tweets* non regroupés

	tweet_one_cluster		*tweet_many_clusters*	
	NC	**NTNR**	**NC**	**NTNR**
Sans expansion	3 931	4 296	4 559	4 402
Expansion avec $CoDBscan_{npmiWithSndx}$	3 015	3 077	3 369	3 115
Expansion avec $CoDBscan_{hyperWithSndx}$	3 305	3 305	3 213	3 293

7.3.3 Expérimentations

Chaque cluster obtenu par *ACI* est supposé contenir des tweets discutant le même sujet. Ainsi, chaque cluster représente un sujet. La fréquence d'un sujet s dans une journée j correspond au nombre de tweets, trouvés dans le cluster correspondant à s, envoyés dans la journée j.

Après avoir calculé la fréquence quotidienne de tous les sujets (représentés par des clusters de tweets) dans cette section, nous envisageons de ne pas sélectionner que les éléments de *EV*. Pour ce faire, nous avons suivi les mêmes étapes de validation utilisées dans les deux sections précédentes. Nous évaluons chaque variante selon les trois critères : *fréquence*, *variation* et *Tf-Idf*.

Le tableau 7.VIII montre la taille de *EV* pour chaque variante et pour les trois critères (*fréquence*, *variation* et *Tf-Idf*).

Tableau 7.VIII – Taille de *EV* obtenus pour chaque variante

	tweet_one_cluster			*tweet_many_clusters*		
	Fr	*Vr*	*Tf-Idf*	*Fr*	*Vr*	*Tf-Idf*
Expansion avec CoDBscan$_{npmiWithSndx}$	102	54	48	274	96	68
Expansion avec CoDBscan$_{hyperWithSndx}$	107	55	58	417	119	85

Nous avons constaté que la variante *tweet_many_clusters* de *ACI* a construit des clusters quasi-similaires. Beaucoup de clusters partageant un nombre important de tweets ont été créés. Pour cette raison, nous avons rejeté les résultats produits par cette variante et nous n'avons évalué que les résultats obtenus par la variante *tweet_one_cluster*.

Les tableaux 7.IX et 7.X montrent les résultats de l'évaluation de nos experts. Le tableau montre le nombre de *EV* sélectionnés pour chaque variante et pour les trois critères (*fréquence, variation* et *Tf-Idf*). Chaque sujet est représenté par les 20 hashtags les plus fréquents dans le cluster.

Comme avec les deux autres méthodes, le critère de *Tf-Idf* est toujours le plus performant pour déterminer l'ensemble *EV*. Tous les résultats obtenus par les trois méthodes montrent que les sujets les plus discutés (fréquents) ne correspondent pas aux sujets importants.

Tableau 7.IX – L'annotation des experts d'évènements détectés entre le 08 février et le 15 avril 2012 ($|EV|$ = 98 (pour le critère fréquence), $|EV|$ = 52 (pour le critère variation), $|EV|$ = 46 (pour le critère *Tf-Idf*)) après l'expansion des tweets par les résultats de $CoDBscan_{npmiWithSndx}$. *EV* : *ensemble de sujets qui réfèrent à des évènements* ; $\overline{EV}$:*ensemble de sujets qui ne réfèrent pas à des évènements* ; **Nr** : *ensemble de sujets non reconnus*.

	Fréquence			**Écart-type**			***Tf-Idf***		
	$\|EV\|$	$\|\overline{EV}\|$	$\|Nr\|$	$\|EV\|$	$\|\overline{EV}\|$	$\|Nr\|$	$\|EV\|$	$\|\overline{EV}\|$	$\|Nr\|$
expert1	59	39	0	44	8	0	44	2	0
expert2	60	37	1	41	10	1	42	4	0
expert3	60	37	1	40	10	2	43	3	0
expert4	60	38	0	42	8	2	44	2	0
expert5	61	37	0	43	8	1	43	4	1
expert6	59	38	1	42	9	1	42	4	0
expert7	59	39	0	43	8	1	43	3	0
expert8	59	38	1	40	10	2	43	2	1
expert9	58	39	1	41	10	1	43	3	0
expert10	60	38	0	40	10	2	42	4	0
Moyenne	59,5	38,0	0,5	41,6	9,1	1,3	42,9	3,1	0,2
Précision	0,61			0,8			0,93		

