EasyChair Preprint

№ 669

MINT: A Tool to Explore Themes in High Velocity Social Media Data

Nishant Agarwal, Subhasis Dasgupta and Amarnath Gupta

December 6, 2018

MINT: A Tool to Explore Themes in High Velocity Social Media Data

Nishant Agarwal[1], Subhasis Dasgupta[1], and Amarnath Gupta[1]

University of California San Diego, La Jolla, California, USA
{niagarwa,sudasgupta, a1gupta}@ucsd.edu

Abstract. An important aspect of social media analytics is understanding themes, i.e., popular discussion topics, as they evolve in time. Techniques like topic models, which assume a generative model of theme distribution over documents, often suffer from three problems – they are expensive to compute in real-time, they usually limit the number of topics, and they do not consider the continuous change in the topic proportions, emergence and noise that occurs in social media. To address these problems, we have developed MINT, a framework that uses a non-uniform stochastic model of term occurrence to identify themes as they evolve in real-time, and periodically stores the identified themes in a semi-structured database. MINT therefore supports both theme detection and evolution queries on social media. Themes encode useful information like their rate of growth, changing user communities around them, links to related real-world events, their representative topic-handles, etc., that can be queried upon. We experimentally show that our extracted themes compare well with more traditional methods but performs well in real-time.

Keywords: Social Media Monitoring · High Velocity Data · Trend analysis.

1 Introduction

Understanding social phenomena[8] ranging from political events to impacts of natural disasters, is a fundamental scientific goal that is undertaken by different professional groups. Sociologists, political scientists, journalists, social behavioral psychologists, and legal researchers are only a few of the many different professions who observe the behavior of individuals and groups to characterize, interpret and predict social phenomena of the past, present and future. Social media channels, including social networking sites, microblogs, responses to online news and many other social outlets have become a significant source of information that today's researchers can effectively use as a huge observatory toward the shared goal of understanding the public mind and in some cases, as a predictor of impending events [12]. While social media information sources are all different from each other, they share a few common characteristics independent of the source or nature of the media they come from.

- Social data is text centric, containing dynamic vocabulary, and often available as a real-time stream of posts.
- The text may contain entity and theme references. Twitter, for instance, uses hashtags to designate the central theme of a tweet; both Twitter and Facebook allow explicit references to members of the network
- There is a wide variety of themes, opinions and viewpoints within the population of social media content at any given time interval, or under any given context (e.g., a newspaper article, or a real-world event that triggers a social media response).
- There is often a non-stationary temporal variation in the theme of any conversation, which sometimes shows crests and troughs in terms of the degree of social interest and volume of conversation. We call newly developing social phenomena as *emergence*, slower variations as *drifts*, sudden increases in intensity as *bursts*, and the decrease in public interest *decays*.
- We often find that the contributors to social media conversations cluster into groups that interact and often reinforce one another, creating a sustained community, at least over a reasonable interval of time.

1.1 A Motivating Example

With the above characteristics in mind, let us consider a journalist who would like to analyze the nature of public opinions from Twitter on the new administration of USA and their policy priorities. She creates a set of keywords to suit her needs and starts collecting from 1% tweet stream with the intention of understanding the following:

1. What is the diversity of themes in the continuously collected tweets? Here diversity is related to *topics* as in the topic modeling literature, and refers to a collection of terms that statistically occur together within any given window of observation. However, to belong to a theme, a term needs to be non-transient, i.e., it cannot appear and disappear in seconds, because a fleeting term does not contribute to the journalist's insight and it is likely to be noise.
2. Is there a significant *burst* on a topic? If so, what real world event does it relate to?
3. What are the top k topics in the last 30 min, hour, day, week on the theme "affordable care act"?
4. Which theme(s) overshadowed the public discussions on the theme "military budget"?
5. Can we predict if a theme observed in social media is going to become a news item in the next few hours?
6. Which user groups are discussing "Obamacare" and "Trumpcare", and how are they evolving with time? This query identifies different user groups discussing the two topics and analyzing the time trends of the group size.
7. Can we assign a theme reference (hashtag in the case of Twitter) to a whole discussion in real-time?

The first query asks for an "aggregate snapshot view" of the world. The remaining queries inquire about the "evolution" i.e., the changing nature of themes. The goal of this paper is to make the case for MINT (Monitor and INterrogate Themes), an information exploration tool that answers snapshot and evolutionary queries over streaming social media and provide deeper insight to day the data so that researcher can build there research strategy through the exploration on data.

