See What's enBlogue – Real-time Emergent Topic Identification in Social Media[*]

Foteini Alvanaki ♣ , Sebastian Michel ♣ , Krithi Ramamritham • , Gerhard Weikum ♠

♣ Saarland University, Germany • IIT Bombay, India ♠ Max-Planck Institute Informatics, Germany

♣ {alvanaki|smichel}@mmci.uni-saarland.de • krithi@iitb.ac.in
♠ weikum@mpi-inf.mpg.de

ABSTRACT

With the increasing popularity of Web 2.0 streams, people become overwhelmed by the available information. This is partly countered by tagging blog posts and tweets, so that users can filter messages according to their tags. However, this is insufficient for detecting newly emerging topics that are not reflected by a single tag but are rather expressed by unusual tag combinations. This paper presents *enBlogue*, an approach for automatically detecting such emergent topics. EnBlogue uses a time-sliding window to compute statistics about tags and tag-pairs. These statistics are then used to identify unusual shifts in correlations, most of the time caused by real-world events. We analyze the strength of these shifts and measure the degree of unpredictability they include, used to rank tag-pairs expressing emergent topics. Additionally, this "indicator of surprise" is carried over to subsequent time points, as user interests do not abruptly vanish from one moment to the other. To avoid monitoring all tag-pairs we can also select a subset of tags, e.g., the most popular or volatile of them, to be used as seed-tags for subsequent pair-wise correlation computations. The system is fully implemented and publicly available on the Web, processing live Twitter data. We present experimental studies based on real world datasets demonstrating both the prediction quality by means of a user study and the efficiency of enBlogue.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval —*information filtering, selection process*

General Terms

Algorithms, Performance

Keywords

Web 2.0 streams, emergent topics

[*]A demonstration of enBlogue [2] was given at SIGMOD 2011, in Athens, Greece (... where it identified, during the demonstration session, the tag pair "Athens Syntagma", describing the riots taking place at the Syntagma square in front of the Greek parliament, ≈400m away from the conference site).

This work has been supported by the Excellence Cluster on Multimodal Computing and Interaction (MMCI).

1. INTRODUCTION

Web 2.0 streams, like blog postings, micro-blogging tweets, or RSS feeds from online communities, offer a wealth of latest news about real-world events and societal discussion. For example, natural disasters, military incidents, celebrity scandals, or an upcoming movie premiere are sometimes covered more candidly and in particular more promptly in these forums than on official news channels. On the other hand, this wealth may easily overwhelm users which are not interested in continuously checking the entire stream of information at all times, but prefer automatic notifications of recent events which are expressed through *emergent topics* arising in social media. For instance, users might be interested in the latest news on certain celebrities or political happenings. We can think of this as a continuously updated portal, summarizing the gist of everything new.

Documents carry information in the form of categorical information or simple tags, i.e., short textual annotations that describe the content of the document. We refer to such information as tags in this paper, which can be either given explicitly or can be derived means of topic classification or *named entity tagging*. However, these tags are in general rather imprecise and highlight only one aspect at a time. For instance, emergent topics with tags such as "France", "Cuba", or "Vancouver" might not draw much attention and topics like "Justin Bieber" or "Lady Gaga" might be interesting only to the respective fan groups. Naturally, a context is needed to make sense of certain observations. In this work, we aim at *monitoring emergent topics that consist of pairs* (or in general, sets) of tags. "Vancouver Riots" pointing on the violence after the Stanley cup defeat of the Vancouver Canucks, "France nuclear power plant" reporting on the explosion in one French nuclear power plant in mid September 2011, and "earthquake Cuba" telling us about the happening and the location of the event. Note that such combinations could come directly from a taxonomy, but immediately after

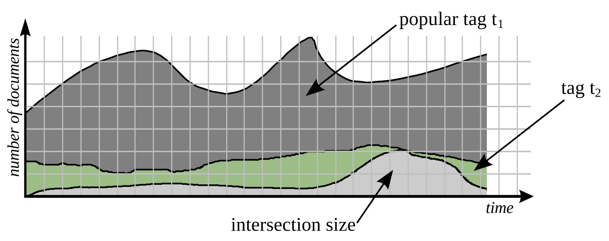

Figure 1: Interesting shift in correlation of two tags.

the event, no category for this topic would exist in the taxonomy; it may take the Web 2.0 platform providers a few hours or even days to create categories like "France nuclear plant explosion". Such manual creation of categories might not be desirable or feasible, given that many events are very volatile and of interest for only some hours/days.

Tag combinations not only carry much more information than single tags, they also allow us to *identify events right after they happen*; small changes in the popularity of a topic (used for trend detection) are much more visible for narrow topics (e.g., "Obama scandal") compared to generally popular ones (e.g., "Obama"), where an additional amount of a couple of dozen new messages can hardly be observed.

Spotting trends is very different from identifying popular topics; we are interested in sudden changes in the popularity, expressed through unusual overlap of tag usage, computed based on co-occurrence statistics. This is much more meaningful for our purpose than looking at the pairwise covariance of single-tag frequency statistics (i.e., covariance/correlation between time series).

Figure 1 gives an example of two tags and the corresponding overlap in terms of the number of documents (within a time window of interest) that contain both tags. The figure shows the behavior over time for two tags, a popular tag t_1 and a less popular tag t_2. By inspecting the size of the intersection over time, we see that the peaks in the popular tag have no influence on the size of the overlap. In contrast, we observe in Figure 1 that the size of the intersection increases towards the end of the shown time interval, an explanation for which cannot be given solely by looking at the individual frequencies of t_1 and t_2.

This paper addresses the issue of detecting the onset of an emerging topic in Web 2.0 streams. We focus on new topics that are not yet reflected in the taxonomy, but we leverage the fact that postings are often pre-annotated with categories or named entities by their authors. Alternatively, we may use machine-learning tools for automatically classifying a posting or extracting entities.

The identified emergent topics open up opportunities for a full exploration of social media given the detected tag sets as input. Hence, we see enBlogue as a *portal to stay tuned* on issues that become *"en vogue"* while inspecting only the very essence of the massive amount of available information.

In the following, we refer to explicitly given categories as *tags*, and use the word *topic* solely for implicitly observed emerging topics.

1.1 Contributions and Outline

We consider unexpected but significant changes in the correlation of tag pairs as an indicator of an emergent topic. We refer to such changes in correlation patterns as *shifts*. For example, consider two tags "Cuba" and "earthquake", which used to be uncorrelated. When a high overlap of items assigned to both tags is observed, we may interpret this as the onset of an emergent topic.

A simple approach would define the correlation between two tags to be given by the frequency of the intersection of these tags or the Jaccard coefficient. In practice none of these measures alone is able to identify accurately a shift. We use a combination of these two measures in order to identify emergent topics. To escape the scalability problem due to the quadratic number of tag pairs, we can choose a subset of tags, called seed tags, and monitor only the tag pairs that consist of at least one seed tag. We choose seed tags to be the most popular tags at every evaluation since topics consisting of popular tags are more likely to be interesting to a large fraction of users. However, seed tags may also be chosen based on their volatility (i.e., the first derivative in frequency), or a combination of volatility and popularity.

