Automatically Identifying Periodic Social Events from Twitter

Florian Kunneman
Centre for Language Studies
Radboud University
f.kunneman@let.ru.nl

Antal van den Bosch
Centre for Language Studies
Radboud University
a.vandenbosch@let.ru.nl

Abstract

Many events referred to on Twitter are of a periodic nature, characterized by roughly constant time intervals in between occurrences. Examples are annual music festivals, weekly television programs, and the full moon cycle. We propose a system that can automatically identify periodic events from Twitter in an unsupervised and open-domain fashion. We first extract events from the Twitter stream by associating terms that have a high probability of denoting an event to the exact date of the event. We compare a timeline-based and a calendar-based approach to detecting periodic patterns from the event dates that are connected to these terms. After applying event extraction on over four years of Dutch tweets and scanning the resulting events for periodic patterns, the calendar-based approach yields a precision of 0.76 on the 500 top-ranked periodic events, while the timeline-based approach scores 0.63.

1 Introduction

As a popular communication channel for both sharing news, experiences, and intentions, Twitter has been found to provide an accurate reflection of many aspects of the real world (Bollen et al., 2011; Zhao et al., 2011). For example, the periodicity of daily life can be exposed by visualizing the frequency of hashtags such as '#breakfast' and '#goodmorning' (Preoţiuc-Pietro and Cohn, 2013). In addition, real-world events can be automatically detected by signaling a sudden rise and fall of word occurrences in tweets (Petrović et al., 2010; McMinn et al., 2013). We propose a system that can identify *periodic* events from Twitter: provided with a continuous stream of raw tweets, it returns an overview of periodic social events.

Surprisingly, this topic of periodicity has not yet been studied in the context of events mentioned on Twitter, while the identification of periodicity in recurring events has obvious gains for a system that detects events in the Twitter stream. Detected periodicity patters can be used to predict future events before they are referred to on Twitter. For instance, if World Food Day is detected on the 16th of October for a number of consecutive years, it can be expected and put on the calendar for the next year.

The rich set of references to the real world made on Twitter make it a suitable platform to mine for periodic patterns in relation to events of any type. At the same time, the non-standard language and large amount of streaming messages make it a challenging task. We facilitate this task by applying an event extraction approach that identifies terms that might represent a social event, and that relates them to a frequently and explicitly mentioned date of the event. After this first event extraction stage, periodicity detection can be applied to the clean date sequences linked to event terms.

2 Related Work

Finding periodic patterns is a valuable task in many contexts of sequential data, such as DNA or protein sequences (Zhang et al., 2007), market basket data (Mahanta et al., 2008), and complex signals such as sound (Sethares and Staley, 1999). Elfeky et al. (2005) distinguish between 'segment periodicity' and 'symbol periodicity'. The first refers to the repetition of a specific sequence, while the second refers to single symbols in a sequence that recur at roughly constant time intervals. The latter is what we aim to detect.

Several patterns of periodicity have been analyzed in social media. Chu et al. (2012a) aim to distinguish bots from human user accounts on Twitter, and find that the periodicity of tweet postings is a strong indicator to recognize bots. They

Proceedings of Recent Advances in Natural Language Processing, pages 320–328,
Hissar, Bulgaria, Sep 7–9 2015.

estimate periodicity by the entropy rate of post intervals, where a low entropy points to a non-random, periodic pattern. Chu et al. (2012b) adopt this entropy-based periodicity feature to help distinguish spam campaigns from proper campaigns on Twitter. Fan et al. (2014) analyze temporal patterns in topics discussed on Weibo, and find that the topic 'business' displays a highly periodic pattern. Yang et al. (2013) aim to classify Twitter users in predefined categories. They find that the periodicity pattern of words linked to a category is a strong indication, as users tend to mention their topic of interest at similar times of the day and week. At the word level, Preoţiuc-Pietro et al. (2013) apply Gaussian processes to model the periodicity of hashtag mentions. They use this information to predict hashtag frequencies at any hour.