2 Prior Work

Prior research in this are can be summarized along two axes [16]. The first considers the way themes are defined, which leads us to Clustering vs. Topic Model Based approaches. The second axis is based on data processing techniques, which leads to batch-based vs. online/real-time processing discussion. **Clustering vs. Topic Modeling.** Most clustering based systems define a topic as a collection of related documents. Using some distance measures, they compute the proximity of documents with one another to construct clusters. Yang et al. [17] use refined hierarchical and online document clustering algorithms to detect events from a news stream. In [1] each document is represented as a TF-IDF vector. However, this approach has many drawbacks. Finding k-nearest neighbors is costly for real-time loads. Often a topic is defined as a coherent set (or cluster) of keywords [7], [11] or hashtags [3], [9]. LDA[6] is the basic topic models initially designed for static document corpora. Algorithms like [4] for microblogs tried to increase the length of documents by aggregating tweets in conversation groups for effective application of topic models on short-texts. To prevent assignment of a tweet to a topic distribution, Twitter-LDA algorithm [19] assigns one topic per tweet. However, it cannot be extended to do online topic inference [5].

We construct a similar dynamic topic model with stochastic Gibbs sampling to compare our results later. In other related work, many researchers [15][16] build topic model to find correlated bursty topics from text streams. However, most of the work can't be extended to perform evolutionary queries in real time or near real time. On the other hand, Yang et al. [17] propose methods for both batch and online event detection.

3 Formulating the Problem

The premises of our approach are (i) most social media and themes can be computed and described by a generic data model (Section 3.1) and (ii) the key behind real-time theme finding is to consider the temporal arrival pattern of message tokens rather than the distribution of topics over documents and the distribution of terms, since they change over time. We hold that while it is possible to estimate word/topic distribution models for already collected data, we cannot really make the assumption that the next chunk of data that will adhere to a pre-determined or recently estimated data distribution. Instead, we would

like to assume that the distribution of content terms will keep changing with time. But, if we consider related terms in social media as tokens, the temporal distribution of tokens arrival is a variant of the Poisson distribution together with an excitatory process. The excitatory process accounts for the common observation that if a token picks up public interest, it will "invite" more social media data in a short amount of time. We further assume that this "pattern of excitement" is itself ephemeral and is likely to wane out within a finite time. Our goal therefore is to find themes in a way to reflect this excitatory arrival behavior, and after a time window of observation, store this data with sufficient parameters to enable us capture the evolutionary nature of the data.

3.1 Data Model

For the purposes of this paper, social media can be viewed as an infinite, time-indexed collection of messages

$$M = \{ts, cr, \overrightarrow{ti}, \overrightarrow{rec}, \overrightarrow{ref}, con\}$$

where ts is a timestamp of a media object $m \in M$, cr is the creator of m, $\overrightarrow{ti}$ is a list of topic indicators, $\overrightarrow{rec}$ is a set of message receivers, $\overrightarrow{ref}$ is a set of entity references and con is the textual content of the message. Using Tweets as our prototypical example, ti corresponds to a hashtag, rec to a "mention" indicated by the @ symbol, and ref to items like URLs or images. Often a ref has the form $\{ref-type, ref-pointer\}$, to distinguish between different referred entities. Although we use Tweets as our running example throughout, the model generalizes to other forms of social media. Let $M[t_1, t_2]$ be a sub-collection of messages with ts in the time-interval $[t_1, t_2]$. We define a **theme** as a vector

$$\omega[t_1, t_2] = \{f_1(ref[t_1, t_2]), f_2(rec[t_1, t_2]), f_3(kw(con[t_1, t_2]))\}$$

where $kw(con)$ is a key-term extraction function operating on the textual content of messages, and $f_1(ref)$ (resp. $f_2(rec)$ and $f_3(kw(con))$) is a summarizing function that selects a subset of $M[t_1, t_2].ref$ elements together their count. We refer to each component of ω as a *theme component*. Since theme components are temporal objects themselves, one can compute a set of time-varying properties of these components. For example, section 4.3 describes how the temporal property *theme velocity* is computed by the MINT system. A **theme set** $\mathbf{T}[t_1, t_2]$ is a set of n themes $\omega_{1..n}[t_1, t_2]$ such that $\omega[i][t_1, t_2]$ is sufficiently different from $\omega[j][t_1, t_2]$ – this is called the *diversity* property of a theme set. The theme extraction algorithm in Section 4.5 creates a theme set, which is subsequently checked to ensurer adequate diversity. Finally, a **theme history** is a structure Ω that records for each component of every theme of a theme set and in each time interval, the theme value (e.g., the hashtags or key-terms), its count and a list of its temporal properties.

