Event Detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering

Georgiana Ifrim
Insight Centre for Data Analytics
University College Dublin
Dublin, Ireland
georgiana.ifrim@insight-centre.org

Bichen Shi
Insight Centre for Data Analytics
University College Dublin
Dublin, Ireland
bichen.shi@insight-centre.org

Igor Brigadir
Insight Centre for Data Analytics
University College Dublin
Dublin, Ireland
igor.brigadir@insight-centre.org

ABSTRACT

Twitter has become as much of a news media as a social network, and much research has turned to analyzing its content for tracking real-world events, from politics to sports and natural disasters. This paper describes the techniques we employed for the SNOW Data Challenge 2014, described in [16]. We show that aggressive filtering of tweets based on length and structure, combined with hierarchical clustering of tweets and ranking of the resulting clusters, achieves encouraging results. We present empirical results and discussion for two different Twitter streams focusing on the US presidential elections in 2012 and the recent events about Ukraine, Syria and the Bitcoin, in February 2014.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Event Detection, Twitter, Social Media, Digital Journalism, News Aggregation

1. INTRODUCTION

Micro-blogging platforms such as Twitter have emerged in recent years, creating a radically new mode of communication between people. Every day, 500 million users send more than 500 million tweets (as of end 2013) [10], on every possible topic. Interactions and communication in Twitter often reflect real-world events and dynamics, and important events like elections, disasters, concerts, and football games can have immediate and direct impact on the volume of tweets posted. Because of its real-time and global nature, many people use Twitter as a primary source of news content, in addition to sharing daily life, emotion and thoughts.

Journalists also increasingly adopt social media as professional tools and are gradually altering their processes of news selection and presentation [11, 18]. They use Twitter to monitor the newsworthy stories that emerge from the crowd, and to find user-generated content to enrich their stories. However, it is very hard for a person to spot the useful information in Twitter without being overwhelmed by an endless stream of redundant tweets.

As a response to this problem and the SNOW Data Challenge 2014, we propose a system to detect novel, newsworthy topics/events as they are published on Twitter. Provided with a Twitter stream that is initially filtered by a list of seed terms corresponding to known events (e.g., Ukraine) and possibly a list of user ids, the system automatically mines the social stream, to provide a set of headlines and complementary information (photo and tweets) that summarize the topics for a number of time slots of interest. Although Topic Detection and Tracking [2] has been well-studied for static document corpora, in the social media context there are a few new factors that make the problem more challenging, e.g., different language styles between Twitter and traditional news media, the fragmented and possibly ambiguous nature of tweets due to their 140 character length constraint, the high amount of noise in the user-generated content and the real-time data processing aspect.

In this paper, we present our topic detection approach: a combination of aggressive data pre-processing, hierarchical clustering of tweets, time-dependent n-gram and cluster ranking and headlines re-clustering. We analyze how factors such as event type, data pre-processing and parameters in the framework affect the quality of topic extraction results. The evaluation simulates a real-world application scenario, where the system works on the data of the live tweet stream and produces (close to real-time) detected topics in each user-specified time window (e.g., new headlines for every 15 minutes). The selected datasets cover the US presidential Elections (2012) and recent events in Ukraine and Syria (2014).

2. RELATED WORK

Recently [1] has compared several techniques for event detection in Twitter, and promoted a technique based on term clustering for obtaining trending topics. The six compared techniques in [1] fit into two main categories: document-clustering versus term-clustering, where a cluster represents a potential topic of interest. These approaches can be further categorized into three different classes: probabilistic models (e.g., Latent Dirichlet Allocation (LDA)), classical

SNOW WWW Workshop 2014 Korea

Topic Detection and Tracking (e.g., Document-Pivot Topic Detection (FSD)) and feature-pivot methods (e.g., n-gram clustering (BNgram)).

LDA [5] is a topic model that associates with each document a probability distribution over topics, which are in turn distributions over words. Every document is considered as a bag of terms, and the topic distribution per document as well as the term distribution per topic are estimated through Bayesian inference. According to results in [1], LDA models can capture stories happening during events with narrow topical scope, while their performance can be dramatically lower when considering more *noisy* events.

