Event-based Classification of Social Media Streams

Timo Reuter
CITEC, Universität Bielefeld
Universitätsstr. 21–23
33615 Bielefeld
treuter@cit-ec.uni-bielefeld.de

Philipp Cimiano
CITEC, Universität Bielefeld
Universitätsstr. 21–23
33615 Bielefeld, Germany
cimiano@cit-ec.uni-bielefeld.de

ABSTRACT

Events play a prominent role in our lives, such that many social media documents describe or are related to some event. Organizing social media documents with respect to events thus seems a promising approach to better manage and organize the ever-increasing amount of content in social media applications. A challenge is to automatize this process so that incoming documents can be assigned to their corresponding event without any user intervention. We present a system that is able to classify a stream of social media data into a growing and evolving set of events. By doing this, we successfully address two key problems that arise in this context: i) scaling to the data sizes and rates encountered in social media applications, and ii) tackling the new event detection problem, i.e. the problem of determining whether an incoming data item belongs to a new or a known event. We successfully address these problems by i) including a candidate retrieval step that retrieves a set of event candidates that the incoming data point is likely to belong to and ii) by including a function trained using machine learning techniques to determine whether the incoming data item belongs to the top scoring candidate or rather to a new event. We show that our system addresses the above mentioned challenging issues successfully and that it outperforms other state-of-the-art approaches in terms of quality and scalability.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; D.2.8 [**Software Engineering**]: Metrics—*Complexity Measures, Performance Measures*

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Clustering, Classification, Event Identification, SVMs

ICMR '12, June 5-8, Hong Kong, China

1. INTRODUCTION

Social Media Applications are proliferating and they are characterized by an ever-increasing amount of content that represents a never-ending data stream growing at high rates.

As events play a prominent role in our lives, many of the documents uploaded to social media sites are related to some event. Classifying social media documents by the events they represent or are related to is thus a promising approach to better manage and organize the ever-increasing amount of user-generated content in social media applications.

Many social media sites support tagging to better organize the content and thus facilitate the search for specific data items. Such tags can also be used to organize data according to the events they belong to. For this purpose, *last.fm*[1] hosts an event database with unique identifiers in the form of so called *"machine tags"* that can be used to tag pictures – for example when uploading them to Flickr – thus assigning them to one or more uniquely identified events. While some users exploit such tags, the majority of pictures on Flickr are not assigned to events via such machine tags.

Given the above explanations, it seems clear there is an impending need for automatic techniques that perform the assignment of a social media item (newly uploaded to some social media site) to its corresponding event (if it already exists) or creates a new event to which future data items can be assigned to. In line with Becker et al. we refer to this problem as the *event identification problem* [4].

From the perspective of the organizations hosting a social media application, the uploaded postings, messages, pictures etc. can be seen as an exponentially growing and never-ending stream of data. When considering the task of *event identification*, we are thus faced with the challenge of classifying a massive and never-ending stream of data into their corresponding events. There are at least three challenges involved in developing a system that can classify data streams as originating in social media applications into event categories. First, the challenge is to deal with the massive amount of data arriving per minute. Second, the challenge is to classify data into potentially millions of events. Thirdly, we need to deal with the fact that the set of events that we assign data items to is constantly growing. This is a problem that has been referred to as ''concept drift" [17, 35]. Due to the growing nature of the target categories, standard supervised learning techniques which assume that the target categories are fixed are less suited to the task.

In this article we present a system for the classification of social media stream data into corresponding events that ad-

[1] http://last.fm

dresses the above mentioned three challenges. We frame the problem as a stream data classification problem. Each new data item uploaded to the system is either classified into one of the existing events or a new event is created in the system. In particular, we provide the following contributions:

- We present a complete system that classifies an incoming set of documents into the events they describe or are related to, creating a new event if necessary. We experimentally demonstrate that our system successfully addresses the above mentioned challenges.
- We show in particular that the problem of assigning a new data item to its corresponding event can be modeled successfully as a ranking problem where a preselected number of events is ranked according to how likely the new data point belongs to. By choosing the top-ranked event, we show that the accuracy of assigning a new data item to its corresponding event is close to 100%. We use a function learned using machine learning techniques and discuss the impact of the different features on this decision.
- We further show that the decision whether the new data item belongs to one of the existing events or a new event can also be taken with reasonable accuracy (85.9%) and also discuss a subset of optimal features that lead to this result.
- We show that the system can indeed scale to large numbers of events by using an appropriate candidate retrieval step which retrieves a set of event candidates that the incoming data point is likely to belong to. With this step we avoid scanning all the events in the database and thus allow our system to scale to very large numbers of events. We show in particular that retrieving around 300 candidates yields a reasonable trade-off between scalability and clustering quality.
- As baseline, we compare to the state-of-the-art system of Becker et al. [4] with respect to quality and efficiency, showing that our systems outperforms this baseline by 15% points with respect to F-Measure, having in particular a much higher precision (80% vs. 46%). Our system is also much more efficient than the one of Becker et al., showing a nearly constant processing time independently of how many documents the system has processed so far, a requirement for scaling to much larger numbers of documents. We show that these two improvements in terms of quality and scalability are due to the candidate retrieval step that we rely on (see above).