Tableau 7.X – L'annotation des experts d'évènements détectés entre le 08 février et le 15 avril 2012 ($|EV|$ = 104 (pour le critère fréquence), $|EV|$ = 51 (pour le critère variation), $|EV|$ = 46 (pour le critère *Tf-Idf*)) après l'expansion des tweets par les résultats de $CoDBscan_{hyperWithSndx}$. *EV* : *ensemble de sujets qui réfèrent à des évènements* ; $\overline{EV}$:*ensemble de sujets qui ne réfèrent pas à des évènements* ; **Nr** : *ensemble de sujets non reconnus*.

	Fréquence			**Écart-type**			***Tf-Idf***		
	$\|EV\|$	$\|\overline{EV}\|$	$\|Nr\|$	$\|EV\|$	$\|\overline{EV}\|$	$\|Nr\|$	$\|EV\|$	$\|\overline{EV}\|$	$\|Nr\|$
expert1	58	46	0	38	13	0	44	2	0
expert2	57	46	1	37	13	1	44	2	0
expert3	58	46	0	35	16	0	42	4	0
expert4	56	48	0	37	14	0	42	4	0
expert5	56	48	0	37	14	0	43	4	1
expert6	56	48	0	34	16	1	43	3	0
expert7	57	47	0	35	16	0	43	3	0
expert8	57	47	0	36	15	0	44	3	1
expert9	58	46	0	35	15	1	44	2	0
expert10	57	46	1	38	13	0	42	4	0
Moyenne	56,8	48,4	1,4	35,0	14,5	1,5	43,1	3,1	0,2
Précision	0,54			0,69			0,94		

7.4 Conclusion

Dans ce chapitre nous avons testé trois méthodes afin de déterminer l'ensemble *EV* dans une période donnée. Les deux premières méthodes considèrent qu'un sujet est représenté par un ensemble de termes (hashtags, mots). Un tweet est assigné à un sujet si au moins un terme de ce dernier est trouvé dans le tweet.

La première méthode utilise les *clusters* obtenus par la tâche de normalisation. Avec cette méthode, nous avons obtenu une bonne précision (nombre de vrais évènements dans *EV*). Mais le problème de cette méthode est que le sujet est souvent représenté par plusieurs *clusters*, ceci conduit à l'obtention de sujets similaires dans *EV*. Ce problème provient du le fait que le sujet n'est représenté que par les termes (hashtags) qui ont une prononciation similaire. Ce problème a été atténué avec la deuxième méthode qui utilise les *clusters* obtenus par *LDA* pour déterminer *EV*. Comme *LDA* se base sur des techniques statistiques, et non sur la prononciation, pour déterminer les mots sémantiquement similaires, nous avons regroupé dans le même document les tweets partageant le même code *Soundex* d'un hashtag ce qui aide à détecter des relations entre des mots qui n'ont pas cooccurré auparavant. Toutefois, le problème avec *LDA* est, notamment, le choix du nombre de sujets discutés, ce qui est difficile. Le vocabulaire utilisé dans notre corpus est trop riche [7]. Même si nous avons utilisé des techniques de stemming et de lemmatisation pour normaliser les termes, nous n'avons pas réussi à reconnaitre certains mots, tels que des mots vides censés être supprimés.

Dans la troisième méthode proposée, nous avons regroupé les tweets similaires afin d'atteindre notre objectif. La similarité entre les tweets est basée sur les hashtags qui constituent un indice important pour déterminer le(s) sujet(s) discuté(s) dans le tweet. Le regroupement a été effectué par un *ACI* qui détermine lui-même le nombre de clusters (*cluster* de tweets similaires). Pour améliorer le regroupement, nous avons proposé un algorithme d'expansion qui étend les tweets avec d'autres hashtags.

Suite à nos observations et celles de nos experts, nous pouvons considérer que cette méthode est la plus efficace pour déterminer l'ensemble *EV* à partir de notre corpus.