The automatic identification of periodic patterns related to events has not been applied in the context of Twitter. The detection of single events, on the other hand, is a popular strand of research. Many studies have leveraged the notion of burstiness, the sudden rise and fall of word frequency, to find events from Twitter. Either by looking at the rapid growth of tweet clusters (Petrović et al., 2010; McMinn et al., 2013; Diao et al., 2012) or words with peaky behavior (Weng and Lee, 2011; Li et al., 2012). The explicit reference to events in tweets has also been shown to help find scheduled events; social events in particular (Ritter et al., 2012). A possible reason the aforementioned approaches have not been employed to search for periodically recurring events, is that it requires a longitudinal effort to increase the chances of observing periodic behavior. To this end, we make use of TwiNL (Tjong Kim Sang and van den Bosch, 2013), a database of IDs of Dutch tweets gathered from December 2010 onwards.

3 Approach

3.1 Open-domain Event Extraction

Our approach to event extraction is similar to Ritter et al. (2012). The approach relies on explicit references to a future point in time combined with terms, and favors date–term pairs with a strong connection. We apply this approach to Dutch tweets, though most of its components are language-independent.

3.1.1 Tweet Processing

Each incoming tweet from a stream is initially scanned for future referring time expressions. We manually specified a set of rules that focus on a future date in time as expressed in the Dutch language. Examples of the English equivalents of these rules are displayed in Table 1.[1]

Category	Examples (English)
Date	Sept. 13th 2014
Exact	in a month
Weekday	this Wednesday

Table 1: Examples of the three types of rules for the extraction of time expressions

The set of rules can be divided in three categories that each relate to different types of conversion of the time expression into a date. The 'Date' category of rules consists of the different variations of date mentions, and link directly to a future date. The 'Exact' rules comprise a variety of phrase combinations that specify an exact number of days remaining to the event. The 'Weekday' rules match a mention of a weekday, preceded by a future referring phrase like 'deze' ('this') or 'volgende week' ('next week').

Tweets found to have a future referring time expression are subsequently scanned for meaningful words and word n-grams that might denote an event. We refer to such n-grams as 'event terms' henceforth. As off-the-shelf named entity taggers display a poor performance when applied on the non-standard language in social media (Ritter et al., 2011), as alternative we applied the commonness metric (Meij et al., 2012) and extracted any hashtag as event term.

Commonness is formulated as the prior probability of a concept c (the n-gram) to be used as an anchor text q in Wikipedia (Meij et al., 2012):

$$Commonness(c, q) = \frac{|L_{q,c}|}{\sum_{c'} |L_{q,c'}|} \quad (1)$$

Where $L_{q,c}$ denotes the set of all links with anchor text q pointing to the Wikipedia page titled c, and $\sum_{c'} |L_{q,c'}|$ is the sum of occurrences of q as an anchor text linking to other concepts.

[1] Although commonly available time taggers such as Heideltime (Strötgen and Gertz, 2010) could be applied to this end, Heideltime does not specialize in future time expressions.

We downloaded the Dutch Wikipedia dump of 2015/02/05[2], and parsed it with the Annotated-WikiExtractor[3]. Then, we used Colibri Core[4] to calculate the commonness of any concept that has its own Wikipedia article, and is used as an anchor text on other Wikipedia pages at least once. These statistics are used to extract event terms from a tweet. Tweets that match a future time reference in the first stage are stripped of this time reference, and n-grams with $n \leq 5$ (covering the length of most event names) are extracted. Any n-gram found to have a commonness score above 0.05 is extracted as an event term. We set the threshold based on analyzing a sample n-grams with their score.

Aside from concepts with an above-threshold commonness score, we directly selected any hashtag in tweets with future time references as event terms. Hashtags can be seen as user-designated keywords, and are often employed as event markers.

3.1.2 Event Ranking

Not all pairs of dates and event terms that result from the tweet processing stage represent a significant event. Some candidate terms might not refer to an event, and some terms might denote a personal rather than a social event. A first step to identify significant events is to employ a minimum threshold of five tweets in which an event term should co-occur with the same date. Following Ritter et al. (2012), event terms more frequently mentioned with a specific date are seen as the more significant events. We therefore calculate the fit between any frequent event term and the date with which it is mentioned, by means of the G_2 log likelihood ratio statistic:

$$G_2 = \sum_{z \in \{e, \neg e\}, y \in \{d, \neg d\}} O_{z,y} \times \ln\left(\frac{O_{z,y}}{E_{z,y}}\right) \quad (2)$$

The fit between any event term e and date d is calculated by the observed (O) and expected (E) frequency of the four pairs $\{e, d\}$,$\{e, \neg d\}$,$\{\neg e, d\}$ and $\{\neg e, \neg d\}$. The expected frequency is calculated by multiplying the observed frequencies of z (denoting either e or $\neg e$) and y (denoting either d or $\neg d$) and dividing them by the total number of tweets in the set.

