**Int J Multimed Info Retr manuscript No.**
(will be inserted by the editor)

# Detection of Social Events in Streams of Social Multimedia

**Jonathon Hare · Sina Samangooei · Mahesan Niranjan · Nicholas Gibbins**

Received: date / Accepted: date

**Abstract** Combining items from social media streams, such as Flickr photos and Twitter tweets, into meaningful groups can help users contextualise and consume more effectively the torrents of information continuously being made available on the social web. This task is made challenging due to the scale of the streams and the inherently multimodal nature of the information being contextualised.

The problem of grouping social media items into meaningful groups can be seen as an ill-posed and application specific unsupervised clustering problem. A fundamental question in multimodal contexts is determining which features best signify that two items should belong to the same grouping.

This paper presents a methodology which approaches social event detection as a streaming multi-modal clustering task. The methodology takes advantage of the temporal nature of social events and as a side benefit, allows for scaling to real-world datasets. Specific challenges of the social event detection task are addressed: the engineering and selection of the features used to compare items to one another; a feature fusion strategy that incorporates relative importance of features; the construction of a single sparse affinity matrix; and clustering techniques which produce meaningful item groups whilst scaling to cluster very large numbers of items.

The state-of-the-art approach presented here is evaluated using the ReSEED dataset with standardised evaluation measures. With automatically learned feature weights, we achieve an $F_1$ score of 0.94, showing that a good compromise between precision and recall of clusters can be achieved. In a comparison with other state-of-the-art algorithms our approach is shown to give the best results.

**Keywords** Social Event detection · Clustering methods · Scalability · User-generated content

Jonathon Hare, Mahesan Niranjan, Nicholas Gibbins
University of Southampton, United Kingdom,
E-mail: {jsh2, mn, nmg}@ecs.soton.ac.uk

Sina Samangooei
Amazon, Poseidon House, Castle Park, Cambridge, U.K.
E-mail: samangoo@amazon.com

## 1 Introduction

In their June 2013 WWDC keynote, Apple announced a new photo collection feature for their iOS mobile operating system. With the evocative tag-line "Life is full of special moments. So is your photo library", Apple noted the importance of clustering social streams by their belonging to some real life event. This, along with the plethora of mobile and desktop applications which offer some degree of event detection in user photo streams, demonstrates that detecting events in multimedia streams can have both real and practical utility for end users.

If detection and clustering by events within private collections is useful, detecting events and clustering items in a multi-user social media context should also have practical benefits. For example within the EU funded ARCOMEM project [20] we have been exploring how such techniques can be applied to the archiving and contextualisation of collective memories of certain events. Within this context, challenges of scale and noise inherent in non-curated collections must be addressed to create meaningful groupings of social media artefacts which afford the user the ability to better understand, consume and contextualize social streams.

This work presents an approach to achieving clustering of social media artefacts into "Social Events". We use the definition of a "Social Event" from the Social Event Detection (SED) challenge of the 2013 MediaEval benchmark [18]:

> Events that are planned by people, attended by people and the media illustrating the events are captured by people. A social event of interest can be specified in terms of event-related metadata (e.g., location, time, venue, and performers), example tags or other social information, example media items (images), or a combination of the above.

The novel approach developed in this work builds upon our submission to the 2013 SED challenge which was briefly outlined in a working notes paper [21], by further developing the underlying theory, performing detailed comparative evaluation, and discussing insights into the problem. The approach is grounded in the idea that social multimedia can be considered to be a temporal stream of data, and that stream of data has a structure or pattern that is a result of the nature of real-world (social) events that result in multimedia items being shared. In particular, the notion that users tend to share multimedia artefacts representing an event around the same time as each other is exploited to aid both effective and efficient clustering. In particular this notion enables scalability of the proposed approach by limiting the amount of data that needs to be considered at any given time. The following points summarise the novel combination of features that make up our approach:

- Efficient metadata-based feature extraction using a fast inverted index.
- Multimodal features and distance metrics coupled with effective weighting schemes used to construct a sparse affinity matrix.
- Exploration of two clustering techniques which cluster events using this affinity matrix; namely a modified DBSCAN that consumes affinity matrices, and Spectral Clustering.
- Development of an incremental event clustering technique which enables the base clustering techniques to be used at scale, whilst at the same time exploiting the temporal nature of the data.