In this work we present an in-depth description of our framework, algorithms, and implementation details. Our approach has high predictive power regarding topics that are about to become hot. It can handle high rates of new data items, by using efficient techniques for tracking correlation measures and identifying interesting shifts. We experimentally evaluate the approach using real world data, studying the performance and accuracy.

Our solution is *fully implemented* in a prototype system called *enBlogue*, as we aim to identify trendsetters from blogs (and other streams) that are "en vogue". The system can be tried out live on the Web at URL `http://blogue.mmci.uni-saarland.de`, continuously processing 10% of all Twitter data. The result ranking is updated every hour.

The paper is organized as follows. Section 2 discusses related work. Section 3 presents our framework for Web 2.0 streams. Section 4 discusses statistical measures for characterizing interesting seed tags and tag pairs over time. Section 5 presents the algorithms used in enBlogue and the necessary implementation details. Section 6 reports on the implementation details of our competitor. Section 7 gives experimental results with real-life datasets from news, blogs, and tweets. Section 8 concludes the paper.

2. RELATED WORK

There is a vast amount of literature on *stream mining* and *time-series analysis*; overviews are given in [16, 14]. In particular, the state-of-the-art on efficient frequent-item mining is fairly mature [12], based on a rich suite of statistical synopses and other approximation techniques. However, these methods do not carry over to detecting correlations between items evolving over time. Analyzing correlations between time-series is a standard problem, but considers only time-series of scalar values as opposed to our richer setting of tagged document sets. Moreover, with a few exceptions such as [33], time-series correlation mining does not scale to many thousands of per-item time-series. Generally, our problem differs from traditional time-series mining in two ways: 1) We aim to find *shifts*, i.e., strong gradients, in correlations, rather than correlations per se. 2) Instead of scalar values, we consider streams of *tagged postings* and thus face a much richer structure in the underlying data.

Interestingness and novelty measures have been studied in the context of association-rule mining (e.g., [30, 21] and references given there), incremental clustering of documents (e.g., [19]), and novelty detection in time-series (e.g., [24]).

Our measures resemble the ones in this prior work, but our problem is fairly different. Interestingness is a time-invariant or snapshot-oriented measure in the literature. Novelty typically refers to anomalies and surprising patterns in a time-series, based on regression models. In that sense, shift detection alone is a well studied problem. However, our problem setting requires detecting prominent shifts in the correlations among a huge number of different tag-wise organized but overlapping streams.

Mining online news for events was a hot issue in information retrieval a decade ago (see, e.g., [1, 31, 20, 23]). The focus was on identifying keyword contexts that constitute specific events (e.g., political elections, sports competitions, etc.). This early work did not consider implicitly emerging topics, and disregarded scalability issues. Recent work on discovering and tracking *events and themes in blogs* (or even tweets) include [4, 5, 6, 7, 15, 22, 28, 32, 18]. This line of work focuses on single themes, as expressed, for example, by distinctive phrases. There is no analysis of emerging correlations. Many of these approaches compute clusters or latent-aspect models of related keywords or annotations; these clustering or EM-style learning methods are not able to cope with the scale and dynamics of our problem setting. The Taglines project [13] on mining the evolution of tag clouds emphasizes visualizations, not detecting interesting correlation trends.

The recent work by Budak et al. [9] uses structural information obtained from the social network graph for trend detection, focusing on single tags rather than tag combination. According to [9], structural information helps fighting spam, hence, increasing result quality. However, availability of user graphs can not be taken granted in a general scenario, considered in our own work.

Das Sarma et al. [27] aim at detecting temporal relationships between entities. First, co-peaking entities are extracted from the stream, which are then grouped according to co-occurrence. Subsequently, these groups are separated depending on time intervals.

The general idea of first detecting bursty tags is also the core idea in the works by Cataldi et al. [11], and Mathioudakis and Koudas [26]. Cataldi et al. focus mainly in the trend analysis and they seem to disregard any runtime issues that could harm a real-time application. In their work they consider a topic to be emergent if it can be described by at least one bursty keyword, which, as we claim at the introduction of this work, can be misleading.

The work by Mathioudakis and Koudas [26] is the closest competitor to our enBlogue system. Their system, coined *Twitter Monitor*, discovers topic trends in tweets, by detecting bursty single tags. Tag groups are formed by clustering co-occurring bursty tags or using spectral analysis. This approach is quite different from our setting: unlike looking solely for bursty tags, we detect shifts in tag *correlations* as they dynamically arise.

3. PROBLEM STATEMENT AND METHODOLOGICAL FRAMEWORK

We consider a stream of incoming documents (news, blogs, tweets), each carrying a timestamp reflecting the time of creation. We assume that the document metadata includes sets of tags, or alternatively an annotation mechanism (e.g., an automatic classifier) that assigns tags to documents. In both cases, the stream of `(timestamp, document)`-pairs can be interpreted as a stream of `(timestamp, document, tagset)`-triplets. Tags would also include taxonomic categories or named entities extracted from the document contents. We consider in-order streams, where items arrive in timestamp order. The problem then is to compute a ranking of tag pairs by their likelihood that they contain an emergent topic.

notation	**explanation**
tuple	- a triple consisting of timestamp, document identifier and set of tags/entities
tag	- a simple annotation to a document, explicitly given or extracted on the fly. Extracted named entities are also treated as tags.
topic	- a pair of tags (or in general sets of tags)
W	- size of the sliding window that determines at each time point the set of recent tuples to consider
min	- minimum number of documents to qualify a tag for further analysis (otherwise it is considered noise)
ϱ	- the number of past values stored for each tag as the history, used in the prediction task
k	- the size of the final ranking of emergent topics

Table 1: Overview of used notation.

Looking at the raw document stream at some point in time does not give insights on popular tags or tag correlations. A standard approach to look at continuous time ranges of a data stream is to define a sliding window. Such a window, of size W, defines a temporally "coarser" view, consisting of all documents from the original stream with a timestamp inside the current window, also called active documents. At evaluation time we perform our analysis using all currently active documents.

The size of the sliding window is application dependent. For instance, when considering news streams obtained from blogs or newspapers, a window size of $W = 6h$, $12h$, or $24h$ seems to be reasonable. For Twitter, much shorter sizes, for instance, $W = 1h$ or $2h$ is more appropriate.

It is straightforward to count the number of documents for a particular tag, to look at the number of documents that contain two or more specified tags at the same time, or to apply more sophisticated means to capture tag correlations.

To further enrich the incoming documents, we make use of an entity tagging approach which extracts named entities from the document contents.

We aim at identifying emergent topics consisting of pairs (or in general sets) of tags. Emergent topics are identified by observing changes in the popularity of topics; an emergent topic exhibits an unexpected behavior. We believe that unexpected behavior triggers user interest. More precisely, as introduced later, we model "interestingness" as a mixture of the topic's level of emergence and general popularity.