The article is structured as follows: in Section 2 we present our overall approach and discuss its single components. In Section 3 we describe how we created our dataset consisting of 1 million Flickr pictures together with an appropriate gold standard developed using last.fm machine tags. We also describe how the machine learning components were trained and provide experimental results. Before concluding, we discuss related work on event identification in social media, stream-based classification and outlier detection.

2. SYSTEM DESCRIPTION

Our system processes an incoming stream of documents D as follows. For each incoming document $d \in D$:

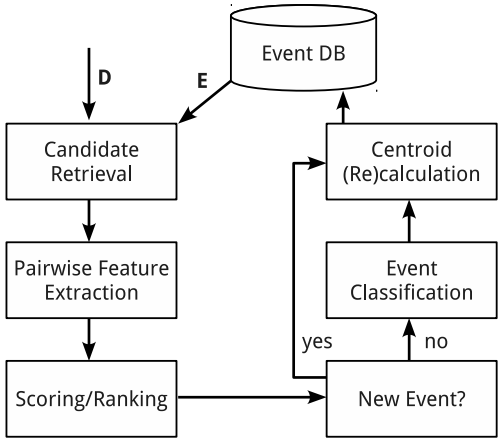

Figure 1: Overview of the event identification system

1. A set E of k events that d is likely to belong to are retrieved from the event database (*candidate retrieval)*
2. For each of these candidates $e \in E$, the probability that d belongs to e, $P(e|d)$, is computed, and all candidates are ranked according to this probability (*scoring and ranking*). Let e_{max} be the top scoring candidate
3. Given this ranked list of candidates, the probability that d belongs to a new event, $P_{new}(d)$, or that it belongs to the first event in the list, $P_{\text{belongs\_to\_top\_cand}}(d)$, is computed. We assume that these are the only two options, i.e. $P_{new}(d) + P_{\text{belongs\_to\_top\_cand}}(d) = 1$.
4. If $P_{new}(e) > \theta_n$, a new event e' is created and d is assigned to this newly created event; $\vec{e'} := \vec{d}$ (*new event detection*).
5. Otherwise, d is assigned to e_{max}; the centroid $\vec{e}_{max}$ is recomputed.

Algorithm 1 Stream-based classification into events

```
for all d ∈ D do
  Top_k(d) = retrieve a ranked list of promising event candidates to which d could belong
  for all e ∈ Top_k(d) do
    compute P(e|d) – the prob. that d belongs to e
  end for
  e_max = max_{e'∈Top_k(d)} P(e'|d)
  compute P_new(d) – the prob. that d belongs to a new event
  if P_new(d) > θ_n then
    create a new event e'
    e' = {d}
    e'⃗ = d⃗
  else
    e_max = e_max ∪ {d}
    recompute e⃗_max
  end if
end for
```

The above procedure is illustrated by the pseudocode in Algorithm 1 and depicted graphically in Figure 1. We com-

ment on some crucial aspects of our approach in what follows, pointing to the relevant sections where the corresponding components are described in more detail:

- The candidate retrieval step relies on a set of SQL queries that build on appropriate inverted indices (for text data like tags) as well as on timestamps to retrieve a set of promising candidates. This step is crucial to keep the approach scalable and to avoid having to process all events stored in the database. This step is described in more detail in Section 2.1.
- As feature representation, we compute a number of similarities between a document d and each event e from the list of top-k retrieved events. This is described in more detail in Section 2.2.
- The above mentioned probabilities $P(e|d)$ and $P_{new}(d)$ are computed using support vector machines trained on the basis of an appropriate training dataset. Below, we describe in more detail how these SVMs are trained and how the probabilities are computed. Each of these SVM models is trained on a separate split of training data. This is described in more detail in Section 2.3.
- An important parameter is the hyperparameter θ_n, which essentially determines whether a new event is created or the new data item is assigned to an existing event in the database. This hyperparameter is tuned on a separate training data split.

The different steps mentioned above are described in detail in the following sections.

2.1 Candidate Retrieval

In order to make our approach scalable, we implement a candidate retrieval step that retrieves a set of k promising events in the database that d could belong to. Essentially, this corresponds to a *blocker* or *candidate generator* used in record linkage approaches [27].