The approaches and techniques proposed in this work are evaluated using the ReSEED dataset [19], together with the evaluation methodology first proposed for the first task (Task 1) of the the Social Event Detection (SED) challenge of the 2013 MediaEval benchmark [18]. The proposed technique is shown to achieve state-of-the-art performance under the conditions of this evaluation. The key challenge of the evaluation is to organize a large collection of Flickr photos into social event groupings.

The ReSEED dataset contains a large and diverse array of Flickr images corresponding to a heterogeneous assortment of different social events and social event types. Each image in the dataset is accompanied with additional metadata. Namely, the Flickr photos are guaranteed to include accurate: Flickr photo IDs, user IDs and time posted (the server-time at which the image was uploaded). The photos also contain, albeit with varying degrees of accuracy: location information, the time stamp[1] according to the capture device (or user), and textual information including the title, tags and a free-text description. Further details on the dataset and the MediaEval SED evaluation methodology can be found in Section 2.2.

The remainder of this paper is structured as follows: the following section places this work in context by exploring other techniques in Social Event Detection (SED). Section 3 describes our approach in detail including a broad overview of our feature weighting, fusion and clustering strategies. Section 4 describes the dataset and experimental framework of the SED task, including evaluation metrics and more details on the feature weighting and cluster parameter selection process. Section 5 provides experimental results, comparative evaluation, and discussion. The final section provides a summary of the major points of note from the experiments, together with concluding remarks and thoughts on how the limitations of this work may be addressed in the future.

## 2 Multimedia Social Event Detection

As outlined by Scherp et al [22] there is a great deal of interest in detecting multimedia related to high level events in which humans participate. A subtype of such events are the Social Events the MediaEval Social Event Detection task asks participants to detect. The challenge distinguishes Social Events as those events which were "planned by people, attended by people and that social media depicting the events are taken by people". This calls for events beyond such definitions as *birthday party* and towards more specific definitions such as *Sina's 30th Birthday Party.*

When attempting to organise multimedia items into those belonging to the same social events, temporal and spatial metadata is the most powerful indicator of event membership. It is clear that if the true time and true location of a multimedia item could ever be

[1] called *time taken* in the task

known with complete accuracy, and if an assumption is made that all the multimedia items being processed represent events, then the clustering of items into social events would be made far easier in most contexts. This is stated explicitly in the problem definition provided by Becker et al [1] who say that an event is something that: "... occurs in a certain place at a certain time...". Indeed, it is difficult to imagine two items of multimedia taken in the same place at the same time which would not in some sense depict the same social event.

However, a strong edge case of this statement is a restaurant which might have many, though separate, groups of people in the same evening. Though the restaurant is one physical location, it is reasonable to assume that each table in the restaurant might be host to different social events. Therefore two multimedia items from two different groups, though geographically and temporally similar, should not be assigned to the same social event. This edge case highlights the issue of scale — namely, if an event occurs across a large enough space, sub geographic locations within that space could feasibly hold unrelated events. This means precise geographic and time information alone cannot help us distinguish events in these scenarios. It is worth noting that a study by Zandbergen and Barbeau [31] showed geographical accuracy of mobile camera phones has a root mean square accuracy of 12.5 meters when used outdoors and 21.6 metres indoors, so there is a real issue with co-located simultaneous events.

This issue notwithstanding, precise geographic and time information can almost perfectly achieve most of the desired goals in a standard SED task. Many of the approaches to social event detection highlight time and geographic location as the most important dimensions of separation in their solutions. A majority of the other efforts either explicitly or implicitly handle cases where time and location of information are noisy or non-existent.