FSD [17] is designed to detect the first document discussing a topic in a large corpus via document clustering. It works with a document-term matrix, where coordinates represent the frequency of a particular term in a document, and documents are clustered by cosine similarity of their $tf - idf$ term representation. FSD uses Locality Sensitive Hashing to rapidly retrieve the nearest neighbor of a document. Although the initial topic recall of plain FSD is not very high, it can significantly improve when employing document aggregation [1] via two stage clustering, to avoid initial topic fragmentation (i.e., the same topic being discussed by several clusters).

As proposed in [1], BNgram is a n-grams feature-pivot method that clusters terms rather than documents, where the distance between terms is defined by the proportion of documents where two terms co-occur. BNgram extracts topics in each time slot, and a time-dependent ranking is introduced to penalise topics that began in the past and are still popular in the present, via the use of a term burstiness score ($df - ifd_t$). [13] has compared the performance when using different types of n-grams and found that the 3- and 4-grams gave similar results and were 3 times better than using unigrams. BNgram has good performance on topic recall as well as keywords recall/precision, however, considering more top topics and topic/time aggregation does not improve topic recall.

Other than the six methods compared in [1], a keyword-lifecycle event detection framework was recently introduced in [14], in which a keyword's standard behavior is modeled by its frequency and its average daily behavior. An event is detected when a keyword's frequency is abnormal. 80% of the strong earthquakes are detected by this framework, and its false positive rate is very low. The Window Variation Keyword Burst Detection [7] is another recent topic detection method.

Building on recent work, we propose an approach based on tweet-clustering combined with a few layers of filtering, aggregation and ranking in order to deliver an efficient topic detection method. Our choice of tweet (vs term) clustering is based on the following observations: (1) tweet clustering methods have shown high recall, in particular when allowing a higher number of topics to be retrieved; (2) tweets are the main unit of content, lending themselves naturally to meaningful and human-readable news-like headlines, while term-clustering approaches have to deal with the challenge of re-creating a meaningful unit of content (e.g., swapping the order of terms in a cluster can change the meaning of a headline-topic); (3) we can introduce various tweet-importance metrics for re-weighting the retrieved tweet-clusters, e.g., up-weighting tweets from trustworthy or high clout sources, such as journalists.

3. DATA CHALLENGE SETUP

Details of the SNOW Data Challenge can be found in [16].

4. METHOD PROPOSED

The main approach behind our results for the data challenge is based on: (1) Aggressive tweet and term filtering, to remove noisy tweets and vocabulary; (2) Hierarchical clustering of tweets, dynamic dendrogram cutting and ranking of the resulting clusters, to obtain topics.

We describe our method in detail in the following subsections. For collecting the Twitter stream we used code provided by the SNOW challenge organizers [16] based on the Twitter4J API[1]. For all other development (e.g., data pre-processing, clustering, ranking, producing final topics), we have used Python2.7 and available python libraries. We chose Python due to the ease of development and its available range of powerful libraries (e.g., *scipy, numpy, sklearn*). In particular for tweet-NLP, e.g., named entity recognition, we have used a Python wrapper (*CMUTweetTagger* library [9]), and for efficient hierarchical clustering of tweets, we have used the *fastcluster* library [15]. Our code for topic detection is available online from `https://github.com/heerme`.