4 Our Approach to Solution

Considering the temporal nature of events in social networks data, we borrowed Hawkes process[14]. The Hawkes process is a statistical process over a list of discrete events localized in time, $\{t_i\}$, with $t_i \in \mathbb{R}^+$. It can be represented as a counting process, $N(t)$, which records the number of events before time t. Let the history τ be the list of times at which events take place, $\{t_1, t_2, t_3, t_4, ...t_n\}$ upto but excluding time t. Then in a small time window dt between [0, t), the number of observed event is

$$dN(t) = \sum_{t_i \in \tau} \delta(t - t_i)dt, \tag{1}$$

and hence $N(t) = \int_0^t dN(s)$, where $\delta(t)$ is a Dirac delta function. We assume that the window of size dt is small enough so that only one event can happen in that interval, and hence $dN(t) \in \{0, 1\}$

A conditional intensity function is the stochastic model for the next event time given all previous events. Within a small window $[t, t + dt)$, $\lambda(t)dt$ is the probability for the occurrence of a new event given the history τ :

$$\lambda(t)dt = P\{event \in [t, t + dt)|\tau\} \tag{2}$$

A **homogeneous Poisson process** assumes the intensity to be independent of the history τ and constant over time, i.e., $\lambda(t) = \lambda_0 \geq 0$. However, in an **inhomogeneous Poisson process** , the intensity is also assumed to be independent of the history τ but it can be a function varying over time, i.e., $\lambda(t) = g(t) \geq 0$. In both case, we will use notation Poisson(λ(t)) to denote a Poisson process. Hawkes formulation captures the mutual excitation phenomena between events, and its intensity is defined as

$$\lambda(t) = \gamma_0 + \alpha \sum_{t_i \in \tau} \gamma(t, t_i) \tag{3}$$

where $\gamma(t, ti) \geq 0$ is the triggering kernel capturing temporal dependencies , $\gamma_0 \geq 0$ is a baseline intensity independent of the history and the summation of kernel terms is history dependent and a stochastic process by itself. The kernel function can be chosen in advance, e.g., $\gamma(t, ti) = \exp(-|t - t_i|)$ or learned from data.

4.1 Noise Tolerance

Social media data is polluted with noise (i.e., uninformative messages). For instance, tweets that contain only emoticons, single word tweets, tweets with only hashtags that are not picked up by subsequent tweets are considered uninformative in terms of theme finding. Our scheme must be tolerant to these low-frequency random events. Therefore, ephemeral terms that only show up rarely, must be penalized early to evict them from the theme detection process. While

our technique does not directly address this more systematic form of noise, we make the empirical observation that these pseudo-messages tend to occur in small bursts whose peak count is far lower than informative tweets. Further, if we consider small time windows, the likelihood of their occurrence has a low uniform probability. This observation is utilized in configuring our cache flushing policy (Section 4.7).

4.2 MINT Architecture

The architecture of the MINT system is shown in Figure 1.

1. The **data acquisition module** acquires media objects by applying a set of *seed filtering predicates*. For Twitter, the predicates may restrict the geographic boundaries of a tweet's location or a list of keywords, one of which a tweet must contain to qualify for theme analysis. The seed filter ensures that the data acquired fits within the general analysis goal of the end user.
2. The **data filter module** applies a set of constraints on the data. The constraints are filter conditions that are deemed appropriate for a specific application domain. Commonly used are size filters (e.g., minimal word count of a tweet), content filters (e.g., stop words, emoticon filters), and time filters (e.g., only work hour tweets).
3. The **data preprocessor module** performs a series of operations to prepare the data for the detection algorithm. These operations include punctuation removal, mapping to lower-case, and word lemmatization.
4. The primary processing element of the architecture is the **theme detection module** which accepts the preprocessed data at real-time and computes theme sets (Section 3). The theme sets are continuously updated in memory and are periodically checked for diversity. Themes that are found to be very close are merged.
5. The **data management module** ingests the flushed-out JSON objects, transforms them and pushes them into a JSON store (AsterixDB) [2]. Suitable inidices are developed for the stored theme history (Section 3.1) to answer evolutionary queries like the examples given in Section 1.1.
6. Somewhat orthogonal to the data management module, the **data monitoring module** have a set of continually running queries on the theme sets. These queries, primarily in the form of *top-k* theme components, are output at real-time. These results are used by applications like Event Finder to confirm that the topics identified are indeed found in external sources like Google News.