[7] beaucoup de nouveaux mots inventés par les utilisateurs et qui se n'existent pas dans des ressources

Contrairement à la première méthode, presque tous les sujets dans *EV* sont distincts. Aussi, nous n'étions pas obligés à fixer, au préalable, le nombre de sujets ou clusters à obtenir.

CHAPITRE 8

CONCLUSION ET TRAVAUX FUTURS

Dans cette thèse, nous nous sommes intéressés à la détection des sujets qui ont stimulé l'intérêt des utilisateurs dans une période donnée. Pour ce faire, nous avons eu recours aux données de médias sociaux qui constituent un excellent endroit pour les internautes pour partager des idées, donner des avis et diffuser des nouvelles. Ces aspects conduisent à l'accumulation d'une énorme quantité de données qui exposent les préoccupations des utilisateurs en temps-réel. Parmi tous les services de médias sociaux, nous avons opté pour Twitter étant donné sa popularité et l'accessibilité de son contenu. Notre corpus dans ce travail est composé de tweets qui portent sur la Tunisie. Ces tweets comportent plusieurs particularités telles que les abréviations, fautes d'orthographe, mots arabes écrits avec des alphabets latins et chiffres, etc. Ce corpus a été collecté d'une manière continue entre les 08 février et 15 avril 2012. Nous avons supposé qu'un sujet stimulait l'intérêt des utilisateurs au cours d'une période donnée, si ses caractéristiques apparaissaient brusquement à un ou plusieurs moments de la période en question. Les caractéristiques d'un sujet sont les termes qui le représentent. Dans cette thèse, nous nous sommes basé principalement sur les hashtags qui constituent de bons indices pour déterminer le (s) sujet (s) d'un tweet. Notre première tâche dans le processus de détection d'évènements consiste à regrouper les termes similaires et/ou discutant le même sujet :

- Nous avons proposé des méthodes permettant de regrouper les hashtags avec une prononciation semblable. Pour ce faire, nous avons utilisé l'algorithme phonétique *Soundex*, que nous avons adapté au dialecte tunisien, qui attribue le même code aux termes qui se prononcent de la même façon. Comme un même hashtag peut être écrit soit en alphabet latin ou en alphabet arabe, nous avons proposé un algorithme qui permet de regrouper un hashtag écrit en arabe avec ses correspondants en alphabet latin.

- La tâche de normalisation permet de regrouper les hashtags avec une prononciation similaire, cependant elle est incapable de regrouper les hashtags référant à un même sujet et qui ne se prononce pas de la même façon. Pour regrouper ces hashtags, nous avons eu recours à l'algorithme de *regroupement DBscan* qui permet de regrouper les hashtags indépendamment de la langue avec laquelle ils sont écrits. Pour déterminer la similarité entre les hashtags nous avons profité de certaines informations trouvées dans les tweets telles que : la cooccurrence et les hyperliens partagés entre les hashtags. Nous avions constaté que la version originale de *DBscan* n'était pas efficace pour notre contexte, nous l'avons donc adapté et nous avons proposé d'autres variantes.

- Nous avons également essayé de regrouper les mots dans les tweets ; pour ce faire, nous avons utilisé l'algorithme *LDA* qui permet de regrouper les mots qui portent sur la même chose. Nous avons regroupé dans un seul document les tweets partageant les mêmes hashtags afin d'améliorer les résultats. Aussi, nous avons utilisé un corpus parallèle de SMS français pour normaliser les mots vides en français qui sont écrits incorrectement.

L'absence de données de référence nous a obligés à évaluer manuellement les résultats de nos méthodes de regroupement de hashtags. L'évaluation a été faite par deux experts familiarisés avec les évènements déroulés en Tunisie. Selon les évaluations, nous avons obtenu une précision qui dépassait souvent 90 %.

Par la suite, nous avons utilisé les *clusters* obtenus précédemment pour déterminer les sujets saillants ou les évènements. Pour ce faire, nous avons proposé deux méthodes. Dans la première méthode, nous avons utilisé les *clusters* obtenus par la normalisation des hashtags et *LDA* et nous avons considéré que chaque *cluster* correspondait à un sujet. Un tweet porte sur un sujet si au moins l'un de ses termes est présent. Cette tâche a permis d'obtenir la fréquence quotidienne de chaque sujet.