4.1 Data Collection

We worked with two different Twitter streams, one about the US presidential elections in 2012, collected between 6 Nov 2012, starting at 23:30, and ending on 7 Nov 2012, at 6:30, and another collected starting on 25 Feb 2014, at 17:30 and ending on 26 Feb 2014, at 18:15. The first stream was collected starting from tweet ids, and had each tweet in the form of a text line, containing the tweet GMT time, unix time stamp, id, user name, the text of the tweet, and whether the tweet is a retweet or not. There were 1,084,200 (252MByte), English and non-English tweets in this stream. In order to extract the user mentions, hashtags and urls from the text of the tweet, we used the twitter-text-python[2] library. For the second stream, the collected data is in JSON[3] format, meaning each line of the output stream is a tweet encoded as a JSON object. This consisted of 1,088,593 raw tweets (4.37GByte), out of which we only used 943,175 english tweets (3.87GByte), filtered using the *lang='en'* field of the tweet JSON object. We further processed each JSON object to extract, for each tweet, only the date, tweet id, text, user mentions, hashtags, urls and media urls, to a text file for faster processing (240MByte). For re-tweets, we replace the text of the re-tweet with the original text of the tweet that was re-tweeted (although we only do this for the tweets in JSON format, since the original tweet text is included in the JSON object). We use this text file, with one tweet per line, for all our experiments.

4.2 Data Pre-processing

An important part of our method is data pre-processing and filtering. For each tweet, we pre-process the text as follows. We normalize the text to remove urls, user mentions and hashtags, as well as digits and other punctuation. Next, we tokenize the remaining clean text by white space, and remove stop words. In order to prepare the tweet corpus, in each time window, for each tweet, we first append the user

[1] http://twitter4j.org/en/index.html
[2] `https://github.com/ianozsvald/twitter-text-python`
[3] See `https://dev.twitter.com/docs/entities` for details

mentions, the hashtags and the resulting clean text tokens. We check the structure of the resulting tweet, and filter out tweets that have more than 2 user mentions or more than 2 hashtags, or less than 4 text tokens. The idea behind this structure-based filtering is that tweets that have many user mentions or hashtags, but lack enough clean text features, do not carry enough news-like content, or are generally very noisy. This step filters many noisy tweets. For example, for the 15 minute time window, starting on 25 Feb 2014, at 18:00, and ending at 18:15, there are 12,589 raw tweets, out of which the first filtering step (that checks the length and structure of tweets) keeps only 9,487. Our initial tweet window corpus contains the above filtered tweets.

The next step is concerned with vocabulary filtering. For each time window, from the window tweet corpus, we create a (binary) tweet-term matrix, where we remove user mentions (but keep hashtags), and the vocabulary terms are only *bi-grams* and *tri-grams*, that occur in at least a number of tweets, where the minimum is set to 10 tweets, and the maximum threshold is set based on the window corpus length, to $max(int(len(window\_corpus) * 0.0025), 10)$. This threshold does not grow very quickly, for example, for 10,000 tweets, the term should occur in at least 25 tweets to be selected in the vocabulary. The idea behind this filtering step, is that clusters should gather enough tweets to be considered a topic at all (e.g., at least 25 tweets in 10,000 tweets should discuss an event). For the above example, the term filtering step reduces the vocabulary to 670 terms, therefore we now have a matrix with 9,487 tweets by 670 terms. In the next filtering step, we reduce this matrix to only the subset of rows containing at least 5 terms (tweets with at least 5 tokens from the vocabulary). This step is meant to remove out-of-vocabulary tweets, as well as tweets that are too short to be meaningfully clustered. We varied the parameters for filtering tweets and terms, and noticed that the above chosen values were stable with regards to the topics produced. This third filtering step further reduces the original tweet-by-term matrix to 2,677 tweets and 670 terms, effectively using only 20% of the original collection of raw tweets. We have found that for Twitter streams where the language information is not available, e.g., for the 2012 US presidential elections stream, it is much faster to filter tweets and terms as above, therefore getting rid of most non-english tweets, than to apply a language identification library.

4.3 Hierarchical Clustering of Tweets

In this section we give the detailed steps for our method.

- **Computing tweet pairwise distance.** We compute tweet pairwise distances and a hierarchical clustering on the filtered tweet-by-term matrix. For pairwise distances we scale and normalize the tweet-term matrix, and use *cosine* as a metric. Our experiments showed that by using euclidean distance we achieved similar results. We use the *sklearn* and *scipy* python libraries for computing distances and the tweet-term matrix.