4.3 Theme Representation

As described in the data model, a theme is defined as

$$\omega = \{f_1(ref), f_2(rec), f_3(kw(con))\}$$

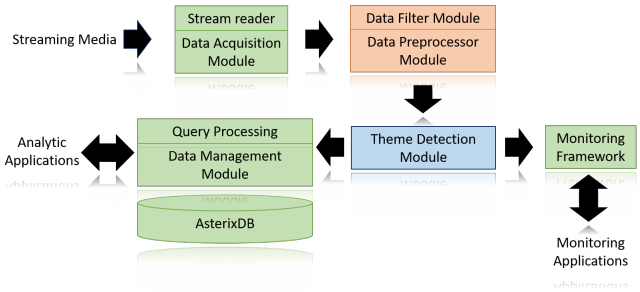

Fig. 1. The MINT Processing Architecture

computed over a time interval, together with a list *attribute-value pairs* for each theme component. The *attribute* represents the string value of the feature (e.g., #IamWithHer) and the *value* represents the number of occurrences of this attribute within this feature within this theme. In the algorithm, we define a theme as a class with the following attributes:

- *Theme Size*: This attribute stores the number of tweets that have been assigned to this theme.
- *Least Recently Used(LRU) caches* : We have 3 LRU caches for hashtags, user-mentions and important tokens receptively – each stores key-value pairs where a key is an entity and value, their frequency. Therefore, the order in which the items are placed in the cache give us a notion of how old that token is in that theme. This helps to compute the recency factor in the affinity function defined later.

Whenever a tweet is assigned to a theme, the size attribute is incremented by 1. Other useful metadata like geolocation of the tweet, links shared in the tweet, etc are also stored as a part of the theme, to facilitate advanced queries on them. A cache is better than a fixed word vector because in a cache, any new word can be added as a key value pair. Before describing the details of the algorithm, let us define an important factor, *affinity function* (Figure 2), which governs theme development.

4.4 Affinity function

The affinity function computes the force with which a theme ω_i attracts a tweet t towards itself. Three factors govern the magnitude of this function- the recency of a similar tweet assigned to the theme in the past(contributes to the point process), the contextual similarity of the incoming tweet to the theme ω_i and finally, the hashtag co-occurrence factor. The tweet t is assigned to the theme with which it has a maximum affinity. If there is no match with any of the current themes(affinity is 0), we create a new theme for this tweet. It is defined as:

$$A(t, \omega_i) = \lambda_1 H + \lambda_2 U + \lambda_3 W + c \tag{4}$$

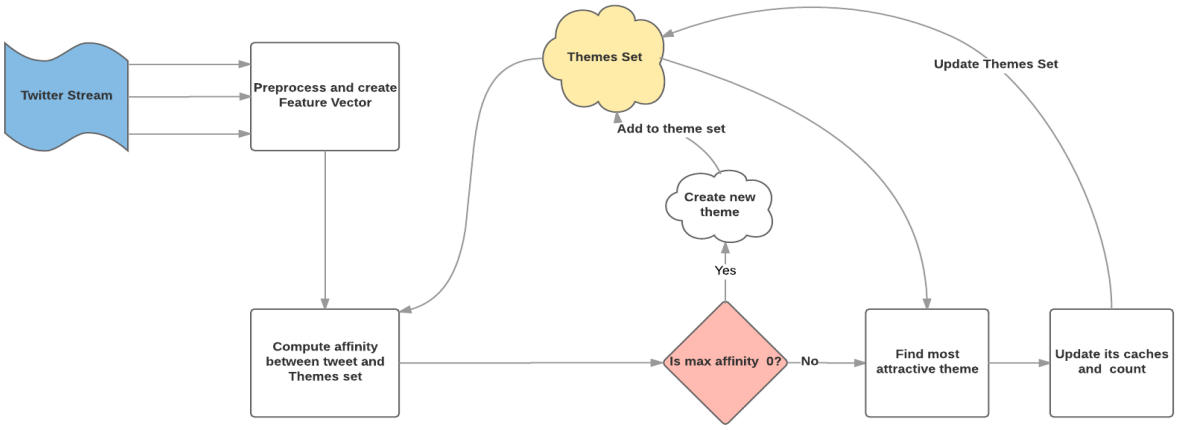

Fig. 2. A flowchart outlining the theme generation process

where λ_i are the weights proportional to the recency of the overlapping members of the attribute. Let $K_{t_j \cap \omega_{ij}}$ be the set of common keys between tweet t's jth list(e.g., hashtag list, users list etc.) and corresponding cache j in Theme ω_i, $j \in \{1, 2, 3\}$. Let $Idx(K, \omega_{ij})$ be the sum of indices of keys K in cache ω_{ij}, the index of more recent key being higher than the older one. Hence $\lambda_j = \frac{Idx(K_{t_j \cap \omega_{ij}}, \omega_{ij})}{length(\omega_{ij})}$. Let $Value(k, \omega_{ij})$ be the frequency value of a key k in the cache ω_{ij}. Also, let $Csm(j)$ be the contextual similarity of an incoming tweet with an entity j(Hashtags, users and words) in the Theme ω_i. We define it as