We prioritize events that are tweeted about by many different users, by multiplying the G_2 log likelihood ratio statistic with the fraction of different users that mention the event. The events are ranked by the resulting G_2u score:

$$G_2u = \left(\frac{u}{t}\right) * G_2 \quad (3)$$

Here, u is the number of unique users that mention the date and entity in the same tweet, while t is the number of tweets in which the date and entity are both mentioned.

The calculation of G_2u for each pair results in a ranked list of date–term pairs. To reduce subsequent computational costs, all pairs with a rank number below 2,500 are discarded.

As an event might be described by multiple event terms, it is likely that the ranked list of date–term pairs contains several event terms that describe the same event. To de-duplicate these, we cluster event terms based on the content of the tweets in which they are mentioned. Each set of tweets in which the same date–term pair occurs is aggregated into one big document. The documents are converted into a feature vector with $tf * idf$ weighting, where the idf value is based on all aggregated documents in the set of 2,500 date–term pairs. A similarity matrix is generated for each set of date–term pairs labeled with the same date. These documents are then clustered by means of Agglomerative Hierarchical Clustering (Day and Edelsbrunner, 1984). The advantage of this algorithm is that it does not require a fixed number of clusters as parameters. Pairs of tweet sets are clustered together if their similarity is above an empirically set threshold of 0.7.

3.1.3 Performance

We tested the approach to event extraction on a large sample of Dutch tweets posted in August 2014, which we collected from TwiNL (Tjong Kim Sang and van den Bosch, 2013). We evaluated the top-250 extracted events, and compared the outcomes with an n-gram baseline, in which the commonness approach to entity extraction is replaced by extracting all n-grams. The results are displayed in Table 2.

The output of the system was rated by a set of four annotators; the results are presented by the

[2] http://dumps.wikimedia.org/nlwiki/nlwiki-20150205-pages-articles.xml.bz2

[3] https://github.com/jodaiber/Annotated-WikiExtractor

[4] http://proycon.github.io/colibri-core/doc/

	50%	75%	100%	Mutual F-score
Ngram	0.52	-	0.42	0.89
Commonness	0.87	0.80	0.63	0.90

Table 2: Precision@250 of output identified as event by human annotators

minimal percentage of annotators that agreed on the event status. A majority of 75% of the annotators agrees that 80% of the output represents an event. In comparison, only 52% of the top-250 output of the n-gram baseline system was rated as event by at least one of two annotators. We also scored the inter-annotator agreement by Mutual F-score, which provides an insight into the agreement for the positive (event) class. On average, annotators yield an F-score of 0.9 on classifying the event class if the decisions of another annotator are seen as the gold standard.

3.2 Online Extraction of Events

The approach described above extracts events from a *fixed* set of tweets. To apply the event extraction in a *streaming* fashion, the procedure should be repeated for any new batch of tweets. We chose to work with a window size of one month. We set the step size to one day, to ensure that events are extracted from any monthly periodic sequence. For each daily event extraction step, the top-2500 events are selected.

This overlapping sliding window setting leads to a large amount of duplicate events in the output. To build a calendar of unique events, a merging procedure is performed after each event extraction. The events in the output are each compared with the existing set of events with the same date. If over 10% of the tweets in the new event overlap with an existing event, the events are merged by adding any new tweets and event terms to the existing event. New events that do not overlap with an existing event are added as a new event.

3.3 Periodicity Detection

The event extraction procedure results in a set of events represented as one or more event terms linked to a date. Next, periodic events can be found by scanning for events that are linked to at least three dates, between which two periods of time occur that are roughly equal.

We compare two approaches to finding periodic patterns from the date sequence related to an event term: a timeline-based approach and a calendar-based approach. We refer to them as 'PerTime' and 'PerCal'.