- **Computing hierarchical clustering.** For computing a hierarchical clustering, we use the $fastcluster$ library [15] that can efficiently deal with thousands of tweets/terms. The idea behind tweet clustering is that tweets belonging to the same topic will cluster together, and thus we can consider each cluster as a detected topic.

- **Cutting the dendrogram.** Finally, we cut the resulting dendrogram at a 0.5 distance threshold. This threshold can control how tight or loose we require our final clusters to be, without having to provide the number of clusters expected a-priori, e.g., as for k-means or other popular clustering algorithms. A higher threshold would result in looser clusters, that potentially collate different topics in the same cluster. A lower threshold would result in tighter and cleaner clusters, but potentially lead to too much topic fragmentation, i.e., the same topic would be reflected by lots of different clusters. We found that a value of 0.5 works well for our method.

- **Ranking the resulting clusters.** Once we obtain clusters with the above procedure, we assign a score to each cluster and rank them based on that score. A first attempt was to score and rank clusters by size, therefore allowing clusters with a lot of tweets to rank first as trending topics. This results in topics that tend to be more casual and are unlikely to make the news headlines (e.g., *This is what happens when you put two pit bulls in a photo booth*), as we show in our evaluation section. Additionally, topics tend to get frequently repeated for several time windows, since we do not consider potential term/topic burstiness in each time window with respect to the previous time windows.

  Next, we introduce term weighting, based on the frequency in the time window, as well as boosting named entities. For the frequency based weight, we use the $df - idf_t$ formula from [1], that discounts the term-frequency in the current time window using the average frequency in the previous t time windows. The formula is shown in Equation 1.

$$df - idf_t = \frac{df_i + 1}{\log\left(\frac{\sum_{j=i}^{t} df_{i-j}}{t} + 1\right) + 1} \quad (1)$$

  Setting the parameter t controls how much the history should affect the current weight of a term. We set $t = 4$ in our approach, in order to allow for hourly updates (where a time window is set to 15 minutes). Note the *log* in the denominator, allowing the current document frequency to have more weight than the previous/historical average frequency.

  Another important focus is on tweet NLP in order to recognize named entities. We experimented with the Stanford NLP [8] and the nltk pos-tagger [3, 4], but found that they many times failed to recognize entities due to the specific language of tweets, e.g., arbitrary capitalization of words (e.g., AWESOME vs obama, many NER taggers rely on capitalization for clues on potential entities [12]), short names (e.g., fb for Facebook). For this reason, we turned to the CMU Twitter NLP and Part-of-Speech Tagging tool[4] for recognizing entities [6]. In particular we used a python wrapper around the CMU Java code [9]. This tool is trained on tweets and had better accuracy for named entity recognition in our tests. We apply this tool to each of the terms in our vocabulary, in order to recognize entities.

[4] `http://www.ark.cs.cmu.edu/TweetNLP/`

Once we compute the $df - idf_t$ and identify the entities in the vocabulary of each time window, we assign each term a weight computed as $df - idf_t * entity\_boost$, where the entity boost was set to 2.5 in our case versus the 1.5 used in [1]. We found that a higher entity weight leads to retrieving more news-like topics. Once the term weight is computed this way, each cluster gets assigned the score of the term with highest weight (as in [1]), but we normalize this by the cluster size. This last normalization step seems to lead to less topic fragmentation, allowing smaller clusters with prominent terms, to rank higher. We have also experimented with cluster scores that average the score of the terms of a cluster. Interesting enough, when using unigrams rather than bi-grams and tri-grams for the vocabulary, ranking clusters by averaging term scores worked better than the maximum term score. We investigate these differences in cluster scoring in our experiments. We rank the clusters using this score, and retain only top-20 clusters, subject to a size constraint, e.g., for a cluster to be considered a topic it should have at least 10 tweets.