This problem is challenging for three reasons:

- *Dynamics and Diversity:* New items arrive at a high rate with many different tags. The rates for different tags can vary rapidly.
- *Unpredictable Interestingness:* Hot topics are not the same as emergent topics. We cannot simply mine in-

teresting tag pairs based on their joint steady-state popularity.

- *Scalability:* The first two challenges make it impossible to rely on precomputed statistics. Thus, we have to dynamically analyze tag-pair correlations and their change patterns. The Web 2.0 settings that we consider here have taxonomies with tens of thousands of tags (categories).

Our framework consists of three stages:

1. **Seed tag selection:** As the name indicates, seed tags are used to trigger the computation in the following steps and can be determined based on different criteria, such as popularity or volatility.

2. **Correlation tracking:** For each tag pair that contains at least one seed tag, we keep track of their correlations. For each such pair, we continuously monitor the documents that are annotated with both tags.

3. **Shift detection:** Based on the correlations computed in the previous stage, we inspect the temporal changes in the correlations to detect emergent shifts.

The third stage provides for each considered pair a quantitative measure for the shift within a configurable time period. These values are used to rank tag pairs and to report the top-k most interesting emergent ones.

The handling of suddenly correlated tag pairs can be generalized to n-tuples of pairs for $n > 2$. In our framework and algorithms, the required additional steps are very straightforward.

4. THE 3 CORE STAGES AND THE FINAL SCORE COMPUTATION

4.1 Identifying Seed Tags

To identify emergent topics (i.e., shifts in tag-pair correlations), we start with a subset of all possible tag pairs. We generate topic candidates by first selecting interesting seed tags and then analyzing only correlations of pairs having at least one seed tag. Seed tags are chosen based on popularity. The rationale is that for a tag pair to become interesting and form an emergent topic, at least one of the two tags should be "hot" by itself. Popularity is easy to measure as it merely requires computing a sliding-window average of the popularity of elements in the document stream. Additionally or alternatively, we can also monitor the first derivative of the popularity function, thus identifying tags with high volatility and select these as seed tags.

4.2 Measuring Tag Correlations

Calculating the correlation of two tags is based on the documents that carry these tags.

Consider two tags t_1 and t_2 and the corresponding document sets S_1 and S_2 that have them as tags. We are interested in calculating how correlated these two tags are, based on their corresponding sets of documents currently present in the sliding window. Generally speaking, a good measure should reflect (i) how important the topic is to the community of users interested in any aspect of it, i.e., how likely it is to see both tags together compared to the separate occurrences (local importance), and (ii) how important the topic is to the whole community, i.e., how likely it is to see both tags together (global importance)

We opted for the Jaccard coefficient as the measure of local importance as it is a well studied/accepted measure for the similarity of two sets. Although it provides an estimate of how strongly connected two tags are, it does not take under consideration how important the pair of these tags is compared to the rest of the pairs in the current window. An estimation of this importance is given by the average number of documents in the current window that contain both tags. We use the product of these two measures and we obtain the following formula for calculating, at any point in time, the correlation of two tags t_1 and t_2, where N is the number of documents in the current window:

$$corr(t_1, t_2) := \frac{\overbrace{|S_1 \cap S_2|}^{local\,importance}}{|S_1 \cup S_2|} * \frac{\overbrace{|S_1 \cap S_2|}^{global\,importance}}{N} \tag{1}$$

4.3 Shift Detection

As we have already mentioned our goal is to present the user with a list of the top-k most interesting emergent topics. A topic is considered to be emergent when its behavior deviates from the expected, similar to [24]. The more it deviates the more emergent the topic is.

The behavior of a topic is said to be expected if we can predict it by taking into account its previous behavior. The prediction of the behavior of a topic is attempted with the use of exponential smoothing. Exponential smoothing is a forecasting technique that uses a weighted moving average of past data as the basis for the forecast. The process gives greater weights to most recent observations and smaller weights to observations in the more distant past. The reason for this is that the future value may be more dependent upon the recent past. The exponential smoothing equation is the following:

$$\hat{v}_t = av_{t-1} + (1-a)\hat{v}_{t-1} \tag{2}$$

where $0 < a \leq 1$, $t > 0$, v_{t-1} is the previously observed value and $\hat{v}_{t-1}$ is the previously predicted value

A topic is emergent if its *real correlation is larger than the predicted value.* The difference between these two values is called prediction error and it is defined as

$$error(t) = v_t - \hat{v}_t$$

The prediction error has the property to be greater when the difference between the predicted and the computed correlated values is large. As a consequence, topics that were not strongly correlated in the past and are a bit more correlated now are not detected in the top-k emergent topics. To avoid this we use the relative prediction error, defined as

$$relativeError(t) = \frac{v_t - \hat{v}_t}{v_t} \tag{3}$$

4.4 Overall Scoring

The relative prediction error detects the topics that are emergent but it cannot detect the emergent topics that are interesting at the same time. We believe that a good measure for the interestingness of a topic is popularity. So we define the score of an emergent topic to be

$$score(t) = \frac{relativeError(t)}{|\ln(popularity(t))|} \qquad (4)$$

We chose to divide by the absolute value of the natural logarithm of the popularity and not just to multiply with the popularity because the natural logarithm has the property of dampening the effect popularity has on the final score. At the same time, the bigger the difference of the relative errors for two topics the more difficult it is for the popularity to affect the ranking of the topics. Consider the following example:

Suppose we have four topics t_1, t_2, t_3 and t_4. Suppose that the relative errors for the topics t_1 and t_2 are related by $\frac{relativeError_1(t)}{relativeError_2(t)} = 1.010$ and the relative errors for the topics t_3 and t_4 are related by $\frac{relativeError_3(t)}{relativeError_4(t)} = 1.100$. In order for the popularity of topic t_2 to affect the ranking of the topics t_1, t_2, i.e. $score_2(t) > score_1(t)$, $\frac{popularity_2(t)}{popularity_1(t)} = 1.047$. For the popularity of topic t_4 to affect the ranking of the topics t_3, t_4 $\frac{popularity_4(t)}{popularity_3(t)} = 1.520$.

In the simple case, where the relative error is just multiplied with the popularity, the relationship of the popularities for the topics t_1, t_2 is sufficient to be $\frac{popularity_2(t)}{popularity_1(t)} = 1.011$ in order to affect the ranking. For the topics t_3, t_4, $\frac{popularity_4(t)}{popularity_3(t)} = 1.101$.

So, in the example it is shown that by using the natural logarithm of the popularity it is more difficult to affect the ranking of the topics. Moreover an increase in the ratio of relative errors of 8.9% (from 1.010 to 1.100) needs an increase in the ratio of popularities of 45.2% (from 1.047 to 1.520) for the ranking to be affected. In the simple case this increase in the ratio of popularities is just 8.9% (from 1.011 to 1.101), the same as the increase in the ratio of relative errors.

This behavior of the natural logarithm is a desired behavior. Since we are interested in emergent topics and not in hot topics, we want to avoid the situation that the overall score depends too much on the popularity, but at the same time we want a big difference in the popularity, compared to the difference in the relative errors, to be able to change the ranking of a topic by affecting its score.