3.3.1 PerTime

PerTime leverages the intervals between sequences of at least three dates. Any date sequence that has roughly similar intervals is seen as periodic. The intervals are measured at the level of days. We estimate the similarity by computing the relative standard deviation over the intervals, RSD:

$$RSD = \frac{s}{\bar{x}} * 100\% \quad (4)$$

The RSD relates the average $\bar{x}$ to the standard deviation s, returning the standard deviation as the percentage of the average values in a set. The RSD is a sensible approach to scoring the periodicity of date intervals, as any deviation in big intervals, such as 365 days, is less penalized than the deviation in smaller intervals, such as 7 days. We set the minimum interval length to 6 days, ensuring weekly events as the minimal periodicity.

3.3.2 PerCal

Rather than looking for regular intervals between dates, PerCal searches for similarities between the dates in a sequence. An event term like 'Christmas Day' would be mostly linked to '25 December'. Likewise, an event term might recur with 'the third Saturday of May'. The calendar-based approach scans a date sequence for such repetitions.

The detection of calendar periodicity has mainly been the focus in studies that aim to find periodic transactional patterns (Li et al., 2001; Li et al., 2003; Mahanta et al., 2008). Li et al. (2001) propose an intuitive calendar scheme to describe a periodic pattern. The pattern has the form of ⟨year,month,day⟩. Any of these fields can be filled with a specific value, while the '*'-character is used to denote 'every'. For example, the pattern ⟨*,2,1⟩ represents 'every year on the 1st of February', while ⟨2011,*,12⟩ denotes 'every twelfth day of the month in 2011'. We adopt this pattern scheme, and extend it with the additional fields week, weekday, and #weekday (the index of a given weekday within a month). We add the '-' character as a possible value, to account for fields that are irrelevant to a pattern. As an additional extension, we allow the model to describe patterns

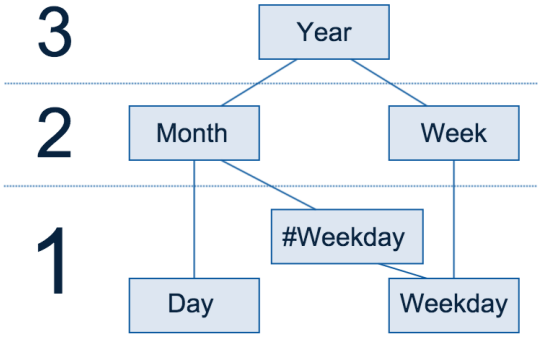

Figure 1: Diagram of included calendar fields and their relation on three levels.

like 'every six months' or 'every two years', by specifying a step size that relates to the field that is described by 'every'. For example, ⟨*2,1,-,-,Sunday,2⟩ denotes 'every two years on the second Sunday of January', and ⟨2011,*,-,1,-,-⟩ denotes 'every first day of the month in 2011'.

The relationship between the included calendar fields is illustrated in Figure 1. The scheme has three levels of granularity. On the first level are 'day' (1–31), 'weekday' (Monday–Sunday) and '#weekday' (1–5). The 'day' field relates to 'month' (1–12) at the second level; any combination between the two values can be made. '#weekday' has a connection to both 'weekday' and 'month', and represents the index of a weekday in a month (for example: the *third* Wednesday of October). Finally, 'weekday' connects directly to 'week' (1–53), which enables relations like 'every Wednesday' or 'Monday on week 40'. At the top level is the 'year' field, so as to describe yearly patterns or patterns during a specific year.

A periodic calendar pattern can be detected by ascending the hierarchy of calendar fields and looking for regularities. Like PerTime, weekly periodicity is the smallest pattern that is searched for. Starting from the lower-level fields (day, weekday and the weekday-#weekday combination), the algorithm scans whether any of the values of these fields occurs three times or more (the minimum requirement for a periodic pattern). If this requirement is met, the dates that contain this value are selected and passed on to the higher level: month (if the day or the weekday-#weekday combination is periodic) or week (if the weekday is periodic). Because the patterns we look for can describe either a sequence on this second level (like 'every two months' or 'every week') or a sequence of years on the third level, we scan both for a sequence and a repetition of the month or the week values on this second level. If a sequence is found, the pattern is finalized. If a repetition is found, the algorithm proceeds to find a yearly pattern.