$$Csm(j) = \frac{\sum\limits_{k \in K_{t_j \cap \omega_{ij}}} Value(k, \omega_{ij})}{\sum\limits_{k \in Keyset(\omega_{ij})} Value(k, \omega_{ij})} \tag{5}$$

where, $H = Csm(j = 1)$, $U = Csm(j = 2)$ and $W = Csm(j = 3)$. Hence, the similarity function is a normalized weighted measure of the degree of overlap between each tweet attribute with a corresponding theme. λ_j gets a high value if all the matching keys are recent in that cache. The underlying assumption is, in a short time interval, if a tweet is recently assigned to a theme, then another similar tweet coming shortly after it might be assigned to the same theme. The term c denotes the hashtag co-occurrence bias which is given by the formula $c_i = h_{matched(i)} - 1$. $h_{matched(i)}$ is the number of common hashtags between the incoming tweet and the theme ω_i. So if there is only 1 hashtag that has matched, then $c = 0$, and there is no co-occurrence bias, and the similarity is calculated based on other terms. If more than one hashtags are common between the tweet and the theme, then c is non-zero and there is a greater probability of the tweet being assigned to the theme with more hashtag matches. This is done under the assumption that a hashtag is a key descriptor of a theme, and a tweet having a

greater overlap of hashtags with a theme, belongs to that theme. If the overlap is same, then other factors govern the assignment.

4.5 Theme Extraction Algorithm

Algorithm 1 Theme Extraction algorithm

```
1: procedure CONSTRUCT MODEL(tweet, themes)
2:     for ω_i in themes do
3:         affinity_i ← A(t, ω_i)
4:         max_a ← max(max_a, affinity_i)
5:     end for
6:     if max_a = 0 then
7:         ω_j ← newTheme()
8:         themes ← themes.append(ω_j)
9:         idx ← themes.length
10:    else
11:        idx ← argmax(affinity)
12:    end if
13:    themes ← themes.assign(idx, tweet)
14:    return idx                          ▷ return the index of theme
15: end procedure
```

The Construct Model method is invoked every time a new tweet arrives. We find the similarity of every tweet with current themes based on the affinity function. If the maximum affinity(max_a) from all current themes is zero, then we create a new theme and assign the tweet to it, else we assign it to the most "attractive" theme. Assignment here refers to updating individual caches of the theme based on the keys in the tweet. Here, idx is the index of the theme where assignment should take place.

Whenever a tweet is assigned to a theme, we update its size count. Its metadata is also stored in a list/cache to construct heatmaps for events, etc. Let $count_a$ be the counts vector containing size of each theme at time=a. We define a velocity vector as:

$$velocity_a = count_a - count_{a-1} \tag{6}$$

$$acceleration_a = velocity_a - velocity_{a-1} \tag{7}$$

where the velocity at time=a is the difference of the count vectors between current and previous time slice. After fixed time intervals, the slowest growing themes are removed, to prevent an outburst of total number of themes. In the actual implementation, we limit the maximum number of themes to a constant. Space for new evolving themes is continuously created by the evacuation scheme described later.Note in the theme creation algorithm that our model is completely data driven and makes no assumption about the distributions from which

the themes arise. In Section 6, we show one example of how this non-assumption helps the quality and speed of our results.

4.6 Computations and Lightweight Cache

As mentioned, we borrowed Hawkes process, however, instead of building costly temporal probabilistic mode, we use temporal excitation to compute the best fit for a tweet among current set of themes in real time, using a deterministic approach. We also claim that our cache-based scheme is as a light-weight implementation of the temporal point process.

In Equation 3 the mutual excitation intensity function, γ_0 is a constant while $\gamma(t, t_i)$ is a kernel function non-increasing in $|t - t_i|$. Thus, $\gamma(t, t_i) \geq \gamma(t, t_j)$ if $event_i$ is more recent than $event_j$. If we only record the last occurrence of an event, instead of considering its last n occurrences, then mutual excitation formula becomes

$$\lambda(t) = \gamma_0 + \alpha.\gamma(t, t_i) \tag{8}$$

where t_i is the last occurrence of the event before time t.

In our approach, an event is the assignment of a token to its respective cache. In the affinity function, observe the contribution from a single cache, say the Hashtag cache. the Affinity contribution for this will be $A(t, T_i)_H = c + \lambda_1 H$. The λ_1 term is a normalized sum of the indexes of matching tokens in the LRU cache, with more recent index having a higher value. These indexes denote the last time that token was assigned to the particular cache. Therefore, between $event_i$ and $event_j$, if $event_i$ attributes to more recent tokens in the cache, the value of $\lambda_1 i > \lambda_1 j$. This is the exact behavior shown by the $\gamma(t, t_i)$ term in the mutual excitation equation.