4.5 Score Smoothing

Naturally, if a topic's behavior does not change much with time, the capability of predicting the next value improves and the topic is not considered emergent anymore. Intuitively though, we can say that a user does not loose interest in a topic from one moment to another, e.g., in case of a big scandal that causes almost every newspaper to write articles about the story, but much fewer on the next day or even having a constant number of articles per day in the following days, which is then not a big surprise anymore, but it is still of some interest to the user. Hence, the interestingness of one day, e.g., the first day the event occurred, should carry over to other days, with a dampening factor, obviously.

This intuition is confirmed by the observation made in articles from the New York Times archive. We used this source as an example as we believe that the newspaper editors have a good understanding of how long an event is interesting to the consumers. We discovered that very often the number of articles referring to one specific event decreases through

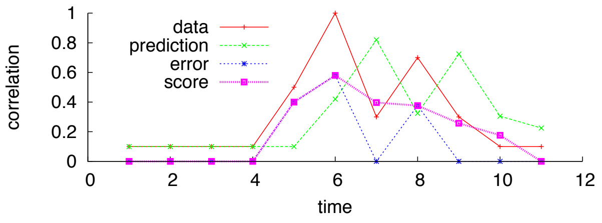

Figure 2: Interesting shift in correlation of two tags.

time by a factor of

$$penalty(\Delta_t) := e^{-\lambda \Delta t}$$

where Δt is the distance in time from the moment when the most articles for the topic were written. After testing a large number of topics we obtained on average a value of $\lambda = 0.38$. This means that an interestingness score of yesterday gets dampened by a factor of $e^{-0.38} = 0.68$, the value obtained two days ago by a factor of 0.46, and so on.

This way, we get, per topic, a dampened interestingness value for each point in the past, plus one current interestingness value, which is obviously not dampened as $\Delta t = 0$ This way, old "surprises" have a chance to influence a topic's ranking today.

$$score(\tau) := max_{t \leq \tau} \left\{ \frac{relativeError(\tau)}{|\ln(popularity(\tau))|} * penalty(\tau - t) \right\}$$

The dampening factor found from the New York Times articles study seems to produce nice results when applied to blog posts but not when applied to tweets. This is obviously due to the fact that topics discussed in Twitter have a fast refresh rate, while topics discussed in blogs have a refresh rate that resembles that of newspapers.

The dampening factor $e^{-0.38\Delta t}$ has a half life of 1.8 evaluations. Since topics in Twitter change very often we are obliged to evaluate new emergent topics more frequent, e.g. every hour instead of one day which is the case for newspapers. This means that with the use of the above dampening factor a topic looses its half score after the second hour. This is too fast and a smaller dampening factor is needed. By experimentally testing various dampening factors, we concluded that the factor $e^{-0.2\Delta t}$, which has half life of 3.4 evaluations, is more appropriate for Twitter.

Since the number of previous values that we can store for each topic is limited, the final score may not be the greatest of all the observed scores of this topic but just the greatest of the scores that we store. In practice this is not a restriction as the dampening factor de-facto erases former scores after a couple of time units in any case. Figure 2 shows an illustration of the score derivation.

5. ENBLOGUE IMPLEMENTATION

With *enBlogue* we have implemented a full-fledged prototype system. The implementation is done in Java 1.6 and follows the standard concepts of a push-based architecture for stream processing. At the data source level, it consists of several wrappers that either consume live streams or replay existing datasets for experiments. Data is represented in an array n-tuple format, consumed by stream operators, and pushed along producer-consumer edges in query-processing plans. The filtered and manipulated data items finally arrive at sinks in the operator DAG. One of the sinks is the operator that computes the final rankings of emergent topics and sends them to our Web server for visualization.

In Figure 4 the workflow is shown. Every block in the illustration represents an operator. Each operator receives data from the previous operator, processes them and pushes the results to the next operator.

The *Entity Tagger Operator* is a preprocessing operator used to further enrich the set of tags. It uses an automatic entity extraction tool and finds, in the document, entities like people, organizations and places. These entities are added to the documents and treated afterwards as common tags. Essentially it works by checking the document content for phrases for which an article in Wikipedia exists.

The *Sliding Window Manager* is responsible for keeping track of the documents arriving in the system. It maintains the window and checks if the incoming document has a creation time inside the window. The size of the time sliding window depends on the dataset. The Sliding Window Manager blocks the information arriving to it until it receives the first document with timestamp greater than the upper limit of the window. When this happens it releases the tuples currently in the window to the rest of the operators. We call this moment *evaluation point*.

The operators that receive information at the evaluation point execute our main algorithm. While the Entity Tagger operator and the Sliding Window Manager execute continues procedures, i.e., a document is parsed as soon as it arrives to the system, the operators that are part of our main algorithm are executed only after every evaluation point and until they process the data they received. We call this period of time evaluation phase.

During the evaluation phase, the seeds are selected from the *Statistics operator* (Figure 3, Algorithm: `FIND SEEDS`). The *Correlation Computation operator* takes the seeds and the total tags and finds the pairs of related tags. Two tags are considered to be related if they co-exist in more than *min* documents. For this (Figure 3, Algorithm: `FIND EMERGENT TOPICS`, Line: 4) we use a number of threads that are running in parallel. Each thread gets a subset of the seed-tags and is responsible of examining only the pairs having at least one seed-tag from the subset the thread is assigned. All threads should finish their processing before the results are sent to the Shift Detection Operator.

The *Shift Detection operator* identifies emergent topics and is the final operator executing a part of our main algorithm. This operator is responsible for detecting shifts on related tag-pairs, using the formulas described in Section 4.3 (Figure 3, Algorithm: `FIND EMERGENT TOPICS`, Lines: 5-20) and score them using the formula described in Section 4.5 (Figure 3, Algorithm: `SCORE A PAIR OF TAGS`). The output of the Shift Detection operator is the final result.

As a last phase, before sending the results to the Web Server, we have a post-processing phase. During this phase the *Diversification operator* groups tag-pairs that refer to the same event. Two tag pairs are placed in the same group if they co-exist in 80% of the documents. The purpose of this procedure is to maximize the number of different events presented to user by avoid showing multiple tag-pairs that refer to the same event. After the post-processing phase we do not have tag-pairs anymore but tag-sets, which are sent

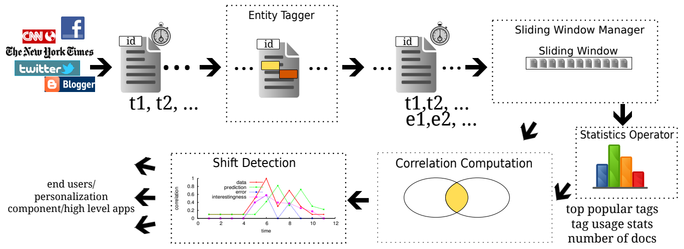

Figure 4: Framework Workflow Illustration

to the Web Server.

Due to the design decisions that we have made (data is pushed from one operator to the other) adding new operators is a simple and straightforward procedure. For example, we could add a personalization operator which will receive the results from the Shift Detection Operator. The *Personalization Operator* can use standard IR techniques such as Language Models or methods based on tf*idf scores [25] to select from the set of the results only those that satisfy the users preferences by computing scores for topics based on the scores of the documents annotated.