In the remainder of this section we explore relevant prior work in the area of social event detection (SED). We then focus in detail on the techniques presented for the SED task in MediaEval 2013 [18], the benchmark against which the techniques in this work were originally evaluated. Finally, we reflect on the similarities of the prior-art to our own technique which is presented in Section 3.

### 2.1 Prior work outside of MediaEval

Zaharieva et al [30] proposed a system that attempts to achieve detection of specific social events as defined by a textual, temporal and geographic query. The approach demonstrates a common pipeline approach wherein all multimedia items are grouped through a sequence of unimodal clustering steps. Firstly, an initial clustering is attempted which uses the temporal and spatial information of the multimedia items. Further spatial clustering is then attempted by detecting geographic words used in the image tags and description. The detected clusters are then compared to the specifications of the cluster query.

Petkos et al [15] proposed a multimodal clustering approach for the detection of social events. In their baseline approach the authors used an aggregated affinity matrix coupled with spectral clustering. This is similar to our spectral clustering approach. In their second approach the authors use pairwise similarities over all modalities to predict a "same cluster" classifier which takes a vector of distances from an image to all other images as the classifier feature vector. However, both their approaches are inherently incapable of scaling to larger multimedia datasets. They either rely upon exhaustively holding all multimedia items to be clustered in memory, or they expect a distance vector to be calculated for each item to be clustered which incorporates a given items distance to all other items being clustered. Both demonstrate reasonable results, but were not applied to sets of data larger than 100,000 items.

Reuter and Cimiano [17] propose a more direct and scalable solution to social event detection. They highlight a set of features which might commonly exist in social multimedia items including: time of capture, time of upload, geo location, title, description and tags. They then implement a procedure where a database of seen items along with their event assignment are held. When a novel document is to be assigned to an event, an initial coarse query using searchable features such as textual components and time is used to receive a set of candidate items. Custom distance metrics for each feature and a distance fusion strategy is then used to calculate a distance between the new item and possible candidate items. Pre-trained SVM classifiers are then used to decide whether the item is more likely to belong to a new event, or whether it actually belongs to the event of one of the candidate items. This approach does not have the scale limitations present in Petkos et al [15], however, the use of a classifier to establish cluster membership relies explicitly on a representative training sample which portrays a similar item to item distance distribution. To cluster novel multimedia items in an ever changing stream *might* require a re-training of this classifier; this would be especially true over longer time-scales than are tested with current data, where language and fashions can change significantly.

An earlier work in the detection of social events by Becker et al [1] explores a multimodal event detec-

tion approach coupled with an incremental, one pass and scalable clustering approach. They highlight the main features against which events can be discovered, proposing the use of relatively straightforward distance metrics for each feature. They also highlight the importance of a scalable, single pass incremental clustering technique if clustering is to be used as the method of detecting events. It is understood that social multimedia is often produced in a data stream context, with novel items arriving over time which might belong to older clusters or which might be part of a new, previously unseen cluster. They outline a clustering technique which holds a record of previous clusters and attempts to assign a novel item to that cluster based either on similarity to a cluster's "centroid" multimedia item, or more directly through the feature comparison to each of the documents currently held in a cluster.

### 2.2 At MediaEval 2013

As mentioned in the introduction, the MediaEval 2013 SED task (challenge 1), involved performing a full clustering of a multimedia dataset of images collected from Flickr. Subsequent to the MediaEval event, the data has since been released openly and is now know as ReSEED [19]. The entire dataset consists of 437,370 creative-commons images uploaded between January 2006 and December 2012. Each image is assigned to an event using the techniques described by Reuter and Cimiano [17] in order to create a gold-standard ground truth of 21,169 events (covering everything from large sporting events to birthday barbecues). Each image has a varying amount of metadata associated with it; for example, 95.6% of the images contained associated tags, whereas only 45.9% contained geographic information. The dataset is split into two parts: a training set of 306,159 images and ground-truth (70% of the data) for training/optimising techniques, and a test dataset with 131,211 images (30%) for testing. During the MediaEval task, the ground-truth for the test dataset was withheld by the organisers. Teams submitted the raw clustering results for the test-set, which was then compared to the ground-truth by the organisers, who used three measures of clustering performance: $F_1$ score, Normalised Mutual Information (NMI) and Divergence from a Random Baseline [5]. Further details on these measures can be found in Section 4.1. During the original MediaEval 2013 SED task, 10 approaches attempting to cluster the data were submitted. Here we describe a few of the noteworthy approaches.