The c term in the similarity expression can be assumed to be similar to the γ_0 term in Equation 5.6. The Hawkes process treats α to be a constant. However, in our method, we multiply the temporal term (triggering kernel), with the contextual similarity H. Thus, our similarity method can be seen as a superimposition of mutual excitation intensities of different features incorporated in the calculation. In the case of tweets, the features are hashtags, user-names, and string tokens. This is a better design than using a single sliding window to store multiple tweets or a Least Frequently Used(LFU) caches because of the following reasons.

- Since tweets follow a mixed model of stochastic temporal process [18], hashtags, user-mentions and tokens, despite their co-temporality, have very different arrival rates. Decoupling them allow them to update independently and store past information proportional to the length of the individual LRU caches.
- An LRU scheme helps calculate the weights for the recency of overlapping terms between the cache and incoming tweet, in the similarity function. A LFU cache does not store this information.

- A LFU strategy might hinder the detection of theme evolution. Suppose a particular term was very frequent in the past but currently it is not popular. It will still continue to remain at the top of the cache until all other members attain a frequency higher than it, which might take a long time. Thus it reduces the effective utilizable length of the cache. On the other hand, in an LRU strategy, it will be removed if it is not observed for some time, as elements are arranged in the order of their recency, thus allowing themes to evolve into new ones.
- An LRU cache stores information as key value pairs. Saving individual tweets with a sliding window requires more space. Thus, an LRU cache provides efficient utilization of memory which is required for real-time evaluation.

4.7 Runtime Analysis

Let the number of terms in a tweet be n and let K be the number of themes present in the system. The ConstructModel method is invoked for each tweet as it arrives. The affinity function $A(t, T_i)$ compares the tweet t with a theme ω_i to compute an affinity value. The denominator term, in Contextual Similarity $Csm(j)$, the running sum of all values in the cache j is updated everytime the matching keys are updated in cache j. The update and insert operations on the cache take $O(1)$ time. As the affinity computation is done for each term in the tweet t, it takes $O(n)$ time to compute the Contextual Similarity between a tweet and a theme, n being number of terms in a tweet. Similarly, to compute λ_i terms, which determine the indices of each matching term in tweet t in its respective cache, take $O(n)$ time. If there are K themes in the system, the total time to compute the affinity values against all themes, and finding the maximum similarity is $O(nK)$.

The length of each cache is fixed, which makes the model scalable. As new keys arrive in the cache, older ones are removed, limiting the size of the data which is stored as themes. At regular intervals, the theme caches are flushed to avoid overflow of frequency values.

4.8 Theme Recovery

As topics are incrementally added if incoming tweets are not similar to the existing ones, we cannot allow total number of themes to arbitrarily increase in number. Hence there are two ways which we use to limit the number of themes.

- **Velocity Check:** To detect stale events, we monitor the slowest growing clusters based on their velocities and acceleration. After fixed intervals (some k times the polling intervals), such themes are removed so that new ones can be allocated in that space.
- **Inter-cluster distance:** Theme discovery on temporal data leads to instances where many such themes start as different, but eventually evolve into similar ones. To retain diversity among themes, we use Spearman's footrule as a distance function and at regular intervals, cluster similar themes into

one, thus limiting the total number of themes in the system. Spearman's footrule is applicable on ranked lists, as is defined as follows: **Spearman's footrule:** Consider two ranked lists L_1 and L_2. Let $idx_1(i)$ and $idx_2(i)$ be the positions of an element i in lists L_1 and L_2 respectively. The displacement of an element i is defined as $\sigma_i = |idx_1(i) - idx_2(i)|$. Spearmans's footrule is defined as $F(\sigma) = \sum_i \sigma_i$, where $F(\sigma)$ is the distance function.

5 Answering Evolution Queries

As the theme cache is periodically flushed on to disk it is converted to a JSON-like structure:

```
theme-history: [
    theme: { id : long,
    timeInterval: {
         startTime: dateTime,
         endTime: dateTime
      },
    objectSet: [<ObjectID>],
    hashtags: [(<HashTag>,<count>)],
    users: [(<User IDs>,<count>)],
    keywords: [(<strings>,<count>)],
    URLs: [(<URLs>,<count>)],
    hashtag_velocity : float,
    hashtag_acceleration: float,...}
]
```

and stored in AsterixDB. The theme components like hashtag are ordered lists, sorted by count. Each theme in the array represents one time interval; AsterixDB supports time interval as a valid data type, which allows us to natively support Allen's interval operations.The keyword list is indexed as n-grams in AsterixDB, while hashtags and users are indexed as plain strings. The objectSet attribute is an unordered set and contains the IDs of objects (e.g., tweet IDs) whose hashtags, user mentions or keywords are represented in the LRU cache. This representation does not maintain the mapping between the objectID and the theme values like individual keywords. In order to support diversity and evolution queries on this structure, we define a number of operations on this structure. We illustrate these operations using some of the example queries from Section 1.1. **Q1.** *Find k most diverse hashtag themes in the time interval $[t_x, t_y]$.*

To measure diversity between theme vectors, we need to define a *theme distance operator* δ_{th} between them.