Pour la deuxième méthode, nous avons regroupé les tweets similaires en utilisant un algorithme de *regroupement* incrémental qui détermine automatiquement le nombre de *clusters*. Chaque *cluster* correspond à un sujet. La fréquence quotidienne d'un sujet

correspond au nombre de tweets s'y rapportant envoyés quotidiennement.

Pour les deux méthodes, nous avons utilisé la fréquence quotidienne des sujets pour déterminer les dates saillantes.

Comme tous les sujets ne sont pas nécessairement des évènements, nous avons adapté la méthode de Palshikar (2009) pour sélectionner les sujets saillants en évaluant trois critères : fréquence, écart-type, *Tf-Idf*. Pour chaque critère, nous avons demandé à 10 experts de déterminer, à l'aide d'une application web, les bons évènements parmi les sujets de ceux que nous avions identifiés. Nous avons constaté que le critère *Tf-Idf* est souvent le plus adéquat pour sélectionner le *EV*.

À l'aide des méthodes proposées dans ce travail, nous avons réussi à atteindre notre objectif qui consiste à déterminer les sujets saillants au cours d'une période.Ceci malgré les difficultés auxquelles nous avons été confrontés tels que le peu de ressources mises à notre disposition et l'absence de données de référence pour évaluer automatiquement nos résultats.

Dans la section suivante, nous présentons certaines tâches que nous aurions voulu faire si nous avions eu plus de temps et nous proposons quelques perspectives de travail.

8.1 Travaux Futurs

Les expérimentations présentées dans cette thèse sont effectuées sur des données collectées entre février et avril 2012. Toutefois, nous avons recueilli un autre corpus (environ 600 000 tweets) entre octobre 2012 et janvier 2013 portant également sur la Tunisie. Bien que nos méthodes soient automatiques, mais nous ne les avons pas appliquées sur ce corpus, car en l'absence de référence les évaluations des résultats sont coûteuses en temps.

Nous prévoyons également tester ces méthodes sur d'autres types de données. Nous avons déjà commencé à extraire des données envoyées à partir du Québec. Cette fois-ci nous avons profité de paramètres de géolocalisation pour extraire ces tweets. Comme mentionné précédemment, ces paramètres ne sont pas fonctionnels pour le cas de la Tunisie. L'option de géolocalisation nous a évité de construire l'ensemble de mots-clés

adéquat pour extraire les données et elle nous a permis d'extraire un grand nombre de tweets. Entre février et juin 2014, nous avons réussi à recueilli environ 4 millions de tweets.