Nguyen et al [13] presented an approach which is rooted in a notion of how events are populated on social networks, namely that:

1. Users generate the multimedia item at the time in which the event occurs;
2. The user uploads, annotates and shares the media into a social network.

The approach directly capitalises on the principle that no one user can be in multiple events at a given time. Images are firstly separated by user, and within user sets separated into events by time according to a fixed threshold. Once these user based separations are made, between user sets are merged if they share location information, similar time taken and text (tag / title / description) information. Apart from the explicit treatment of time as a first-pass mode of separation, their approach did not explore custom feature metrics nor weightings.

Many other techniques, achieving reasonable results though not as good as the one described by Nguyen et al [13] followed a similar logic [16, 32, 10], performing some initial clustering based on user, time and location information, and later performing merging based on heuristic rules or other information such as text.

Another interesting approach is that of Schinas et al [23]. This technique starts by creating an index of images and detecting candidate images against this index which might be part of an event. Against these candidates they apply a feature comparison technique, resulting in a graph of image adjacencies. They then apply a network clustering algorithm called SCAN [29], which has some similarities to the DBSCAN [6] algorithm used in this work (see Section 3.2.1).

## 3 Our Approach

The goal of social event detection is to find groupings of social multimedia items such that the grouping of items represent social events. As described above, the majority of existing techniques work by posing the SED problem as a problem of data clustering. The rationale for this is that if *suitable features* can be extracted to describe the items, then computed similarities between features should indicate how related pairs of items are. The underlying assumption for detecting social events is then that items belonging to the same social event should be similar (or more concretely have similar features). To date, most proposed SED techniques have assumed that they are working with static, monolithic datasets. However, in actuality data items in the SED context have a highly temporal nature that we believe can be exploited to both aid the effectiveness and efficiency of event detectors.

The overarching strategy of the techniques which we describe in this section are based on the idea of treating

SED as an unsupervised *streaming* data clustering task. As a basis for group related media items (using standard clustering techniques), our approach constructs a square symmetric sparse affinity matrix whose elements represent the pairwise similarity of two items of social media computed from a number of feature modalities. However, rather than attempting to cluster an entire dataset in one go, we additionally leverage the notion that the data can be seen as a stream of media items uploaded over time. We hypothesise that this stream contains an embedded structure that occurs from the way users upload and share media items as a result of real-world social events. More specifically we believe that users tend to share multimedia artefacts representing an event around the same time as each other, and that it is common for an event to have items generated which relate to it over some set period of time, after which items belonging to that event are never posted again. To validate this hypothesis, we used the ground-truth dataset of the ReSEED dataset to generate the hourly distribution of uploads event for each event. Analysis of these distributions indicates that the most common events are captured by a single period of uploading within a time period of one hour. To illustrate the nature of these distributions, Figure 1 contains *sparkline* representations of a random sample of 495 distributions (from 3044 unique distributions extracted from the ReSEED training data). Each sparkline was computed using a bin-size of 1 hour, shows a period of two weeks from the initial image upload of an event and has been max-normalised to remove the effect of the number of photos belonging to an event.