While the theme vectors represent histograms, we treat them as ranked lists (the vectors are sorted) and use *Spearman's footrule* as the distance metric. Since the query time interval $[t_x, t_y]$ may not align with the intervals captured from the data. Our approximation strategy is to select data intervals that maximize the overlap between the data and query boundaries using the operation

matchIntervals(t_{low}, t_{high}). Then, pairwise theme distances d_{ij} are computed and a *maxDiversity(D)* operation is used. This operation applies heuristic diversity maximization algorithm [10] and has complexity $\mathcal{O}(kn)$ where n is the cardinality of the distance matrix.

Q2. *Find time intervals in the last 3 hours where a bursting behavior over a Δ interval is observed on hashtags..*

The query seeks all time intervals such that the acceleration of a theme is positive and stays above a threshold θ_a at least for a Δ time period; simultaneously the total count over all hashtags in that interval remains larger than threshold θ_c. Once the set of intervals are retrieved by simple *selection* (and *count*) operations over the themes in every time interval, consecutive intervals in this set must be stitched together using a *condense(set(intervals))* operation.
To set thresholds θ_c and θ_a in an adaptive manner, we maintain a number of system-level statistics, including mean of the average values of total count, velocity and acceleration over different theme components over the last n observations (pragmatically over a day).

Q3. *Which themes overshadowed discussion on "military budget"?*
To answer this query, we first identify themes where the key phrase "military budget" occurs dominantly. This is obtained by selecting themes for which the phrase occurs in the keyword component or if the hashtag #militarybudget occurs in the hashtag component of a theme, and if their count is above the average count of other keywords (or hashtags) within the same theme. Once these themes are identified, their time intervals are condensed as in Q2. We interpret "overshadowed" as follows. Let ω_{mb} be the theme associated with "miliatry budget", the total count of all keywords (resp. hastags) in that bucket be $cnt(\omega_{mb})$, and $[t_x, t_y]$ be *time_interval(ω_{mb}).* The result of the query is a theme ω_{res} such that $cnt(\omega_{res}) > cnt(\omega_{mb})$ and *time_interval(ω_{res}) meets* $\wedge$ *overlaps time_interval(ω_{mb}) overlaps* where *meets, overlaps* are Allen's interval operations and capture intervals that immediately succeed *time_interval(ω_{mb}).*

Q4. *Which user groups are discussing "Obamacare" and "Trumpcare", and how are they evolving with time?*
The first part of the query is similar to Q3, and identifies themes where the two theme keywords are used to retrieve two sets of themes ω_{oc} and ω_{tc}. From these themes we extract the corresponding users (who are mentioned in discussions) using operations *users(ω_{oc})* (resp. ω_{tc}). Since the time intervals associated with the themes are non-overlapping, the user-sets are ordered using *time-order(ω)* operation.

To characterize the nature of evolution of the two time-ordered sets S_1, S_2, several correlation and evolution functions are applied.

1. The composite operations *count(intersection-interval(S_1, S_2))*, *count(union-interval(S_1, S_2))* compute an interval-wise interesection (resp. union) operation of these ordered sets and return the number of total and common users involved in discussions.

2. The operations *autocorrelation-interval*(S_1, S_2) [13] computes the autocorrelation of the two sets across successive time-intervals using the function

$$C(t) = \frac{S_i(t_i) \cap S_i(t_{i+1})}{S_i(t_i) \cup S_i(t_{i+1})}$$

. This determines whether any group continues to participate in the discussion over time. Similarly, *crosscorrelation-interval*(S_1, S_2) computes

$$C(t) = \frac{S_i(t_i) \cap S_j(t_{i+1})}{S_i(t_i) \cup S_j(t_{i+1})}$$

yielding the degree of cross talk across the groups over time.
3. The stationarity function [13]

$$\xi \frac{\Sigma_{t_0}^{t_{max}-1} C(t, t+1)}{t_{max} - t_0 - 1}$$

Intuitively, $1 - \xi$ represents the average ratio of users changing in a group in one observation window.