We have also attempted to assign a boost to terms based on their occurrence in news articles that are streamed in a similar time window as the tweets. Nevertheless, this approach may work for some type of events, such as politics related, where the news travel from the news outlets onto Twitter, but may not work for events that first break on Twitter, such as sports events, that are later reported and summarized by the news outlets. For future work we intend to analyze the connection between news articles and tweets streamed in the same time frame, and for certain type of events.

Furthermore, we have attempted to use deeper NLP in the first stages of our development (e.g., pos-tagging and extracting nouns and verbs), but minimal stop words removal and tweet cleaning/filtering proved to be much more efficient and equally accurate regarding topic detection. We also found, as in [1], that stemming hurts the quality of topics retrieved, so we did not apply stemming to our terms.

- **Selecting topic headlines.** We select the first (with respect to publication time) tweet in each cluster of the top-20, as our headline for the detected topic. This clustering/ranking strategy covers several events but many times suffers from topic fragmentation, e.g., we may get several headlines about the same topic. This issue has also been found previously in [1]. Next we discuss strategies for dealing with topic fragmentation and reducing the set of topics to only top-10.

- **Re-clustering headlines to avoid topic fragmentation.** Our final step involves clustering of only the headlines selected after the first stage of clustering and ranking. These are cleaned tweets used for clustering in the first stage (no user mentions, urls, filtered vocabulary). We build a headline-by-term matrix, using unigrams for our vocabulary, without any other restriction on terms. We re-cluster the headlines using hierarchical clustering, and cut the dendrogram at the maximum distance (e.g., 1.0 for cosine). Again setting this threshold decides how many headlines we want to collate into a single topic. We rank the resulting headline-clusters using the headline with the highest score inside a cluster, therefore if the headlines do not cluster at all, the ranking of headlines will stay the same as in the previous step.

- **Final selection of topics** From this final clustering and ranking step, we select the headline with the earliest publication time, and present its raw tweet (without urls) as a final topic headline. We pool the keywords of the headlines in the same headline-cluster to extract topic-tags (a list of keywords as a description of the topic). For selecting tweet ids relevant to the extracted topic, we use the ids of the clustered headlines (i.e., the id of the tweet corresponding to the headline), and otherwise a single id, if the headline-cluster contains a single headline. The idea behind this strategy is that if the first stage of clustering did not split a topic, the tweets inside the topic-cluster were very similar to each other. For extracting urls of photos relevant to the topic, we first check if the headlines have any *media_url* tags (as extracted from the JSON object), and if not, we loop through the cluster (from stage 1) of tweets to which the headline belongs, in search of a media url in those tweets. Restricting the number of media urls to 1 or 2 directly affects the speed of the overall topic extraction process, since we don't have to dive too deep into the previous (potentially large) clusters.

5. EVALUATION

To evaluate our approach, we use the subset of ground truth topics provided by the challenge organizers for the 2012 US elections stream. For the second 2014 stream, where we were not provided with ground truth topics, we google for the automatically detected topic headline and manually asses how many of our headlines are published news in traditional media from the same time period (25-26 February 2014). We discuss our results for different choices of parameters, vocabulary and cluster scoring functions. The official evaluation results of our method in the Data Challenge are included in [16].

5.1 Results

Parameter Analysis. In this section we investigate the effect of various parameters on the resulting set of topics. For setting parameters we use the subset of ground truth topics provided by the challenge organizers for the 2012 stream, a sample of which is shown in Table 1. For comparison, in Table 2, we show the top10 topics detected by our method (with parameters set as described in the previous section) for the same stream, for the time slot starting at 07-11-2012 00:00. In Table 3, we show the top10 topics produced by our method for the 2014 stream (parameters same as for Table 2), for the time window starting at 25-02-2014 18:00.

Tweet Length and Structure. We relax the requirement that a tweet should be of length at least 5 in the final tweet-term matrix, to length at least 3. This leads from the set of total tweets in window[5] of 22,847, and an initial tweet-term matrix with 12,684 tweets and 588 terms, and filtered tweet-term matrix with 3,258 tweets, 588 terms to a tweet-term

[5]All numbers are for the time window of Table 2.