We implement this candidate retrieval by issuing 6 queries to the database, retrieving the nearest events by capture time (200), by upload time (50), by geographic location (20), by similarity of tags (20), by similarity of title (20), and by similarity of description (20).

With these queries, we retrieve a total of $k = 330$ events that are further processed by our system.

2.2 Feature Extraction

A pair of a document and a candidate event is described in terms of a vector of nine features that describe the match between the document and the event. In particular, we use nine similarity measures which exploit the following information sources:

- **Temporal information:** We rely on the timestamp when the document was created (e.g. when the picture was taken) in order to define a time-based similarity measure as:

$$sim_{time}(d, e) = 1 - \frac{\log(|time(d) - time(e)|)}{y}$$

where $time(d)$ and $time(e)$ are timestamps denoting the number of minutes elapsed since the Unix epoch, and y is the logarithm of the number of minutes in a year. This yields two similarity measures: one calculated on the basis of the capture time ($sim_{capture}(d, e)$) and one calculated on the basis of the upload time ($sim_{upload}(d, e)$).

- **Geographical information:** We use the latitude and longitude indicating the place where the document was created to determine the great-circle distance between two points using the Harversine-formula as in the approach by Reuter et al. [29].
- **Textual information:** contains tags, a title, a description, etc. We describe the textual content by way of TF-IDF vectors. To determine the similarity we rely on the cosine similarity ($sim^{cos}_{text}(e, d)$) as well as the BM25 formula [30] ($sim^{BM25}_{text}(d, e)$).

Overall, a vector describing the similarity between a pair of document and event looks as follows:

$$\vec{sim}(d, e) = \begin{pmatrix} sim_{capture}(d, e) \\ sim_{upload}(d, e) \\ sim_{geo}(d, e) \\ sim^{cos}_{tags}(d, e) \\ sim^{BM25}_{tags}(d, e) \\ sim^{cos}_{title}(d, e) \\ sim^{BM25}_{title}(d, e) \\ sim^{cos}_{description}(d, e) \\ sim^{BM25}_{description}(d, e) \end{pmatrix}$$

2.3 Scoring and Ranking

For each incoming document d, we compute the likelihood that it belongs to a given event e, $P(e|d)$, and order the events retrieved by the candidate retrieval step by decreasing probability. The likelihood that document d belongs to event e is calculated using a support vector machine as follows. We train a SVM that relies on an appropriate training dataset to discriminate between pairs of document and corresponding event (positive examples) and pairs of documents that do not belong to the given event (negative examples). We use 4,000 examples for each of these classes to train binary SVMs and compute the probability $P(e|d)$ as the probability that the pair (d, e) – described by the above mentioned vector of similarities – belongs to the positive class, i.e. $P(e|d) = P(positive|\vec{sim}(e, d))$. We use the C-SVMs implemented in libsvm, which uses Platt's algorithm[24] to compute the above probability as follows:

$$P(positive|\vec{sim}(e, d)) := \frac{1}{1 + exp(A\langle\vec{e}, \vec{d}\rangle + B}$$

where the parameters A and B optimized by the SVM by minimizing the negative log likelihood of the training data.

2.4 Event Detection and Classification

Given the candidates ranked by the likelihood that d belongs to them, an important question is whether the document d belongs to the top ranked event or rather to a new event to be created. This is what we refer to as *new event detection problem*. The top scoring candidate might actually represent an event that d is not related to, such that we need a decision function that decides whether to assign the document d to the top scoring candidate or to a newly created event.

For this purpose, we also employ a C-SVM trained on an appropriate dataset consisting of examples in which the document belongs to the top-scoring event (positive examples) and examples in which the document belongs to a new event (negative examples). As features for this task we use the following:

- **max** (max): $P(e_1|d)$, i.e. the prob. that d belongs to the top-scoring event
- **min** (min): the probability $P(e_{10}|d)$ – the probability that d belongs to the 10-th ranked event
- **average** (avg): $\frac{1}{10}\sum_{i=1}^{10} P(e_i|d)$ – the average probability of the top-10 most likely events
- **standard deviation** (stddev): the standard deviation of the top-10 most likely events
- **maximum capture time** (maxtst): $sim_{capture}(d, e_1)$
- **maximum upload time** (maxtsu): $sim_{upload}(d, e_1)$

For each document d this yields a feature vector $\vec{new}(d)$ that is used to classify d into two classes: belongs to top scoring event (positive) or belongs to a new event (negative). The SVM classifier is trained using an equal number of positive and negative examples and as in the above case returns a probability that the document belongs to a new event: $P_{new}(d) = P(negative|\vec{new}(d))$.