A sequence of weeks, months or years might have steps of unequal size. In such a case we describe the pattern with the smallest step size found. Any date between larger steps is denoted as a missing date. In the sequence '2014/03/04 – 2014/04/04 – 2014/06/04' there is a monthly sequence of step size '1', with a missing date '2014/05/04'.

Some patterns show stronger periodicity than others. As mentioned above, a sequence might contain missing dates, decreasing the evidence for periodicity. In addition, not all dates linked to an event term may combine into a pattern. Following Li et al. (2001), we quantify these two inconsistencies as *confidence* and *support* estimates. Confidence is estimated by dividing the dates that could fill in a pattern (from the first date to the last) by the number of dates that are actually seen. Support is the percentage of all dates that are linked to an event term that satisfy the pattern. To obtain an overall score of the quality of a pattern, we calculate the average of these two metrics.

PerCal searches for periodic patterns at different levels. As a result, it may find multiple patterns in the same date sequence. If two patterns overlap, the one with the highest overall score is selected.

3.3.3 Clustering of Periodic Terms

To de-duplicate output from both the PerTime and PerCal approaches, we cluster event terms with a periodic sequence together. For both approaches, we aggregate all tweets linked to the periodic pattern of an event term, to form big documents. Any pair of terms with 90% overlapping dates for PerTime and any pair with a similar pattern for PerCal were tested as clusters. Clustering was applied in the same fashion as described at the end of Section 3.1.2. The threshold for clustering was set to a cosine similarity above 0.5.

4 Experimental Set-up

4.1 Data

We tested our system on all Dutch tweets that were collected from the Tweet IDs in TwiNL, from the start of the database, December 16th 2010, up to February 16th 2015, amounting to 2.73 bil-

lion tweets in total. After processing these tweets, 24,162,633 were found to have a matching time expression.

4.2 Procedure

We applied the event extraction module on the span of tweets as specified in Section 3.2, with a sliding window of a month and a daily sliding frequency. Events were merged if they were extracted from (partly) the same tweet IDs. After all tweets were processed, a calendar was filled with 94,526 events.

Periodicity detection is applied to single event terms; we kept a log of the dates linked to each term. We searched for periodic patterns in this log by starting with events that took place in 2014. For both PerTime and Percal, whenever a date in 2014 or later was appended to an event term log, the approach was applied to the updated date sequence. If a periodic pattern was already found for an event term, it was overwritten with the pattern that was extracted from the updated sequence. We clustered terms with a similar periodic pattern after all events were processed.

4.3 Evaluation

We ranked the periodic event patterns returned by the two approaches by their respective metrics to score periodicity: RSD for PerTime and the average value of support and coverage for PerCal. One of the authors manually assessed the top-500 patterns from both rankings, deciding for each output whether it represents a regularly recurring sequence of events, rather than events or event terms that share a coincidental temporal regularity. The terms, dates, and tweets linked to each output, and if needed the Google search engine, were consulted to guide this decision.

In order to acquire a sense of agreement for the annotations, a second author annotated the top-200 events of the two systems. The mutual F-score of positive annotations was 0.92 for the PerTime output and 0.93 for the PerCal output.

5 Results

PerTime assigned a periodicity score to 5,301 events out of the total of 94,526 events. PerCal found 7,018 periodic patterns [5]. The precision and recall of their top-500 output are presented in Table 3. 315 correct periodic events were confirmed from the output of PerTime, and 379 from the output of PerCal, resulting in precision-at-500 scores of 0.63 and 0.76, respectively. We approximated a recall score by comparing the periodic event terms that were found by both approaches (637 in total), and calculating which percentage of these was returned by either of them. The recall scores are lower than the precision scores, due to an overlap of only 116 events (18%) between PerTime and PerCal.

	Precision	Recall
PerTime	0.63	0.52
PerCal	0.76	0.69

Table 3: Periodicity detection quality after manual evaluation of the top-500 deteced periodic events by the two approaches.

Precision-at-curves of the top-500 rankings are given in Figure 2. For PerTime, the RSD at rank 500 is 10.2 days. A perfect RSD score of 0.0 was maintained up to rank 81. The ranks of events with equal scores were randomly shuffled. The curve shows a progressing decay towards the end. The temporally increasing precision at rank 200 is due to the detection of a number of periodic events that are characterized by changing intervals, such as Easter and Pentecost, and share the same non-perfect RSD score.