Table 1: Example ground truth topics for the 2012 US elections Twitter stream.

Time	Topic Headline	Topic Keywords	Tweets Ids
07-11-12 00:00	Obama wins Vermont	Obama,Vermont,wins,projects,VT	265966881926688768,265966897793740800
07-11-12 00:00	Romney wins Kentucky	Romney,wins,Kentucky,projects,KY	265966833524424704,265966921537695744
07-11-12 00:00	Bernie Sanders wins Senate seat in Vermont	Sanders,wins,Senate,Vermont, independent,VT	265967450074513408,265967599123316736
07-11-12 00:00	Romney wins Indiana	Romney,wins,Indiana,IN	265966811449810945,265966944522481665
07-11-12 00:30	Romney wins West Virginia	Romney,wins,West Virginia,WV	265974256159039488,265974324148723712
07-11-12 00:30	Romney wins South Carolina	Romney,wins,South Carolina,SC	265975742649729024,265975879736373248
07-11-12 01:00	Obama wins Illinois	Obama,wins,Illinois,IL	265982157355376640,265982400880861184
07-11-12 01:00	Obama wins Connecticut	Obama,wins,Connecticut,CT	265982401157689345,265982401795215360
07-11-12 01:00	Obama wins Maine	Obama,wins,Maine,ME	265982400880861184,265982412897529857

Table 2: Detected top10 topics using our method for the 2012 US elections Twitter stream.

Time	Topic Headline	Topic Keywords	Tweets Ids
07-11-2012 00:00	WASHINGTON (AP) - Obama wins Vermont; Romney wins Kentucky. #Election2012	#election2012, @ap, ap, begins, breaking, calls, carolina, close, cnn, fox, georgia, indiana, kentucky, news, obama, presidential, projects, race, romney, south, vermont, washington, wins	265967355648167937, 265967692161363969, 265967306985844736, 265967261297295361, 265967261297295361, 265967255815340032
07-11-2012 00:00	Not a shocker NBC reporting #Romney wins Indiana & Kentucky #Obama wins Vermont	#obama, #romney, indiana, kentucky, nbc, reporting, vermont, wins	265967338992570368
07-11-2012 00:00	RT @SkyNewsBreak: Sky News projection: Romney wins Kentucky. #election2012	#election2012, @skynewsbreak, indiana, kentucky, news, obama,	265967389974343680, 265967700734533633
07-11-2012 00:00	AP RACE CALL: Democrat Peter Shumlin wins governor race in Vermont. #Election2012	#election2012, ap, bernie, call, democrat, governor, peter, race, sanders, seat, senate, shumlin, vermont, wins	265968208291438592, 265967599123316736
07-11-2012 00:00	CNN Virginia exit poll: Obama 49%, Romney 49% #election2012	#election2012, cnn, exit, obama, poll, romney, virginia	265967764815110146
07-11-2012 00:00	Mitt Romney Losing in Massachusetts a state that he governed. Why vote for him when his own people don't want him?	#Obama2012 #obama2012, governed, losing, massachusetts, mitt, people, romney, state, vote, want	265966841686544385
07-11-2012 00:00	Twitter is gonna be live and popping when Obama wins! #Obama2012	#obama2012, gonna, live, obama, popping, twitter, wins	265968524072218624
07-11-2012 00:00	INDIANA RESULTS: Romney projected winner (via @NBC) #election2012	#dumbasses, #election2012, @huffingtonpost, @nbc, indiana, projected, results, romney, winner	265968527289249792, 265968527289249792
07-11-2012 00:00	If Obama wins I'm going to celebrate... If Romney wins I'm going to watch Sesame Street one last time #Obama2012	#obama2012, celebrate, going, last, obama, one, romney, sesame, street, time, watch, wins	265966816730435584
07-11-2012 00:00	#election2012 important that Romney won INdependents in Virginia by 11 pts. With parties about even, winning Inds is key	#election2012, even, important, independents, inds, key, parties, pts, romney, virginia, winning, won	265968665915191296

Table 3: Detected top10 topics using our method for the 2014 Syria, Ukraine, Bitcoin Twitter stream.