The idea that the act of uploading of media items associated with a particular event takes place over a short period of time can be leveraged in the design of an algorithm for grouping the data. In particular, we can pose the solution to the problem as a streaming algorithm that is allowed to *forget* groupings previously made after a period of time (because these groupings are highly unlikely to ever see new items added to them). More specifically, we propose an approach in which we consider time window on the data stream and incrementally cluster the data within that window by assigning data to a pool of previously discovered clusters or creating new clusters within the pool. If a cluster becomes *stable* (i.e. it hasn't changed over a period of time) it can be forgotten and be removed from the cluster.

A benefit of the streaming approach is that only media items within a time window need to be considered at any point in time, and they only need to be compared to a small set of existing events. This obviously aids in the scalability of the algorithm. However, even when we consider only the media items within a relatively short time window, the creation of the affinity matrix scales poorly. If $n$ is the number of items, the construction of the affinity matrix is a time consuming $O(n^2)$ operation. Therefore, the first stage of our process is the efficient construction of such an affinity matrix via a constrained initial selection of images that might be related. This is followed by a feature comparison step to compute the affinities. Once feature comparison is achieved, a single affinity matrix is constructed through a weighted affinity matrix summation operation.

The following sections provide more details on each part of our approach. Firstly, we describe the construction of the affinity matrices in Section 3.1. We then provide a discussion of the clustering techniques we've used in our approach in Section 3.2. Finally, in section 3.3 we describe how we construct a streaming algorithm on top of the affinity matrix construction and clustering techniques.

### 3.1 Affinity Matrix Construction

The construction of affinity matrices in a dense manner is computationally intractable for the large numbers of images that appear in social streams. However, we can exploit the fundamental sparsity of the social data source to produce affinity matrices in a scalable manner.

For the $300,000$ Flickr images in the ReSEED training dataset there exist $14,000$ ground truth events, or clusters. The average number of items per cluster in the training set is therefore $\approx 20$ (although the actual distribution is highly skewed). From this information we hypothesize that the similarity between most objects must be 0 if similarity is a reasonable indication of cluster membership. This in turn implies sparsity of the affinity matrix. Inducing this sparsity after the feature extraction and comparison of the social media objects (as described in the next section) is an approach without merit — inducing sparsity after feature comparison would mean that the image to image comparison will already be performed thus implying a computationally expensive operation must be performed potentially needlessly.

To address this issue, we construct a Lucene[2] index of the items to be clustered. The items are indexed using their metadata. Each field of metadata is given a field in the Lucene index. Then, for each item in the dataset we construct a custom Lucene query based on the item's metadata, receiving an artificially limited number of documents. We then extract features and compare distances using only the top documents returned by this

[2] `http://lucene.apache.org/core/`

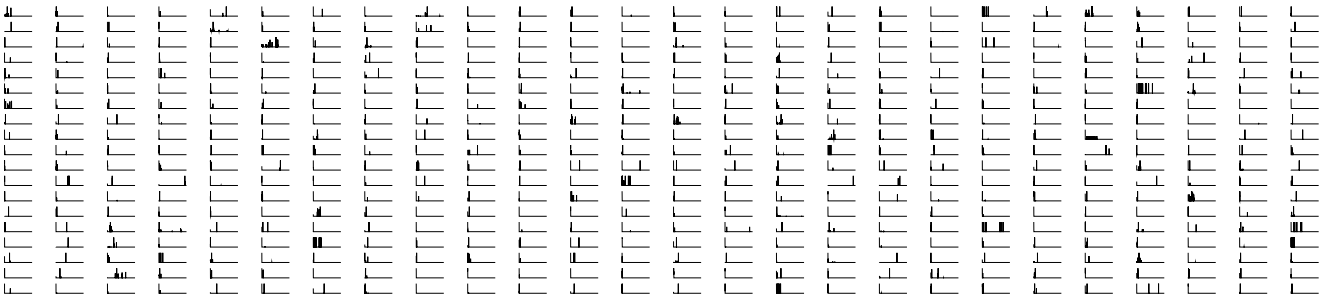


Fig. 1: Illustrations showing the temporal distribution of the uploading of media items corresponding to a specific event (based on the ReSEED training data). The data in each sparkline is max-normalised and each graph shows a 2 week period from the first uploaded image of an event. Upload events were binned into 1 hour time slots in order to compute the frequency.