For PerCal, the pattern score at rank 500 is 0.65. In contrast to PerTime, precision is decreasing at a slower rate with lower-ranked events.

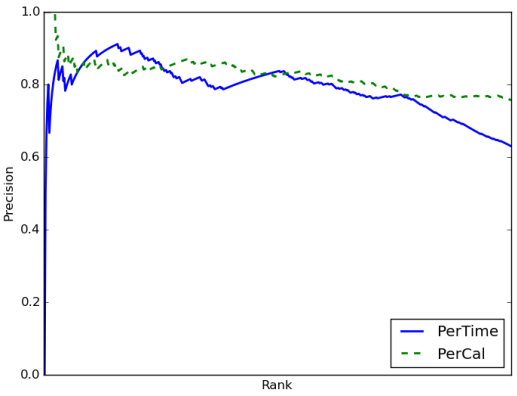

Figure 2: Precision-at-curves for PerTime and PerCal

[5] A dataset with the tweet ID's that relate to all 94,526 events, as well as the periodic event patterns that were found by both systems, will be made publicly available from http://cls.ru.nl/~fkunneman/data_periodicity.zip

	Event term(S)	Dates	Timeline pattern	Calendar pattern
Periodic events found by both approaches	#trendrede	2011/09/13,2012/09/11, 2013/09/10, 2014/09/09	364 -364 - 364	⟨*,9,-,-,Tuesday,2⟩
	#valentinesday	2013/02/14, 2014/02/14, 2015/02/14	365 - 365	⟨*,2,-,14,-,-⟩
Periodic events only found by timeline approach	romantische muziek	2011/08/14, 2012/08/12, 2013/08/25, 2014/08/24	364 - 378 - 364	-
	paaszondag	2011/04/24, 2012/04/08, 2013/03/31, 2014/04/20, 2015/04/05	350 - 357 - 385 - 350	-
Periodic events only found by calendar approach	#7hloop	2011/11/20, 2012/11/18, 2014/11/16	364 - 728	⟨*,11,-,-,Sunday,3⟩
	fortarock	2011/07/02, 2012/06/02, 2012/11/09, 2013/06/01, 2014/05/31, 2015-06-06	336 - 160 - 204 - 364 - 371	⟨*,-,22,-,Saturday,-⟩

Table 4: Examples of periodic events in the top 500 output of the timeline and calendar approach

6 Analysis

Examples of detected periodic events are given in Table 4. To give an idea of the strength of both approaches, a distinction is made between events that are only found by one of them, or by both. An example of a periodic event found by both approaches is '#valentinesday'. Events like this, linked to a fixed date, are characterized by equal yearly intervals (only allowing for a minor deviation of 366 instead of 365 days in leap years).

The event 'romantische muziek' (referring to the 'Day of Romantic Music') is not found by PerCal, which is due to an inconsistent pattern of dates. PerTime can typically deal with such small inconsistencies. The event described by 'paaszondag' ('Easter Sunday') follows the lunisolar calendar, while the calendar approach follows a Gregorian calendar scheme[6]. Again, PerTime only penalizes the inconsistencies in day intervals, without discarding the event altogether.

While PerTime can deal with inconsistencies in the intervals between dates, PerCal displays a higher tolerance towards missing dates. An example is '#7hloop' (a running event in The Netherlands), which was not found by the event extraction module in 2013. The resulting interval of 728 days (two years) at this point results in a poor periodicity score for PerTime. PerCal, having detected the overall pattern, gives a smaller penalty for the missing entry in 2013. The support for these days is 1.0, while the confidence is 0.75, leading to an overall score of 0.88. Similarly, noisy date sequences in which only part of the dates form a periodic pattern can only be dealt with by PerCal. PerTime assigns a low overall periodicity score to the date sequence associated with 'Fortarock' (a music festival in The Netherlands), due the irregular intervals.

[6]To find events like Easter, the framework of PerCal could be extended by including a lunisolar scheme or other existing schemes.

7 Conclusion

We have presented a framework that extracts a calendar of events from the Twitter stream and detects periodic event sequences in this calendar. Applying the procedure to over 4 years of Dutch tweets, a timeline-based and calendar-based approach to periodicity detection yield a precision-at-500 of 0.63 and 0.76, respectively.