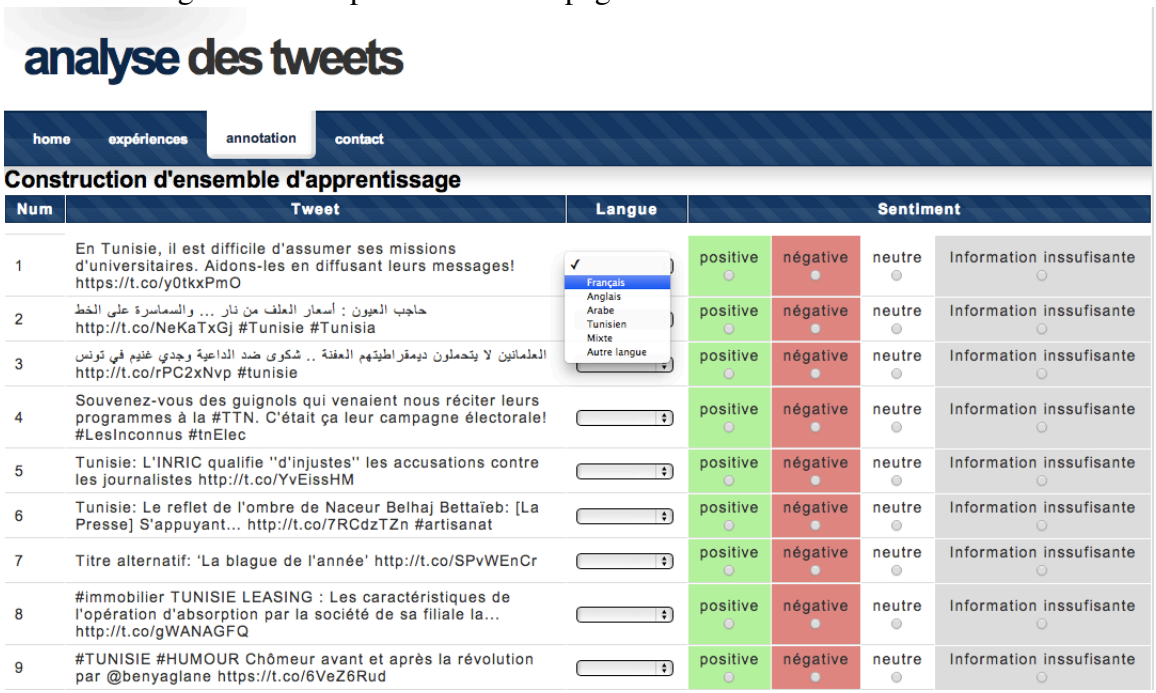
Sauf pour les méthodes utilisées dans la tâche de normalisation, notre système peut en principe fonctionner sur le corpus du Québec. En travaillant sur ce corpus qui devrait contenir des tweets écrits en français et en anglais, il serait préférable d'utiliser un autre *Soundex* car celui présenté dans ce travail est adapté au texte écrit par des Tunisiens, mais il serait inutile de translittérer les hashtags pour un corpus écrit en alphabet latin. Toutefois, si nous travaillions sur l'anglais et le français nous pourrions profiter des ressources linguistiques supportant ces langues.

Les méthodes proposées pour déterminer les termes similaires distinguaient entre les mots et les hashtags trouvés dans les tweets. Aucune méthode ne servait à la création des *clusters* contenant à la fois des mots et des hashtags similaires. L'intuition derrière nos méthodes est de vérifier l'apport de chacun de ces éléments (mots et hashtags) pour déterminer les évènements au cours d'une période. Cependant, il est évident qu'il existe des hashtags et des mots sémantiquement similaires. Ainsi, nous pourrions proposer une méthode permettant de regrouper les hashtags et les mots.

Les résultats présentés dans ce travail prouvent que les hyperliens sont des éléments importants pour déterminer la similarité entre les hashtags. Rappelons que nous avons avons considéré que plus les hashtags partageant les mêmes hyperliens plus ils sont similaires. Néanmoins, tout comme il y a des termes similaires il y a aussi des hyperliens similaires (référant à un même sujet). Il serait donc intéressant de regrouper les hyperliens similaires ce qui permettait d'améliorer le regroupement de hashtags. Aussi, ils peuvent être utilisés comme indices pour identifier les évènements où chaque groupe d'hyperliens réfère à un sujet.

Dans cette thèse, nous n'avons pas distingué entre les tweets discutant d'un évènement et les autres. Comme extension, nous avons imaginé développer un classifieur permettant de déterminer si un tweet porte sur un évènement ou non. Pour ce faire, nous pourrions nous baser sur d'autres aspects que le seul contenu du tweet, p.ex tels que les utilisateurs mentionnés dans le tweet, une réponse ou non, un retweet ou non, etc.

Figure 8.1 – Capture d'écran de page d'annotation de sentiments



Jusqu'ici, un évènement est décrit par ses dates saillantes et les termes (mots ou hashtags) qui le représentent. Même des experts du dialecte peuvent trouver des difficultés à identifier un sujet à partir de ses termes. Pour cette raison, nous pourrions générer un résumé pour chaque sujet dans *EV* constitué par un ensemble de tweets décrivant mieux le sujet. Nous devrions sélectionner des tweets compréhensibles par tout le monde et éviter les tweets écrits en dialecte tunisien et/ou qui contiennent des fautes d'orthographe. Pour atteindre cet objectif, il serait donc préférable de construire le résumé avec des tweets écrits par des utilisateurs qui travaillent dans des journaux ou magazines. Ces utilisateurs écrivent généralement des tweets bien structurés. Ainsi, notre défi consiste à identifier ces utilisateurs et à choisir les tweets qui décrivent l'évènement en question.