The Web-based user interface of enBlogue provides real-time monitoring and user notifications in a push-based manner (i.e., without the user having to continuously poll the server for updates on emergent topic rankings). This has been implemented using AJAX technology, more specifically, the push-based variant offered by the open-source *Ajax Push Engine (APE)* [3]. APE includes a Javascript framework for real-time data streaming to Web browsers, without any installations on the client side.

6. IMPLEMENTATION OF AN ALTERNATIVE APPROACH

We compare the performance of our approach with the *TwitterMonitor* approach described in [26], which performs trend detection in two steps. In the first step it identifies *bursty keywords* and in the second step it groups the identified bursty keywords based on their co-occurrence. Every set of bursty keywords is considered an event.

In TwitterMonitor, a keyword is called "bursty" if it is encountered at an unusually high rate in the twitter data stream. In our implementation of TwitterMonitor, we use hashtags (i.e., of the form #xyz) and named-entities found in tweets, as keywords, and call them simply tags in the following. In order to find bursty tags we follow a procedure very similar to the shift detection procedure of enBlogue, described in Section 4.3. The difference is that instead of processing tag-pairs and their correlations, as is the case in enBlogue, in TwitterMonitor we process tags and their popularities. (Figure 3, Algorithm: `FIND BURSTY TAGS`)

After identifying the bursty tags we group them into disjoint sets, exactly as described in [26] (Figure 3, Algorithm: `GROUP BURSTY TAGS`). Two tags t_1 and t_2 may belong to the same set only if they co-exist in at least one tweet and if t_1 belongs to the same set with keyword t_2 and keyword t_2 belongs to the same set with tag t_3 then the keywords t_1 and t_3 also belong to the same set. We consider every set of tags to represent an event or topic.

To determine a final ranking of the top events, we assign

```
FIND EMERGENT TOPICS
1  for each tag in window
2      if it exists in more that min documents
3          create set of docs containing it
4  find the related pairs of tags found in this evaluation...
           ... where at least one of them is a seed tag
5  for each old pair of tags
6      remove the oldest correlation and popularity value
7      if it was found in this evaluation
8          compute the current correlation and popularity of it
9          add the new correlation and popularity values
10     else
11         add zero as the new correlation and new popularity
12     compute score of pair
13 for each new pair not found in the old ones
14     add zeros as the previous correlation and popularity values
15     compute the current correlation and popularity of it
16     add the new correlation and popularity values
17     compute score of pair
18 remove all pairs that have not been found ...
           ... for $\varrho$ evaluations
19 sort events according to scores
20 return the $k$ top events
```

```
SCORE A PAIR OF TAGS
1  predict current correlation value based on histoty
2  for each $\varrho + 1$ correlation values
3      $dt = number\ of\ evaluation\ passed\ since\ this\ value\ was\ computed$
4      $error = \frac{(computedCorrelaion - predictedCorrelation)}{computedCorrelation}$
5      $reversePopularity = \frac{1}{|log(popularity)|}$
6      $scoreTemp = error * reversePopularity$
7      if Twitter dataset
8          $score = scoreTemp * e^{-0.2*dt}$
9      else
10         $score = scoreTemp * e^{-0.38*dt}$
11     if maximum score computed so far
12         store
13 return maximum score
```

```
FIND SEEDS
1  find all tags in window
2  for each tag in window
3      if it does not exist in more than $min$ documents
4          ignore it
5  sort the remaining tags based on their popularity
6  keep only the top $k$ tags
```

Figure 3: The three core algorithms used in enBlogue

a score to each event. As [26] does not mention a way to compute this, we adjust Formula (4), used in enBlogue, to TwitterMonitor: We replaced the $relativeError(t)$ in Formula (4) with the average burstiness of all tags which are part of an event.

In enBlogue the relativeError of a topic is an indicator of how bursty the topic is. In TwitterMonitor the average burstiness is also an indicator of how bursty a topic is. Overall, Formula (4) is changed in TwitterMonitor to

$$score(t) = \frac{averageBurstiness(t)}{|\ln(popularity(t))|} \quad (5)$$

where *popularity* is defined to be the average number of documents containing all tags of a topic.

7. EXPERIMENTS

We used the enBlogue prototype to conduct a series of experiments, with different datasets. For measurements, the datasets were replayed from files that contain the raw data. All computations were performed on the fly.

We also conducted a user study using live data from Twitter. All measurements were performed on a server with two quad-core 2.4 GHz Intel Xeon processors, 48 GB of RAM, and a 2 TB RAID-5 disk.

Datasets

Blog dataset: We have obtained the ISWCM Spinn3r blog dataset [10] consisting of 44 million blog posts created in the time period from August 1st to October 1st, 2008. Each blog has a set of categories assigned, which we use as tags. Examples of tags are *Election 2008* and *Economics*, or *Entertainment* and *Sports*. We use the blog posts from September 2008.

Twitter dataset: We have access to the "fire hose" stream of Twitter, delivering 10% of all Tweets (in general, all status updates). Tweets contain so called hash-tags, such as #egypt and #revolution. We use the tweets from 02.07.2011 to 15.07.2011 in our study. The user study, however is performed on live data received from the Twitter stream, described below.

For both datasets the *entity tagging* is done on-the-fly, i.e., as documents are streaming in.

Measures of Interest

For each evaluation phase, i.e., whenever a ranking of emergent topics is computed, we obtain the following measures:

Precision: An important aspect for user satisfaction is the precision of the emergent topics that our methods suggest to the user. This measure is used only in the user study on live Twitter data. We report on the average precision@k *precision at k* value, that is the fraction of topics marked by the users to be interesting divided by k. We also include the results for the NDCG@k [17] values (normalized discounted cumulative gain). For both measures, we compute the *statistical significance* of the reported values using *Fisher's randomization test* and the paired t-test [8, 29].

Runtime: We report on runtime cost of our methods for tag-pair tracking and shift detection. This captures the average time spent at each evaluation phase. It does not include pre-processing costs like named-entity tagging but it includes the post-processing cost of diversification.

Relative Accuracy: When we run enBlogue using a specific amount of seed tags, the resulting emergent topics are only approximated. In this measure we compute the relative accuracy of the algorithm running compared to the base-line using all tags as seed tags.