We then use a threshold on this probability as a hyperparameter to be tuned that decides about whether the event is assigned to a new event or assigned to the top ranked event. The optimal threshold is determined empirically using a gradient descent technique on a split of our training data. We report on the performance of this new event detection decision model in Section 3.1.3 describing our experiments.

3. EXPERIMENTAL SETUP & RESULTS

After having presented our system, we now describe how our dataset consisting of 1 million pictures from Flickr has been created as well as our experimental results.

3.1 Experimental Settings

3.1.1 Dataset Creation

Since 2007, *last.fm* provides a freely available event database in which every event contained has a unique event ID. A key function of Flickr is the possibility for the user to assign so-called *machine tags* to a picture. The main difference to normal *tags* is that machine tags follow a fixed schema. Such a machine tag might have the following form: `lastfm:event=#eventid`; this provides us the following information about the picture: a) the picture belongs to an event contained in the last.fm database and b) the corresponding event on last.fm has the ID *#eventid*. Therefore, this allows us to assume that pictures marked with the same machine tag on Flickr belong to the same event. We thus constructed our gold standard by downloading pictures with last.fm machine tags from Flickr using their API, grouping them into events using the event IDs.

We considered pictures with a capture time between January 2006 and October 2011, yielding a dataset of 1 million pictures assigned to 36,782 events. We divided this dataset into 10 splits consisting of 100,000 pictures each, in temporal order. Only 31.3% of the documents have a geo tag assigned, 90.2% have at least one tag, 97.9% have a title, and 45.9% have a description assigned. Machine tags were only used to create the gold standard and were no longer part of the dataset to conduct our experiments on. Splits 1-3 have been used to train the machine learning components and to optimize parameters. We use splits 4-10 to test the system, reporting average performance of our system with respect to precision, recall and F-Measure (defined below).

3.1.2 Baseline and Evaluation Measures

As a baseline we use the CLASS-SVM method described in Becker et al. [4]. Becker et al. present an incremental clustering approach also making use of an SVM as in our approach to find the most likely event that the document belongs to. There are two crucial differences to our approach. First, Becker et al. do not use a second decision model to detect new events, but a simple threshold. Second, Becker et al. do not use a candidate retrieval or blocking step and thus have to scan all the events in the database for each incoming document. Their approach can thus not scale to larger datasets.

We report our results in terms of Precision, Recall and F-Measure which are defined as in Becker et al. [4].

$$P_b = \sum_{d \in D} \frac{1}{|D|} \frac{|Cluster(d) \cap GoldStandard(d)|}{|Cluster(d)|}$$

$$R_b = \sum_{d \in D} \frac{1}{|D|} \frac{|Cluster(d) \cap GoldStandard(d)|}{|GoldStandard(d)|}$$

$$F_1\text{-Measure} = 2 \cdot \frac{P_b \cdot R_b}{P_b + R_b}$$

We reimplemented the approach of Becker et al. and tested it under the same conditions and on the same dataset as our approach.

3.1.3 Training

Decision for Assignment of Data Items to Event.

To create our training examples for the correct event assignment SVM model, we follow Reuter et al. [28] and create a balanced training set with 8,000 samples from split 1. As proposed we use the *nearest* sampling strategy where we choose 4,000 consecutive documents ordered by time as positive examples. The pairs (d,e) selected as negative examples are such that d is temporally very close to e, but does not belong to it. This strategy for creating negative examples was proved effective by Reuter et al. [28]. In order to train this SVM, we use a C-SVM with a linear kernel $\Phi(d, e) = \langle d, e \rangle$. The hyperparameter C denoting the trade-off between the training error and margin is set to 1.0. In addition, we extend the SVMs so that they can handle missing features instead of assuming that the similarity for a missing feature (e.g. geo tag) is 0.0, thus not being able to distinguish between the case where the feature is missing and the the case where the similarity is actually 0.0.

SVM for New Event Detection.

This SVM is trained using the same options as described above with the difference that we use an RBF kernel in the

following form: $\Phi(d,e) = exp(-\gamma \cdot ||d-e||^2), \gamma = \frac{1}{4}$. We use split 2 to train this SVM. As split 2 contains 1,900 unique events, the number of positive examples amounts to 1,900. We use an equal number of negative examples to have a balanced training set. The optimal value for the hyperparameter θ_n is determined by gradient descent on split 3 using F-Measure as optimization criterion, yielding an optimal value of 0.63 for our approach and 0.48 for the approach of Becker et al.

3.1.4 Candidate Retrieval

We calculate the distance between a document and all events in the event database for the single features *capture time*, *upload time*, *geographic position*, *tags*, *title*, and *description* using the corresponding similarity metrics. The events are ordered by their distance to the document. Then, k events beginning from the less distant are retrieved. We determined empirically that – for optimal results – we need to retrieve the top-200 documents by capture time, the top-50 documents by upload time, and the top-20 each for geographic information, tags, title, and description, thus retrieving $200 + 50 + 4 \times 20 = 330$ events per document.