Time	Topic Headline	Topic Keywords	Tweets Ids	Published News?
25-02-2014 18:00	The new, full Godzilla trailer has roared online:	awesome, brand, full, godzilla, landed, new, online, roared, trailer	438373491440500737, 438373702573379584	YES
25-02-2014 18:00	At half-time Borussia Dortmund lead Zenit St Petersburg 2-0. #bbcfootball #ChampionsLeague	#bbcfootball, #championsleague, @bbcsport, borussia, dortmund, half, lead, petersburg, st, time, zenit	438372831081279488	YES
25-02-2014 18:00	Ukraine Currency Hits Record Low Amid Uncertainty: Ukrainian currency, the hryvnia, hits all-time low against ...	amid, currency, hits, hryvnia, low, record, time, ukraine, ukrainian, uncertainty	438373672412143616	YES
25-02-2014 18:00	Ooh, my back! Why workers' aches pains are hurting the UK economy	aches, back, economy, hurting, pains, uk, workers	438372908814303232	YES
25-02-2014 18:00	Uganda: how campaigners are preparing to counter the anti-gay bill	anti, bill, campaigners, counter, gay, preparing, uganda	438373369491505152	YES
25-02-2014 18:00	JPost photographer snaps what must be the most inadvertantly hilarious political picture of the decade	@jerometaylor, decade, hilarious, inadvertantly, jpost, photographer, picture, political, snaps	438372882088226816	YES
25-02-2014 18:00	Fans gather outside Ghostbusters firehouse in N.Y.C. to pay tribute to Harold Ramis	fans, firehouse, gather, ghostbusters, harold, nyc, outside, pay, ramis, tribute	438375154008461313	YES
25-02-2014 18:00	Man survives a shooting because the Bible in his top pocket stopped two bullets	@metrouk, bible, bullets, man, pocket, shooting, stopped, survives, top, two	438373191762059265	YES
25-02-2014 18:00	#Ukraine's toppling craze reaches even legendary Russian commander, who fought Napoleon	#ukraine, commander, craze, even, fought, legendary, napoleon, reaches, russian, toppling	438374254002700288, 438374829339987968	YES
25-02-2014 18:00	Newcastle City Hall. Impressive booking first from bottom on the left...	@robbrydon, booking, bottom, city, first, hall, impressive, left, newcastle	438372863377408000	NO

matrix with 3,777 tweets, and 588 terms. Therefore, we get 500 extra tweets when relaxing the tweet-length constraint. The effect on topics is nevertheless very low, we can thus keep an aggressive length filter without strongly affecting the final set of detected topics.

Unigrams vs Bi-grams/Tri-grams. We change the vocabulary to unigrams, rather than bi-grams and tri-grams, and keep all the other params fixed. This leads to 9,028 tweets and 482 terms (as compared to 3,258 tweets by 588 terms). This triples the number of tweets that qualify for passing the filter conditions, thus making the topic detection process less efficient. The topics detected with unigrams are fairly similar to those detected using bi-grams and tri-grams, but the use of n-grams ($n > 1$) allows for more efficient processing.

Cluster Scoring. We investigate the effect of averaging term scores for computing a cluster score versus assigning the score of the maximum score term in the cluster. We have found that term score averaging for computing a cluster score works better with unigrams, while assigning the maximum term score works better with n-grams.

Topic Precision. For the first stream with provided ground truth, we found that we can retrieve all the provided topics.

In order to asses the quality of our detected topics for the second stream, where we lack ground truth, we googled for the first 100 detected topics (top10 of the first 10 time windows, of 15 minutes each), and evaluated how many were actually published as news on sources other than Twitter. We found that about 80% of our topics are published as news, by news media outlets (see also Table 3).