Algorithms

We compare the following algorithms:

enBlogue: This is our approach for emergent topic detection, described in this paper. For the naming of the different configurations, we will refer to our algorithms by mentioning the number of seed tags used, for example our algorithm using all tags as seeds will be referred as enBlogue-100%, our algorithm using 20% of the tags as seeds will be referred

```
FIND BURSTY TAGS
1  find all tags in the current window
2  for each tag in the old tags
3    if found in current window ...
         ... and exists in more than min documents
4      add its frequency as current frequency of the tag
5      predict current frequency of tag based on history
6      score tag based on predicted and actual frequency
7    else
8      add zero as current frequency of the tag
9      predict current frequency of tag based on history
10     score the tag based on its predicted and actual frequency
11 for each tag found in this evaluation and not existing before
12   if it exists in more than min documents
13     add its frequency as the current frequency of the tag
14     add zeros as the previous frequencies of the tag
15     predict the current frequency of the tag based on history
16     score the tag based on its predicted and its actual frequency
17   else
18     ignore the tag
19 remove all tags that have not been found ...
         ... for $\varrho$ evaluations
20 consider all tags with positive score as bursty tags

GROUP BURSTY TAGS
1  for each tag found bursty by the previous algorithm
2      create the sets of documents that contain it
3  find the related pairs of bursty tags
4  consider an undirected graph g where every node is a tag.
5  for each related pair of bursty tags
6      add an edge in the graph
7  find the connected components of the graph
8  consider each connected component to be an event
9  score each event
10 sort events according to scores
11 return the k top events
```

Figure 5: The two main algorithms used in our implementation of TwitterMonitor.

as enBlogue-20% and so on.

TwitterMonitor (TM): This is our competitor ,the approach by Mathioudakis and Koudas [26], as described in Section 6.

For both algorithms, we study the influence of the number ϱ of data points we consider from history (past) for the trend detection. The number of result topics k has no influence on the runtime of both implementations, we will hence not report on any variation of k and set it to value 20. The window sizes are also set to specific values, $W = 1h$ in case of Twitter and $W = 6h$ in case of the Blog dataset. We do study the effect of the number of tags and documents on the algorithms' performance, by grouping the observed runtime values by the number of tags and documents, respectively.

7.1 User Study

We conducted a user study where we employ emergent topic detection with enBlogue and the TM on live Twitter data. We did not see a viable way to perform such a study on offline (i.e., months or years old) data as it turned out to be almost impossible for users to go mentally back in time to check if an detected event is indeed noteworthy. In particular for events that are of scale smaller than big occasions like the Olympic games, huge hurricanes, US elections, and so on. The events we derive from the live Twitter data are most of the time much smaller, but nevertheless, a lot of them are interesting and worth being shown on the Website. For instance, on 16.09.2011 12:00 GMT enBlogue detected events such as "Assad Syria" and "Lybia Niger Gadhafi". We have set up a website showing results for the two competing algorithms in an anonymized form. We asked colleagues to participate in the study which had the following task description: Every now and then, check the results published on our website and try to identify interesting emergent topics. Those topics that are deemed interesting are supposed to be marked using an HTML checkbox. Since tags alone are sometimes hard to map to a real world event, by clicking on the tags, users are able to see sample Tweets containing the emergent topic's tags. For instance, at the same day as above, we found the tag pair "dolphin Australia" (apparently there was a new species discovered), which is hard to make sense of without looking at additional information in form of sample tweets. Interestingly, and as a support of the whole approach, for a lot of events we discovered there were no media information immediately available, only some minutes/hours later in the breaking news section.

In the period of the user study, the last two weeks in September, we recorded 80 non-redundant evaluations. Non-redundant means that we eliminated duplicate submissions, identified by IP address and the timestamp of the ranking to be evaluated.

Figure 6 reports on the precision at k values for enBlogue and TM. Users were asked to select noteworthy events out of 20 events per algorithm. A precision value of x at 20 means that the fraction of x events have been considered noteworthy in a ranking of 20. We observe that enBlogue clearly outperforms TM. In average, enBlogue has identified 2.5 out of 20 noteworthy events per hour (for the time points considered by the users in our evaluation study). This is in contrast to often not even one noteworthy event (on average 0.8 out of 20 reported ones) delivered by TM. Note that the 20 reported events were not filtered (except for a simple keyword filter aiming at eliminating porn related Tweets).

For completeness we have also calculated the NDCG [17] values, reported in Table2(right). We have computed the paired t-test and Fisher's randomized significance test over both the precision@20 and the NDCG@20 values: The randomized test computed for 100 000 permutations reported a p-value of 0 for both the precision and the NDCG values. The paired t-test reported a p-value of $3.5 * 10^{-13}$ for ndcg@20 and $6.5 * 10^{-25}$ for precision@20.

Table 3 shows sample results of the events detected on three consecutive days (28th, 29th, 30th) in September 2011 with enBlogue. As we can see, enBlogue discovered quite many interesting results at those days. Including the alliance plans between Microsoft and Samsung, the killing of an Anwar al-Awlaki, a member of Al-Qaeda, by US military forces, the case of Michael Jackson's personal physician Conrad Murray, the Hollywood actor Sean Penn visiting the Tahrir place in Cairo, Egypt, and the scandal of Manchester City's Carlos Tevez, refusing the exchange during a game in the European Soccer Champions League.

7.2 Runtime

In Table 4 and Table 5 we can see how the runtime varies for the five versions of enBlogue and TM for different number of tags and documents. The runtime depends on three factors:

28.09.2011	29.09.2011	30.09.2011
lfc, liverpool, ynwa	redsox, shocked, stunned, seasonover	in america, occupywallstreet
orioles, red sox	nadarkhani, iran, yousef	redsox, terry francona
dana, danafacts	enoughisenough, occupysf, occupywallstreet	motegi, motogp
europe, soteu	carlos tevez, manchester city	derby, liverpool
intel, samsung	ownacolour, unicef	bahrain, u.s.
detroit, tedx	bahrain, egypt, usa	anonymous, antisec
bahrain, twitition, u.s. ambassador	conrad murray, michael jackson	rugby, samoa, southafrica
fifa, tevezexcuses	fact, healthcare reform	arsenal, spurs
bieberfacts, justin bieber	bologna, occupywallstreet, ows	israel, awlaki, alqaeda, yemen
microsoft, samsung	bahrain, syria	egypt, noscaf
messi, fcblive, mascherano, barca, puyol, abidal, xavi	real madrid, kaka, realmadrid	assad, syria
arshavin, rosicky, sagna	bologna, occupywallstreet	sean penn, tahrir
anelka, cfc, ivanovic, kalou, drogba, romeu	nationalcoffeeday, peetscoffee	awlaki, obama
nadarkhani, iran, irani, yousef	celtic, udinese	manutd, mufc
		libertysquare, armenia, opposition, rally, yerevan
		boston, terry francona

Table 3: Sample of the events detected in enBlogue and marked as relevant by at least one of the user study participants.