query. For the purposes of our experiments described in Sections 4 and 5 we limited the maximum number of items retrieved to 200. Once the work is done to construct this Lucene index, this operation has a complexity of $O(n)$ which allows a much faster construction of the affinity matrix. This process of limiting the number of retrieved items is similar in spirit to the *blocker* or *candidate generator* described by [17]. However, in our embodiment, it is used in a rather different way as we're using it specifically for item-item comparison whereas Reuter and Cimiano specifically use it in the context of comparing a social media item against a set of indexed event descriptions.

#### *3.1.1 Multi-modal Affinity*

The Flickr images being clustered into social events are inherently multi-modal. These modalities include time information (both posted and taken), geographic information, textual information (tags, descriptions and titles) as well as the visual information of the Flickr photos themselves. Any of these modalities might serve as a strong signal of cluster membership. Photos taken in the same place, or at the same time, or containing similar text might all serve as strong indication of these photos being of the same event. However, on their own the features might also serve to confuse unrelated events, for example, two events happening on a Friday, but one in Nottingham and one in London.

Therefore, the first stage in the construction of a unified affinity matrix is a separate affinity matrix for each of these features, while the second step is the intelligent combination of the affinity matrices.

Inspired by Reuter and Cimiano [17] we use a normalised logarithmic similarity function for our two time features, although we additionally set negative similarities to zero to ensure sparsity of the affinity matrix:

$$sim_{time}(t_1, t_2) = max(1 - \frac{log(|t_1 - t_2|)}{log(t_{norm})}, 0) \quad (1)$$

where $t_1$ and $t_2$ are the times of the items being compared and $t_{norm}$ is the normalisation factor. Any difference between the times of the items that exceeds $t_{norm}$ will result in a similarity of zero. For all our experiments, $t_{norm}$ was set to the length of a year (in our implementation this was measured in minutes, and the times $t_1$ and $t_2$ were expressed as minutes since the Epoch). We also used the same form of truncated normalised logarithmic similarity function for geographic Haversine distance between the geo-coordinates of two items, $\mathbf{g_1}$ and $\mathbf{g_2}$:

$$sim_{geo}(\mathbf{g_1}, \mathbf{g_2}) = max(1 - \frac{log(Haversine(\mathbf{g_1}, \mathbf{g_2})}{log(g_{norm})}, 0) \quad (2)$$

The normalisation factor $g_{norm}$ forces Haversine distances beyond the normalisation factor to count as being infinitely far, or as having 0 similarity. In our experiments, the $g_{norm}$ was set to a distance of 10000 metres.

For the textual features we use the TF-IDF score with the IDF statistics calculated against the entire corpus of Flickr objects. We also (briefly) experimented with SIFT visual features (using locality sensitive hashing to compare feature matches [8]) for image feature affinity matrix construction, however, we found this feature only made $F_1$ scores worse in the training set. For all the experiments presented in this work, the visual features are completely ignored in all runs against the test set. We do however discuss visual features further in Section 7.

If any given feature is missing or empty for either object represented by a particular cell in the affinity matrix, for the purpose of the sparse affinity matrix it is treated as being "not present" rather than having 0 similarity. The distinction here is important for the process of combining the separate affinity matrices for each feature. In essence, our approach is to use the average feature value for any case where the actual value is unknown (as is common in the matrix completion and recommender system literature). Full details of how this is achieved are given in Section 4.2 where we describe different experimental approaches to building the fused affinity matrix.

While Reuter and Cimiano [17] constructed vectors of similarity, we choose to fuse the similarity features into a single similarity score to construct a fused affinity matrix. This single, sparse affinity matrix makes for easy application of existing techniques like spectral clustering as well as for efficient implementations of DBSCAN.