Notre thèse a été initiée dans le cadre d'une collaboration avec la société *MediaBadger*[1]. Notre objectif était le développement d'un système automatique pour l'analyse de sentiment d'un sujet particulier à partir de microblogs. Le rôle de médias sociaux dans les révolutions arabes et nos compétences qui nous permettent de comprendre les textes

[1] `http://www.mediabadger.com/`

écrits par les Tunisiens, nous avaient encouragé de traiter les tweets portant sur la Tunisie. L'analyse de sentiment nécessite des données d'apprentissage et/ou une liste de *traits* (termes) permettant de déterminer la polarité (positive, négative ou neutre) d'un texte. Toutefois, ces ressources ne sont pas disponibles pour le dialecte tunisien.

Pour cela, nous avions décidé de créer nos propres ressources. Nous avions demandé à des experts (des étudiants tunisiens au Canada) d'annoter, via notre site web[2] (voir figure 8.1), des tweets portant sur la Tunisie dont la plupart sont écrits en dialecte tunisien. L'expert doit lire le tweet et déterminer sa polarité (positive, négative ou neutre). Jusqu'à l'écriture de cette thèse, nous avions réussi à annoter plus que 4 000 tweets. Cependant, cette collaboration a été interrompue à cause des problèmes de financement de l'entreprise.

Ce corpus pourrait permettre une extension de notre système pour qu'il puisse déterminer l'opinion publique de chaque évènement détecté. Par exemple, *Noël* est un heureux évènement tandis qu'un tremblement de terre est un triste évènement. Notre système devra déterminer la proportion de chaque polarité (positive, négative ou neutre) dans l'ensemble de tweets qui représentent l'évènement en question.

Nous prévoyons augmenter notre corpus à environ 10 000 tweets annotés puis le rendre disponible à la communauté.

8.2 Conclusion

À la première partie de ce chapitre, nous avons rappelé de différentes étapes et méthodes proposées dans ce travail afin d'identifier les sujets saillants durant une période. Malgré les particularités des textes disponibles dans les médias sociaux, notamment, les textes écrits dans un dialecte particulier, nous avons réussi à développer un système robuste.

Par la suite, nous avons présenté des alternatives que nous aurions pu développer avec plus de temps. Finalement, nous avons proposé des extensions de notre système.

[2] `http://rali.iro.umontreal.ca:8080/dridihou/`

BIBLIOGRAPHIE

J. Allan. *Topic detection and tracking : event-based information organization*, volume 12. Kluwer Academic Publishers, 2002.

J. Allan, J.G. Carbonell, G. Doddington, J. Yamron et Y. Yang. Topic detection and tracking pilot study final report. 1998.

L. Barbosa et J. Feng. Robust sentiment detection on Twitter from biased and noisy data. Dans *Proceedings of the 23rd International Conference on Computational Linguistics : Posters*, pages 36–44. Association for Computational Linguistics, 2010.

H. Becker, M. Naaman et L. Gravano. Beyond trending topics : Real-world event identification on Twitter. Dans *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM11)*, 2011.

A. Bermingham et A. F. Smeaton. Classifying sentiment in microblogs : is brevity an advantage ? Dans *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1833–1836. ACM, 2010.

D.M. Blei, A.Y. Ng et M.I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

J. Bollen, H. Mao et X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

G. Bouma. Normalized (pointwise) mutual information in collocation extraction. Dans *Proceedings of the Biennial GSCL Conference*, pages 31–40, 2009.

D. Chakrabarti et K. Punera. Event summarization using tweets. Dans *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 66–73, 2011.

K.W. Church et P. Hanks. Word association norms, mutual information, and lexicography. Dans *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 76–83. Association for Computational Linguistics, 1989.

E. Clark et K. Araki. Text normalization in social media : progress, problems and applications for a pre-processing system of casual English. *Procedia-Social and Behavioral Sciences*, 27:2–11, 2011.