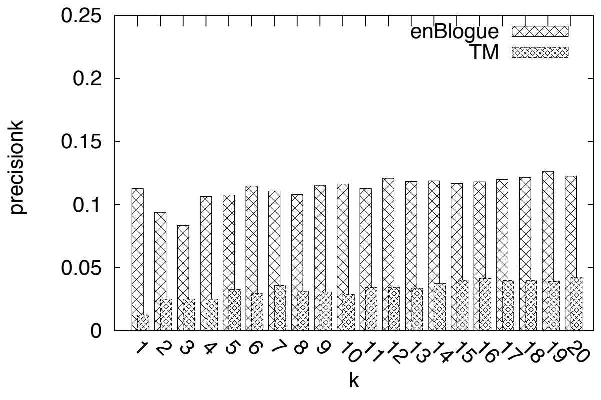

Figure 6: Precision@k values

	Precision@k			NDCG@k	
k	enBlogue	TM	*k*	enBlogue	TM
1	0.112	0.012	1	0.112	0.012
2	0.094	0.025	2	0.094	0.025
3	0.083	0.025	3	0.089	0.026
4	0.106	0.025	4	0.115	0.028
5	0.108	0.032	5	0.123	0.033
6	0.115	0.029	6	0.137	0.033
7	0.111	0.036	7	0.151	0.041
8	0.108	0.031	8	0.160	0.041
9	0.115	0.031	9	0.177	0.044
10	0.116	0.029	10	0.191	0.045
11	0.112	0.034	11	0.198	0.053
12	0.121	0.034	12	0.220	0.056
13	0.118	0.034	13	0.229	0.059
14	0.119	0.038	14	0.239	0.068
15	0.117	0.04	15	0.246	0.073
16	0.118	0.041	16	0.258	0.077
17	0.120	0.040	17	0.270	0.078
18	0.122	0.040	18	0.285	0.081
19	0.126	0.039	19	0.3	0.083
20	0.122	0.042	20	0.304	0.09

Table 2: Precision@k results (left) and NDCG@k results (right) achieved in the user study by the competing algorithms.

initial pairs: We call initial pairs the pairs created by matching all tags with all seeds tags, for enBlogue, all bursty tags with all bursty tags, for TM. The initial pairs are checked in order to find he pairs of related tags, called from now on related pairs.
new related pairs: We call new related pairs the related pairs found during the current evaluation. All new related pairs are checked during the shift detection procedure (Section 4.3) and during the scoring procedure (Section 4.4). The number of new related pairs depends on the number of initial pairs
old related pairs: We call old related pairs the related pairs found during previous evaluations, but not during the current evaluation, that are still stored and processed during the scoring procedure (Section 4.4). The number of old related pairs depends on the number of the new related pairs of the previous evaluations.

Given a number of tags, the number of initial pairs is proportional to the percentage of seeds used. This can be verified by the results shown in the rows of Table 4 and Table 5, where we can se that, for every group of tags, the runtime increases while the percentage of seeds increase. Since the new related pairs depend on the initial pairs, in the same tables, we can also see that the related pairs increase as the percentage of seeds increases.

Given a percentage of seeds, the number of initial pairs is proportional to the number of tags. This can be verified by the results shown in the columns of Table 4 and Table 5, where we can see that the average runtime increases while the number of tags increases. The same is also true for the new related pairs.

For TM there is no direct analogy between the number of tags and the number of bursty tags. However it is expected the number of bursty tags to increase as the number of tags increase, which leads to an increase in the initial pairs. The experiments showed that TM had, on average, 60% of the maximum number of initial pairs. The maximum number of initial pairs is defined to be the number of pairs produced when all tags are used as seeds. By taking into account only the number of initial pairs, TM should have been slower than

	Twitter		Blog	
	2	4	2	4
enBlogue-10%	2.61	3.34	0.83	2.65
enBlogue-20%	3.11	4.02	1.09	3.50
enBlogue-40%	3.54	4.68	1.66	4.94
enBlogue-70%	4.67	6.02	1.85	6.22
enBlogue-100%	6.05	6.57	2.17	6.67
TM	2.44	3.99	0.54	0.96

Table 6: Average Runtime in seconds per past values used for the prediction for Twitter and Blog datasets

	Twitter		Blog	
	Runtime	Accuracy	Runtime	Accuracy
enBlogue-10%	3.34	0.14	2.65	0.13
enBlogue-20%	4.02	0.23	3.50	0.18
enBlogue-40%	4.68	0.37	4.94	0.29
enBlogue-70%	6.02	0.60	6.22	0.53
enBlogue-100%	6.57	1.00	6.67	1.00

Table 7: Average Runtime in seconds and Relative Average Accuracy for Twitter and Blog datasets

enBlogue-10% and enBlogue-20% in any case. However, this is not true because, as can be verified form Table 4 and Table 5, there are groups of tags where TM has to process less new related pairs than enBlogue-20%. Moreover, in the TM algorithm the notion of old related pairs does not exist, since only the new related pairs are taken under consideration during the scoring procedure. So, there are cases where TM has been found to be faster even from enBlogue-10%.

Since an increase in the number of documents causes an increase in the number of tags, the results for different number of documents are similar to those for different number of tags. However, an increase in the number of documents affects the runtime in one more way. It causes an increase in the size of document-sets associated with each tag. This results in greater runtimes when comparisons between sets are performed. Such comparisons are needed to check the initial pairs and when we compute the correlations for the new related pairs.

In Table 6 we can see how the number ϱ of past values stored affects the average runtime for the five versions of enBlogue and TM for the twitter (left) and the blog (right) datasets. An increase in ϱ causes an increase in the number of the old related pairs, which affects the runtime of enBlogue versions. It also causes an increase in the number of tags that are found to be bursty, which affects the runtime of TM. Since the characteristics of the two datasets are very different we can see that the average runtimes are affected in different ways on the two datasets. In the Twitter dataset an increase in ϱ affects more the number of tags found bursty and less the number of old related pairs. This causes a rapid increase in runtime for TM and smaller increases in the runtimes of the enBlogue versions. In the Blog dataset an increase in ϱ affects more the number of old related pairs and less the number of tags found to be bursty. This causes a rapid increase in the runtime of the five enBlogue versions and a smaller increase in the runtime of TM.

7.3 Runtime and Relative Accuracy

In Table 7 we can see how the average runtime and the average relative accuracy are affected by the percentage of seeds. From this table it is obvious that a decrease in the percentage of seeds does not cause the same percentage of decrease in the average runtime. This is because the percentage of seeds affects by the same percentage the number of initial pairs but not the number of new and old related pairs.

In Table 7 we can also see that a small decrease in the percentage of seeds causes a big decrease in the relative accuracy. This is due to the fact that by using a smaller percentage of seed-tags there are some related pairs that cannot be found in the results (the pairs that do not have at least one of the selected seed-tags). This lack in related pairs affects also the groups of tags, created during the diversification procedure, and is responsible for the reduced relative accuracy.

What is worth mentioning is that it is not clear whether the results with 100% seeds are more of user interest or not. Since we have chosen the seeds to be the most popular tags there is the possibility that the results using a lower percentage of seeds are more interesting than the results with the 100% seeds. However the percentage of seeds might be of importance since with a small number of seeds there is the danger of having low diversity in the results.

8. CONCLUSION

This paper addressed the problem of information overload that arises with the advent of Web 2.0 streams by introducing an approach, coined enBlogue, that automatically discovers emergent topics based on tagged postings. enBlogue incorporates techniques to analyze shifts in the correlations of tags, as an indicator of the onset of a newly emerging topic. We evaluated enBlogue by performing a user study based on live Twitter data, where users were asked to select those reported events they considered to be noteworthy. This showed that enBlogue achieves, on average, a 3 times higher result quality compared to the state-of-the-art approach, at the cost of a slightly increased runtime, which is well below 15 seconds for a real-world workload of 10% of all Twitter data on an hourly basis. Thus, we believe that enBlogue is well positioned to handle highly demanding real world workloads, such as Twitter, which receives more than 100 million tweets per day. In addition to the development of the concepts and algorithms necessary to discover emergent topics, we have built a system that incorporates them and is available on the web at `http://blogue.mmci.uni-saarland.de` showing results based on live Twitter data.