6 Conclusions and Future Work

In this paper, we propose an improved method to compute themes from tweets in real-time. Our technique is more straightforward to compute in real-time and provides a useful summary of theme dynamics that can be fruitfully used for evolution queries. The system also allows for new themes to be discovered and it slowly removes the obsolete ones. The underlying algorithm can be extended to various other applications like event and community detection. We are measuring performances of the algorithm using various streaming data set. The future work will lead to an exploratory data analyzer infrastructure. We are considering multiple data modeling features like aggregate calculations, theme and topic detection, graph modeling of the data.

References

1. Allan, J., Papka, R., Lavrenko, V.: On-line new event detection and tracking. In: Proc. of the 21st Int. ACM SIGIR Conf. on Research and Development in Information Retrieval. pp. 37–45 (1998)
2. Alsubaiee, S., Altowim, Y., Altwaijry, H., Behm, A., Borkar, V., Bu, Y., Carey, M., Cetindil, I., Cheelangi, M., Faraaz, K., et al.: Asterixdb: A scalable, open source bdms. Proc. of the VLDB Endowment **7**(14), 1905–1916 (2014)
3. Alvanaki, F., Michel, S., Ramamritham, K., Weikum, G.: See what's enblogue: real-time emergent topic identification in social media. In: 15th Int. Conf. on Extending Database Technology (EDBT). pp. 336–347 (2012)
4. Alvarez-Melis, D., Saveski, M.: Topic modeling in twitter: Aggregating tweets by conversations. In: Proc. of 10th Int. AAAI Conf. on Web and Social Media (2016)

5. Blei, D.M., Lafferty, J.D.: Dynamic topic models. In: Proc. of the 23rd Int. Conf. on Machine Learning. pp. 113–120. ACM (2006)
6. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (Mar 2003)
7. Cataldi, M., Caro, L.D., Schifanella, C.: Personalized emerging topic detection based on a term aging model. ACM TIST **5**(1), 7:1–7:27 (2013)
8. Dasgupta, S., Gupta, A.: Analyzing community dynamics in social media. In: Proc. of the 1st VLDB Workshop on Social Data Analytics and Management (SODAM) (2016)
9. Feng, W., Zhang, C., Zhang, W., Han, J., Wang, J., Aggarwal, C., Huang, J.: STREAMCUBE: hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream. In: Proc. of 31st IEEE Int. Conf. on Data Engg.(ICDE). pp. 1561–1572 (2015)
10. Glover, F., Kuo, C.C., Dhir, K.S.: Heuristic algorithms for the maximum diversity problem. Journal of information and Optimization Sciences **19**(1), 109–132 (1998)
11. He, D., Jr., D.S.P.: Topic dynamics: an alternative model of bursts in streams of topics. In: Proc. of the 16th ACM SIGKDD Int. Conf. on Knowl. Discovery and Data Mining. pp. 443–452 (2010)
12. Lin, Y.R.: Event-related crowd activities on social media. In: Social Phenomena, pp. 235–250. Springer (2015)
13. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. Nature **446**(7136), 664–667 (2007)
14. Rizoiu, M.A., Xie, L., Sanner, S., Cebrián, M., Yu, H., Hentenryck, P.V.: Expecting to be hip: Hawkes intensity processes for social media popularity. In: Barrett, R., Cummings, R., Agichtein, E., Gabrilovich, E. (eds.) Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017. pp. 735–744. ACM (2017), http://doi.acm.org/10.1145/3038912
15. Wang, X., Zhai, C., Hu, X., Sproat, R.: Mining correlated bursty topic patterns from coordinated text streams. In: Proc. of the 13th Int. Conf. on Knowl. Discov. and Data Mining. pp. 784–793 (2007)
16. Xie, W., Zhu, F., Jiang, J., Lim, E., Wang, K.: Topicsketch: Real-time bursty topic detection from twitter. IEEE Trans. Knowl. Data Eng. **28**(8), 2216–2229 (2016)
17. Yang, Y., Pierce, T., Carbonell, J.G.: A study of retrospective and on-line event detection. In: Proc. of the 21st Int. ACM SIGIR Conf. on Research and Development in Information Retrieval. pp. 28–36 (1998)
18. Zadeh, A.H., Sharda, R.: Modeling brand post popularity dynamics in online social networks. Decision Support Systems **65**, 59–68 (2014)
19. Zhao, W.X., Jiang, J., Weng, J., He, J., Lim, E.P., Yan, H., Li, X.: Comparing twitter and traditional media using topic models. In: Proc. of Euro. Conf. on Info. Retrieval (ECIR). pp. 338–349. Springer (2011)