5.2 Efficiency

The tweet clustering method presented above runs[6] in around 1h for the full 24h data stream (96 time windows of 15 mins each). The most time consuming parts are the tweet pairwise distance computation and the hierarchical clustering, but we observed that aggressive filtering of both tweets (based on structure/length) and terms (bi-grams and tri-grams) with strict thresholds on document frequency (minimum 10 tweets), can address the efficiency aspect.

6. CONCLUSION

We present a method for topic detection in Twitter streams, based on aggressive tweet/term filtering and two stage hierarchical clustering, first of tweets and second of resulting headlines from the first clustering step. The topics obtained seem encouraging, many of them being published as news in the traditional news media. Our topic-headlines are actual tweets, so the user can trace the news back to its original tweet, and are presented in the context of photos (from tweet media urls) and tags selected from those tweets.

One of the potential weaknesses of our method consists in the aspect of topic fragmentation, where topics get repeated across several clusters. This is most pronounced when news break and the same story is discussed from different points of view. We intend to investigate this further. Additionally, some headlines may get collated into a single topic: for the US 2012 elections stream, Peter Shumlin and Bernie Sanders both running for governor and Senate seats in Vermont respectively, got collated into the same topic (see Table 2, headline about Peter Shumlin and topic keywords about both candidates), therefore the issue of how to choose the headline remains (e.g., we could show the collated cluster-headline).

A big advantage of our method is its simplicity and efficiency, since it runs in less than an hour for a full 24 hour, 4GByte Twitter stream, therefore coming closer to real-time processing requirements. Strong filtering of tweets and terms seems to lead to efficient and clean results, overcoming the heavy noise aspect of Twitter content.

For the future, we intend to compare our method to BN-grams [1] and study the use of news articles and topic-focused streams to obtain a topic zoom-in effect (e.g., topic detection on focused streams separately: Ukraine vs Syria, and combining the topics in the end).

7. ACKNOWLEDGMENTS

This work was supported by Science Foundation Ireland under grant 07/CE/I1147 and SFI/12/RC/2289.

8. REFERENCES

[1] L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Goker, I. Kompatsiaris, and A. Jaimes. Sensing trending topics in twitter. *IEEE Transactions on Multimedia*, 2013.

[2] J. Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer, 2002.

[3] S. Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, 2006.

[4] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[7] J. Guzman and B. Poblete. On-line relevant anomaly detection in the twitter stream: an efficient bursty keyword detection model. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 31–39. ACM, 2013.

[8] http://nlp.stanford.edu/software.

[9] https://github.com/ianozsvald/ark-tweet nlp.

[10] http://www.telegraph.co.uk/technology/twitter/9945505/Twitter-in numbers.html.

[11] M. Jordaan. Poke me, i'm a journalist: The impact of facebook and twitter on newsroom routines and cultures at two south african weeklies. *Ecquid Novi: African Journalism Studies*, 34(1):21–35, 2013.

[12] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee. Twiner: named entity recognition in targeted twitter stream. In *ACM SIGIR*, 2012.

[13] C. Martin, D. Corney, and A. Göker. Finding newsworthy topics on twitter. *IEEE Computer Society Special Technical Community on Social Networking E-Letter*, 2013.

[14] T. Matuszka, Z. Vinceller, and S. Laki. On a keyword-lifecycle model for real-time event detection in social network data. In *IEEE International Conference on Cognitive Infocommunications*, 2013.

[15] D. Muellner. fastcluster: Fast hierarchical, agglomerative clustering routines for r and python. *Journal of Statistical Software*, 53(9):1–18, 5 2013.

[16] S. Papadopoulos, D. Corney, and L. M. Aiello. Snow 2014 data challenge: Assessing the performance of news topic detection methods in social media. In *Proceedings of the SNOW 2014 Data Challenge*, 2014.

[17] S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.

[18] S. Schifferes, N. Newman, N. Thurman, D. Corney, A. Goker, and C. Martin. Identifying and verifying news through social media: Developing a user-centred tool for professional journalists. *Digital Journalism*, 2014.

[6] On a PC with OS X 10.9.2, 8GByte memory and 2.7GHz Intel CPU.