3.2 Results

3.2.1 Candidate Retrieval

Using our candidate retrieval as described above, in 99.9% of all cases the correct event cluster is in the subset of candidates returned. This is clearly a satisfactory results corroborating the effectiveness of our candidate generation strategy.

3.2.2 Scoring and Ranking

Figure 2 plots the accuracy of the SVM used to calculate $P(e|d)$ over different combinations of features. Here accuracy represents the number of cases in which the top-ranked event is also the correct one. It can be appreciated that using all features yields the best results (99.4%). But even when using a subsets of all features, the results are compelling. Using only the time and tag similarities, the accuracy is already 98.9%.

3.2.3 New Event Detection

For the new event detection task, we conducted a feature analysis in order to find an optimal combination of features considering the accuracy of the decision. In particular, we employed a greedy strategy, adding the next best feature in each step as long as the accuracy increased. The six features we consider here are the ones described in Section 2.4. Figure 3 depicts the steps of this greedy search strategy. We can see that the combination of the four features *max, maxtsu, avg, stddev* yields the highest accuracy of 85.9%.

3.2.4 Overall System Performance

We compare the overall performance of our system in terms of Precision, Recall and F-Measure to the performance of the baseline system of Becker et al. We compare in fact the following three systems:

- **Becker et al. (Baseline)**: this is the re-implemented CLASS-SVM method described Becker et al. [4].
- **Becker et al. with Blocking**: the method of Becker et al., extended with our candidate retrieval strategy.

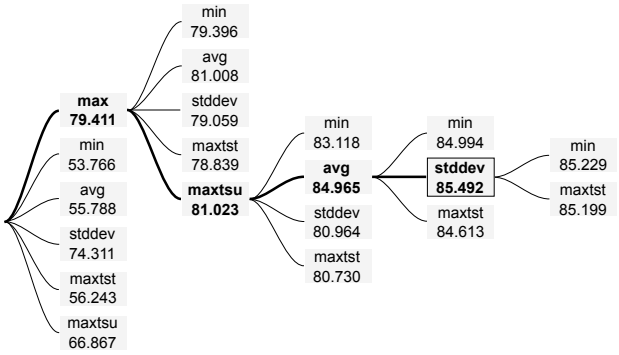

Figure 3: Greedy search for optimal features for the new event detection task

- **Our method**: our system with the two decision models and the candidate retrieval step.

The results are reported in Table 1 in terms of Precision, Recall and F-Measure, averaged over 7 folds. The key observations here are the following:

- Blocking has a crucial impact on the performance in terms of quality, corroborated by the fact that our method and the approach of Becker et al. extended by our candidate retrieval step clearly outperform the baseline. It is indeed surprising that the candidate retrieval step does not only increase the efficiency of the system, but also increases the classification performance. The reason for this is that the candidate retrieval step is eliminating many events that the document is unlikely to belong to, thus eliminating noise that might confuse the scoring and ranking as well as the new event detection components.
- While our method has a comparable performance to an extension of the baseline of Becker et al. by our candidate retrieval component in terms of F-Measure, our method has a much higher precision (80% vs. 71%), which is an advantage in our settings as we assume that a user is interested in seeing "pure" clusters with only few spurious examples. Further, we are currently experimenting with a post-processing strategy that aims to increase recall by merging clusters of similar events. However, as increasing recall always comes at the expense of reducing precision a higher precision as obtained by our approach in comparison to the one of Becker et al. is benefitial.

It is important to note that the performance of the Becker et al. approach is much lower compared to the results reported in their paper [4], i.e. F=59% vs. the F-Measure of 81%. This difference has a reasonable explanation. We have tested our system with a much higher number of documents compared to Becker et al. who used a dataset consisting of 27,000 documents. When using more documents, there are also more events and the chance of assigning a document to a wrong event is thus much higher. This explains the lower F-measures we report here compared to the ones reported by Becker et al. When testing our system and our re-implementation of Becker et al. on splits of 27,000 documents only, we get in fact F-measures of around 81%. If