9. REFERENCES

[1] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR*, pages 37–45, 1998.

[2] F. Alvanaki, S. Michel, K. Ramamritham, and G. Weikum. Enblogue - emergent topic detection in web 2.0 streams. In *SIGMOD Conference*, 2011. `http://qid3.mmci.uni-saarland.de/sigmod2011.pdf`.

[3] APE – Ajax Push Engine. `http://www.ape-project.org/`.

[4] R. Balasubramanyan, F. Lin, W. W. Cohen, M. Hurst, and N. A. Smith. From episodes to sagas: Understanding the news by identifying temporally related story sequences. In *ICWSM*, 2009.

		enBlogue - seeds percentage										TM	
		10		20		40		70		100			
		Time	Pairs	Time	Pairs	Time	Pairs	Time	Pairs	Time	Pairs	Time	Pairs
Tags	2000-2999	1.54	1307	1.93	2007	2.39	2873	3.02	3668	3.41	4223	2.63	2403
	3000-3999	2.56	1910	3.63	2837	3.77	3970	4.95	5002	5.55	5710	3.40	2933
	4000-4999	4.02	2566	4.75	3710	5.69	5126	7.08	6408	7.87	7286	4.73	3453
	5000-5999	6.53	3564	6.83	4978	8.15	6640	11.14	8155	11.14	9222	5.76	4160
Documents	15000-29999	1.26	1099	1.61	1740	2.04	2537	2.47	3308	2.87	3838	1.79	2223
	30000-44999	1.80	1583	2.25	2379	2.75	3352	3.73	4221	3.85	4824	2.85	2667
	45000-59999	2.95	2135	3.99	3148	4.36	4434	5.67	5601	6.36	6390	4.16	3135
	60000-74999	4.76	2919	5.51	4155	6.60	5616	8.48	6953	9.19	7894	5.00	3687
	75000-89999	11.78	3862	10.37	5310	11.78	7031	13.50	8574	14.81	9624	6.80	4084

Table 4: Average Runtime in seconds and New Related Pairs for Twitter dataset

		enBlogue - seeds percentage										TM	
		10		20		40		70		100			
		Time	Pairs	Time	Pairs	Time	Pairs	Time	Pairs	Time	Pairs	Time	Pairs
Tags	0-999	1.72	2171	2.36	3340	3.27	5076	4.03	6780	4.61	8052	0.54	5282
	1000-1999	2.70	5627	3.52	8248	4.75	11737	6.37	15165	6.71	17656	0.92	9731
	2000-2999	3.25	11581	4.48	16673	8.01	23002	7.49	29121	8.47	33396	1.40	19761
	3000-3999	4.32	20941	5.12	29445	7.00	40169	8.51	49781	9.73	56511	3.03	27975
Documents	0-4999	2.33	3447	2.84	5207	3.89	7614	5.54	10023	5.53	11836	0.70	6705
	5000-9999	2.90	8066	4.07	11668	5.88	16318	6.78	20838	7.66	24034	1.14	13592
	10000-14999	3.54	19193	4.57	27188	6.31	37509	7.55	46623	8.64	53246	2.47	25073
	15000-19999	5.10	22689	5.67	31701	7.69	42828	9.47	52938	10.82	59776	3.58	30877

Table 5: Average Runtime in seconds and New Related Pairs for Blog dataset

[5] N. Bansal and N. Koudas. Blogscope: A system for online analysis of high volume text streams. In *VLDB*, pages 1410–1413, 2007.

[6] H. Becker, M. Naaman, and L. Gravano. Event identification in social media. In *WebDB*, 2009.

[7] H. Becker, M. Naaman, and L. Gravano. Learning similarity metrics for event identification in social media. In *WSDM*, pages 291–300, 2010.

[8] G. E. P. Box, W. G. Hunte, and J. S. Hunter. *Statistics for experimenters: an introduction to design, data analysis, and model building.* John Wiley & Sons, 1978.

[9] C. Budak, D. Agrawal, and A. E. Abbadi. Structural trend analysis for online social networks. *PVLDB*, 4(10):646–656, 2011.

[10] K. Burton, A. Java, and I. Soboroff. The ICWSM 2009 Spinn3r dataset. In *ICWSM*, 2009.

[11] M. Cataldi, L. Di Caro, and C. Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pages 4:1–4:10, New York, NY, USA, 2010. ACM.

[12] G. Cormode and M. Hadjieleftheriou. Finding the frequent items in streams of data. *Commun. ACM*, 52(10):97–105, 2009.

[13] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. *TWEB*, 1(2), 2007.

[14] M. N. Garofalakis. Distributed data streams. In *Encyclopedia of Database Systems*, pages 883–890. 2009.

[15] M. N. Grinev, M. P. Grineva, A. Boldakov, L. Novak, A. Syssoev, and D. Lizorkin. Sifting micro-blogging stream for events of user interest. In *SIGIR*, page 837, 2009.

[16] J. Han and B. Ding. Stream mining. In *Encyclopedia of Database Systems*, pages 2831–2834. 2009.

[17] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[18] N. Kawamae. Trend analysis model: trend consists of temporal words, topics, and timestamps. In *WSDM*, pages 317–326, 2011.

[19] S. Khy, Y. Ishikawa, and H. Kitagawa. Novelty-based incremental document clustering for on-line documents. In *ICDE Workshops*, page 40, 2006.

[20] G. Kumaran and J. Allan. Text classification and named entities for new event detection. In *SIGIR*, pages 297–304, 2004.

[21] S. Lallich, O. Teytaud, and E. Prudhomme. Association rule interestingness: Measure and statistical validation. In *Quality Measures in Data Mining*, pages 251–275. 2007.

[22] J. Leskovec, L. Backstrom, and J. M. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, pages 497–506, 2009.

[23] Z. Li, B. Wang, M. Li, and W.-Y. Ma. A probabilistic model for retrospective news event detection. In *SIGIR*, pages 106–113, 2005.

[24] J. Ma and S. Perkins. Online novelty detection on temporal sequences. In *KDD*, pages 613–618, 2003.

[25] C. D. Manning, P. Raghavan, and H. SchÃijtze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[26] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *SIGMOD Conference*, pages 1155–1158, 2010.

[27] A. D. Sarma, A. Jain, and C. Yu. Dynamic relationship and event discovery. In *WSDM*, pages

207–216, 2011.

[28] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *ICWSM*, 2009.

[29] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, pages 623–632, 2007.

[30] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *KDD*, pages 32–41, 2002.

[31] Y. Yang, T. Pierce, and J. G. Carbonell. A study of retrospective and on-line event detection. In *SIGIR*, pages 28–36, 1998.

[32] Q. Zhao and P. Mitra. Event detection and visualization for social text streams. In *ICWSM*, 2007.

[33] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.