As far as we know this is the first work that deals with the task of periodic event detection on Twitter data, which serves to extract long-range patterns from Twitter, detect periodic events among those patterns, and predict events before they are mentioned on Twitter. Although we obtained encouraging results, there is room for improvement. To clarify whether the event extraction approach that we applied is most suitable as a first step before periodicity detection, other approaches to event detection or extraction, such as burstiness, may be applied as well during this stage for comparison.

The calendar-based approach may be extended in a knowledge-driven way with schemes that describe the lunisolar calendar, the lunar calendar, as well as other historical and religious calendars, so as to enable the detection of periodic patterns that relate to Easter, the Ramadan, and Hindu festivals for example.

Acknowledgements

This research was funded by the Dutch national program COMMIT. We thank Erik Tjong Kim Sang for the development and support of the

`http://twiqs.nl` service.

References

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In *ICWSM*.

Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2012a. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing*, 9(6):811–824.

Zi Chu, Indra Widjaja, and Haining Wang. 2012b. Detecting social spam campaigns on twitter. In *Applied Cryptography and Network Security*, pages 455–472. Springer.

William H. E. Day and Herbert Edelsbrunner. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. 2012. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics.

Mohamed G. Elfeky, Walid G. Aref, and Achmed K. Elmagarmid. 2005. Periodicity detection in time series databases. *Knowledge and Data Engineering, IEEE Transactions on*, 17(7):875–887.

Rui Fan, Jichang Zhao, Xu Feng, and Ke Xu. 2014. Topic dynamics in weibo: Happy entertainment dominates but angry finance is more periodic. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 230–233. IEEE.

Yingjiu Li, X. Sean Wang, and Sushil Jajodia. 2001. Discovering temporal patterns in multiple granularities. In *Temporal, Spatial, and Spatio-Temporal Data Mining*, pages 5–19. Springer.

Yingjiu Li, Peng Ning, X. Sean Wang, and Sushil Jajodia. 2003. Discovering calendar-based temporal association rules. *Data & Knowledge Engineering*, 44(2):193–218.

Chenliang Li, Aixin Sun, and Anwitaman Datta. 2012. Twevent: segment-based event detection from tweets. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 155–164. ACM.

Anjana K. Mahanta, Fokrul A. Mazarbhuiya, and Hemanta K. Baruah. 2008. Finding calendar-based periodic patterns. *Pattern Recognition Letters*, 29(9):1274–1284.

Andrew J. McMinn, Yashar Moshfeghi, and Joemon M. Jose. 2013. Building a large-scale corpus for evaluating event detection on twitter. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 409–418. ACM.

Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 563–572. ACM.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181–189. Association for Computational Linguistics.

Daniel Preoţiuc-Pietro and Trevor Cohn. 2013. A temporal model of text periodicities using gaussian processes. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 977–988.

Alan Ritter, Sam Clark, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 1104–1112, New York, NY, USA. ACM.

William A. Sethares and Thomas W. Staley. 1999. Periodicity transforms. *IEEE Transactions on Signal Processing*, 47(11):2953–2964.

Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324. Association for Computational Linguistics.

Erik Tjong Kim Sang and Antal van den Bosch. 2013. Dealing with big data: The case of twitter. *Computational Linguistics in the Netherlands Journal*, 3:121–134, 12/2013.

Jianshu Weng and Bu-Sung Lee. 2011. Event detection in twitter. In *Proceedings of the AAAI conference on weblogs and social media (ICWSM-11)*, pages 401–408.

Tao Yang, Dongwon Lee, and Su Yan. 2013. Steeler nation, 12th man, and boo birds: classifying twitter user interests using time series. In *2013 IEEE/ACM International Conference on Advances in Social*

Networks Analysis and Mining (ASONAM), pages 684–691. IEEE.

Minghua Zhang, Ben Kao, David W. Cheung, and Kevin Y. Yip. 2007. Mining periodic patterns with gap requirement from sequences. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(2):7.

Siqi Zhao, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2011. Human as Real-Time sensors of social and physical events: A case study of Twitter and sports games. Technical Report TR0620-2011, Houston, TX: Rice University and Motorola Labs.