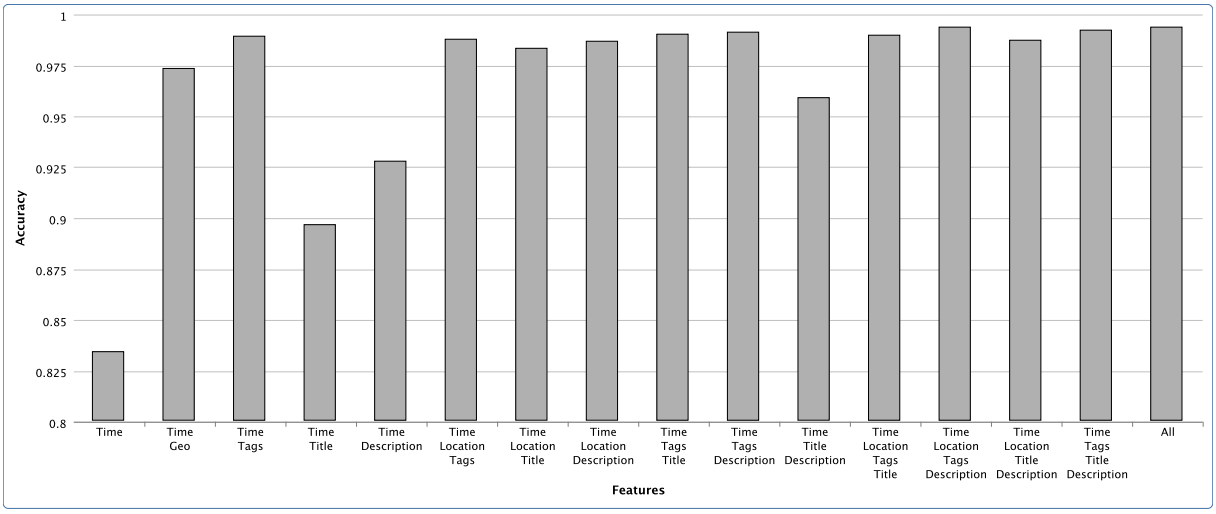

Figure 2: Accuracy of SVM calculating $P(e|d)$ for different combinations of features

Table 1: Results of different methods using 100,000 splits

Method	Prec	Rec	F-Measure
Becker et al. (Baseline)	0.464	0.818	0.589 (±3%)
Becker et al. w/ Blocking	0.714	0.784	0.744 (±2%)
Our Method	0.803	0.696	0.744 (±4%)

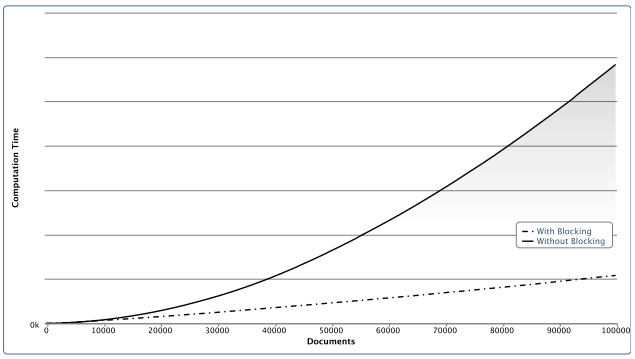

Figure 4: Comparison of runtime for system using blocking or not

we use 10,000 documents we get F-Measures around 87%, clearly showing that classifying documents with respect to a lower number of events is inherently easier.

3.2.5 *Efficiency*

We also compare our approach to the baseline in terms of efficiency, both in terms of absolute runtime as well as in (average) processing time per incoming document. Figure 4 shows the absolute runtime of our system compared to the one of Becker et al. While the absolute runtime increases exponentially in the approach of Becker et al., our runtime increases linearly. This result is particularly remarkable given the fact that we have much higher quality in terms of F-Measure compared to the approach of Becker et al. Figure 5 shows the average processing time per document over the number of documents processed by the system. While the

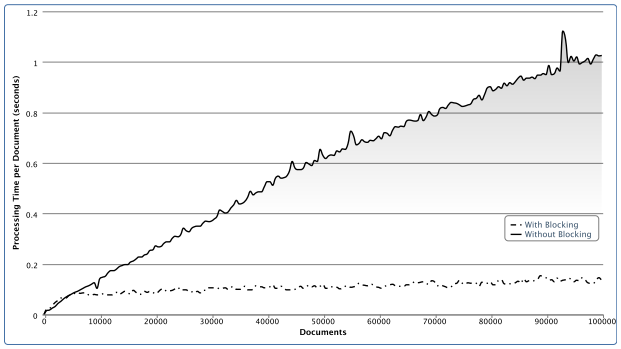

Figure 5: Comparison of computing time for for document processing using blocking or not

system of Becker et al. shows a linear increase in the number of documents, our system shows a nearly constant processing time. Our approach can thus scale to data streams of infinite size (theoretically), assuming that all events and documents can be stored physically.