A. Go, R. Bhayani et L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

J. Halpern. The challenges and pitfalls of arabic romanization and arabization. Dans *Proc. Workshop on Comp. Approaches to Arabic Scriptbased Lang*, 2007.

L. Jiang, M. Yu, M. Zhou, X. Liu et T. Zhao. Target-dependent Twitter sentiment classification. *Proc. 49th ACL : HLT*, 1:151–160, 2011.

W. Jianshu et L. Bu-Sung. Event detection in Twitter. Dans Lada A.A, Ricardo A.B.Y et Scott C., éditeurs, *ICWSM*, pages 401–408. The AAAI Press, 2011.

G. Kaiser. *A friendly guide to wavelets*. Springer, 2011.

J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

A. Kontostathis, L.M. Galitsky, W.M. Pottenger, S. Roy et D.J. Phelps. A survey of emerging trend detection in textual data mining. Dans *Survey of Text Mining*, pages 185–224. Springer, 2004.

H. Kwak, C. Lee, H. Park et S. Moon. What is Twitter, a social network or a news media ? Dans *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

V. Lampos et N. Cristianini. Tracking the flu pandemic by monitoring the social web. Dans *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*, pages 411–416. IEEE, 2010.

P. Langlais, P. Drouin, A. Paulus, E.R. Brodeur et F. Cottin. Texto4Science : a Quebec french database of annotated short text messages. Dans *Proceedings of Language Resources and Evaluation Conference (LREC) 2012*.

V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

J. Makkonen, H. Ahonen-Myka et M. Salmenkivi. Simple semantics in topic detection and tracking. *Information Retrieval*, 7(3):347–368, 2004.

A.K. McCallum. Mallet : A machine learning for language toolkit. http ://mallet.cs.umass.edu, 2002.

F. Morstatter, J. Pfeffer, H. Liu et K. M. Carley. Is the sample good enough ? comparing data from Twitter's streaming API with Twitter's firehose. *Proceedings of ICWSM*, 2013.

B. OConnor, R. Balasubramanyan, B.R. Routledge et N.A. Smith. From tweets to polls : Linking text sentiment to public opinion time series. Dans *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 122–129, 2010.

O. Ozdikis, P. Senkul et H. Oguztuzun. Semantic expansion of tweet contents for enhanced event detection in twitter. Dans *ASONAM*, pages 20–24. IEEE Computer Society, 2012a. ISBN 978-0-7695-4799-2.

O. Ozdikis, P. Senkul et H. Oguztuzun. Semantic expansion of tweet contents for enhanced event detection in Twitter. Dans *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 20–24. IEEE, 2012b.

G.K. Palshikar. Simple algorithms for peak detection in time-series. Rapport technique, TRDDC, 2009.

S. Petrovic, M. Osborne et V. Lavrenko. Streaming first story detection with application to Twitter. Dans *HLT-NAACL*, pages 181–189. The Association for Computational Linguistics, 2010. ISBN 978-1-932432-65-7.

R.C. Russell. Soundex coding system. *United States Patent*, (1,261,167), 1918.

T. Sakaki, M. Okazaki et Y. Matsuo. Earthquake shakes Twitter users : real-time event detection by social sensors. Dans *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

D. A. Shamma, L. Kennedy et E. Churchill. Statler : Summarizing media through short-message services. Dans *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW10)*, pages 551–552, 2010.

L. Shao et H.T. Ng. Mining new word translations from comparable corpora. Dans *Proceedings of the 20th international conference on Computational Linguistics*, page 618. Association for Computational Linguistics, 2004.

R. Sproat, A.W. Black, S. Chen, S. Kumar, M. Ostendorf et C. Richards. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333, 2001.

J. Sutton, L. Palen et I. Shklovski. Backchannels on the front lines : Emergent uses of social media in the 2007 southern california wildfires. Dans *Proceedings of the 5th International ISCRAM Conference*, pages 624–632. Washington, DC, 2008.

K. Zhang, J. Zi et L.G. Wu. New event detection based on indexing-tree and named entity. Dans *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 215–222. ACM, 2007.