4. RELATED WORK

The problem of *event identification in social media* was introduced by Becker et al. [4] who presented an incremental clustering algorithm that classifies social media documents into a growing set of events. They use a comparable but much smaller dataset than that in our work, thus reporting higher F-Measures than we do. We have re-implemented their approach, evaluating it on our dataset and under the same conditions, showing that it is clearly outperformed by our approach both in terms of quality and efficiency. There has been other work on detecting and indexing events in social media streams. Chen and Roy [9] present an approach to detect events which is based on exploiting the temporal and locational distribution of tags, distinguishing between periodic and aperiodic events. In this sense their work is related to the one of Rattenbury et al. [26] who have presented a method called Scale-structure Identification that can also be applied to identify tags that represent events as

well as places. The task is however not directly related to our task as we are not only concerned with identifying tags which represent events, but with producing clusters of social media items that collectively describe or are related to an event. Mattivi et al. [23] have presented an application of the concept of indexing media by events to the organization of photo collections which corroborates the usefulness of organization of data by events from an end user perspective. In contrast, Liu et al. [20] have addressed the opposite problem, i.e. the one of finding media that describe or illustrate a given event. Sayyadi [31] have presented an approach that exploits user community detection methods used in social network analysis to compute keyword graphs based on keyword co-occurrence in order to discover and describe events.

As we have argued in the introduction, the problem of classifying a stream of social media data into events can be seen as an instance of stream data classification where the set of classes is constantly growing and evolving. Several supervised learning approaches have been developed to perform classification of the data items in a data stream. The main challenge addressed by the various approaches is the detection of concept drift, consisting essentially in the decision whether a new data point belongs to one of the patterns or classes seen so far or to a new class [11, 10, 19]. The *new event detection problem* can be seen as an instance of the outlier, anomaly or novelty detection problems [21, 22]. Different approaches have been proposed to solve this problem, among them i) unsupervised or clustering-based, ii) supervised or classification-based and iii) statistical approaches [8].

Clustering-based approaches attempt to discover low-density regions in the space, assuming that outliers or anomalies belong to a cluster with a density below a threshold [7, 13].

Other techniques assume that "normal data points" belong to clusters, while anomalies do not belong to any cluster [15, 18]. The algorithm by Yu et al. [37] for example regards all documents which have not been assigned to any cluster as outliers. Frequent item-set mining has also been applied to detect normal patterns in training data, regarding all those data points that do not fit the frequent item-set patterns [3]. While most of these algorithms work only in an offline setting, some authors have proposed online methods and applied them to sequence data [5, 6]. Allan et al. [2] proposed an incremental clustering approach which allows to detect events in text document streams. Other approaches rely on a distance-based criterion and define an outlier point to be one that is, on average, furthest away to all other points (see [34]).

Classification-based approaches use discriminative techniques to separate high-density from low-density regions. SVMs, Neural Networks and Bayesian Networks have been applied in this context [8]. Ratsch et al. [25] consider the problem of outlier detection as a one-class learning problem, using RBF kernels to define complex regions that contain the training data instances. For each test data instance, it is determined if the instance falls in this region, regarding it as an outlier if it does not. Other authors use Neural Networks to detect novelties. The underlying idea is to test an input for acceptance in the network. If it does not get accepted, it is regarded as an outlier [22]. Theofilou et al. [33] presented an approach called *Long-term Depressant SOM* (LTD) where the weight vectors of the cells in the map become more and more dissimilar to the seen training instance. Contrary to the classic Kohonen network where weight vectors move close to the input distribution, in the LTD-like case weights move away from it. New data points that are close to these weight vectors are then regarded as outliers.

Statistical approaches can be divided into parametric and non-parametric. Parametric techniques rely on standard statistical tests such as Grubb's test [1], the t-test [32] and χ^2 [36]. They have been used to check whether a new data point indeed stems from the training distribution or not. Non-parametric approaches do not make strong assumptions about the distribution or density of data. Histogram-based techniques that maintain a profile of the normal data have been applied here [12, 14, 16].

5. CONCLUSIONS

We have presented a system that is able to classify a stream of social media data into a growing and evolving set of events. We have applied our method to a dataset derived from Flickr, using machine tags representing a unique event from last.fm to define a gold standard. We have shown in particular that our approach addresses successfully two key problems: i) scaling to the data sizes and data rates encountered in social media applications, and ii) tackling the new event detection problem, i.e. the problem of determining whether an incoming data item belongs to a new or a known event. We successfully address these problems by i) including a candidate event retrieval steps that retrieves a set of event candidates that the incoming data point is likely to belong to and ii) by including a function trained using machine learning techniques to determine whether the incoming data item belongs to the top scored candidate or rather to a new event. We have in particular shown that the new event detection decision can be taken with reasonable accuracy and carried out a feature analysis using a greedy strategy to find an optimal combination of features. We have directly compared our system to the approach of Becker et al., showing that it outperforms this baseline both in terms of F-Measure as well as in terms of efficiency. The benefit of our approach is that it scales as the processing time per documents remains nearly constant with increasing documents, thus addressing the scalability challenge mentioned above. There are two important issues that we intend to address in future work. First, we intend to systematically analyze the impact of different blocking strategies involving different computational costs on the task of classifying social media documents into events. Second, we intend to investigate whether the performance, in particular the recall, can be increased by extending our approach by a second step which merges different events into one.

6. REFERENCES

[1] C. Aggarwal and P. Yu. Outlier detection for high dimensional data. *ACM Sigmod Record*, 30(2):37–46, 2001.

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

[3] D. Barbará, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *Proceedings of CIKM*, pages 582–589, 2002.

[4] H. Becker, M. Naaman, and L. Gravano. Learning similarity metrics for event identification in social

media. In *Proceedings of WSDM*, pages 291–300, 2010.

[5] G. Bejerano and G. Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17(1):23–43, 2001.

[6] S. Budalakoti, A. Srivastava, R. Akella, and E. Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. *NASA ARC, Tech. Rep. NASA TM-2006-214553*, 2006.

[7] P. Chan, M. Mahoney, and M. Arshad. A machine learning approach to anomaly detection. *Department of Computer Sciences, Florida Institute of Technology, Melbourne*, 2003.

[8] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[9] L. Chen and A. Roy. Event detection from flickr data through wavelet-based spatial analysis. In *In Proceedings CIKM*, 2009.

[10] Q. Ding, Q. Ding, and W. Perrizo. Decision tree classification of spatial data streams using peano count trees. *Proceedings of the ACM Symposium on Applied Computing*, 2002.

[11] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of KDD*, pages 71–80. ACM, 2000.

[12] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of ICML*, 2000.

[13] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection. *Applications of Data Mining in Computer Security*, 6:77–102, 2002.

[14] E. Eskin, W. Lee, and S. Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 165–175. IEEE, 2001.

[15] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of KDD*, pages 226–231, 1996.

[16] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of KDD*, pages 53–62. ACM, 1999.

[17] J. Gao, W. Fan, J. Han, and P. Yu. A general framework for mining concept-drifting data streams with skewed distributions. *Proc. of SDM'07*, 2007.

[18] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[19] M. Last. Online classification of nonstationary data streams. *Intelligent Data Analysis*, 6(2):129–147, 2002.

[20] X. Liu, R. Troncy, and B. Huet. Finding media illustrating events. In *Proceedings of ICMR'11*, pages 1–8, 2011.

[21] M. Markou and S. Singh. Novelty detection: a review–part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497, 2003.

[22] M. Markou and S. Singh. Novelty detection: a review—part 2:: neural network based approaches. *Signal Processing*, 83(12):2499–2521, 2003.

[23] R. Mattivi, G. Boato, and B. G. D. Natale. Event-based media organization and indexing. *Infocummunications*, (3):9–18, 2011.

[24] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[25] G. Ratsch, S. Mika, B. Scholkopf, and K. Muller. Constructing boosting algorithms from svms: an application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1184–1199, 2002.

[26] T. Rattenbury, N. Good, and M. Naaman. Towards automatic extraction of event and place semantics from flickr tags. In *Proceedings of SIGIR*, pages 103–110, 2007.

[27] S. Rendle and L. Schmidt-Thieme. L.: Scaling record linkage to non-uniform distributed class sizes. In *Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2008.

[28] T. Reuter and P. Cimiano. Learning similarity functions for event identification using support vector machines. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval (KDIR)*, 2011.

[29] T. Reuter, P. Cimiano, L. Drumond, K. Buza, and L. Schmidt-Thieme. Scalable event-based clustering of social media via record linkage techniques. In *Fifth International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.

[30] S. Robertson and K. Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[31] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *In Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2009.

[32] C. Surace and K. Worden. A novelty detection method to diagnose damage in structures: an application to an offshore platform. In *Proceedings of the International Offshore and Polar Engineering Conference*, number 8, pages 64–70, 1998.

[33] D. Theofilou, V. Steuber, and E. Schutter. Novelty detection in a kohonen-like network with a long-term depression learning rule. *Neurocomputing*, 52:411–417, 2003.

[34] A. Vinueza and G. Grudic. Unsupervised outlier detection and semi-supervised learning. Technical report, Technical Report CU-CS-976-04, University of Colorado at Boulder, 2004.

[35] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of KDD*, pages 226–235. ACM, 2003.

[36] N. Ye and Q. Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2):105–112, 2001.

[37] D. Yu, G. Sheikholeslami, and A. Zhang. Findout: finding outliers in very large datasets. *Knowledge and Information Systems*, 4(4):387–412, 2002.