We experimented with various feature-fusion techniques to combine these affinity matrices, including: *product*, *max*, *min*, and *average*; amongst which the *average* strategy was found to work best. We also explored different feature weightings by performing a search across the simplex between each feature weighting. We discuss the search strategy in further detail in Section 4.2 along with the related search to find optimal parameters for the clustering techniques discussed in the next section.

## 3.2 Event Clustering

The exact number of social events in a given corpus cannot be known accurately in advance, and may be difficult to accurately estimate. This is especially true in a streaming corpus where the number of clusters is inherently dynamic and fluctuates over time. To help cluster images in such a context, and therefore detect events, we explored two clustering techniques which work without an explicit prior number of clusters, $k$. In this section we review these techniques. It should be noted that there are many techniques we could chosen to explore, but we decided to focus on two very different, but exemplary, techniques for clustering without apriori knowledge of the number of clusters or cluster size. The results later in the paper indicate that there is little difference in performance between these two approaches.

The first technique we consider is a simple yet powerful modification of the classic DBSCAN algorithm, implementing a fast version of the neighbourhood selection stage for sparse affinity matrices. Secondly, we explore spectral clustering techniques, exploiting the sparsity of the affinity matrix to discover a data projection which aims to better separate clusters.

### *3.2.1 DBSCAN*

The Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was initially proposed by Ester et al [6]. It has been applied to handle large-scale data problems [29] over numbers of data points which other clustering methods fail to handle. DBSCAN can be implemented as a one pass algorithm, requiring only a function which can define which individual data items are within a neighbourhood of given starting data item. Such an operation can be implemented efficiently given appropriate indexing structure. Therefore DBSCAN enjoys $O(n \log(n))$ complexity as compared to $O(n^k \log(n))$ complexity of k-means or the $O(n^3)$ time complexity of the eigenvector decomposition required for spectral clustering (described below). Other one pass algorithms have been attracting a great deal of attention [11] to meet the challenges of clustering large datasets.

These efficiency benefits aside, DBSCAN:

1. Successfully finds clusterings with arbitrary shape (as compared to the spherical-Gaussian distributions implicit in k-means);
2. Does not require an explicit statement of the number of clusters expected;
3. Can report that a given data item is likely noise rather than a member of a cluster.

The basic DBSCAN algorithm defines the *$\epsilon$-neighbourhood* of a data point $p$ as a number of points, $N_\epsilon(p)$, within $\epsilon$ of $p$:

$$N_\epsilon(p) = \{q \in \mathcal{D} \,|\, \mathrm{dist}(p, q) \leq \epsilon\} \tag{3}$$

However a more general definition of DBSCAN requires only the points within some neighbourhood $p$ — a notion which is potentially independent of measurements of distance. A point can be seen as being densely surrounded, and therefore within a cluster, if $|N_\epsilon(p)| \geq minPts$, where $minPts$ is the second variable of DBSCAN. Two points $p$ and $q$ are *directly density-reachable* if $p$ is densely surrounded and $q \in N_\epsilon(p)$, and are said to be *generally density-reachable* if there are a chain of points between $p$ and $q$ such that each point in the chain is *directly density reachable* from the previous point. The *generally density reachable* property is not symmetric. If one were to start at $q$, a point on the edge of the cluster. $p$ would not be density reachable because $q$ itself would not be considered to be *directly density reachable* to any point (being on the edge of a cluster). However, by starting at $p$, which is part of the cluster,

one could find $q$. Therefore the final definition is that of *density connected* points $p$ and $q$ which are density reachable to some shared point $o$. A cluster in DBSCAN is a collection of points which are all mutually density connected to one another.

In our affinity matrix DBSCAN algorithm, we exploit the structure of a sparse matrix and a threshold to create an efficient neighbourhood function. The parameters of our affinity matrix DBSCAN approach are the threshold of affinity used to define the $\epsilon$-*neighbourhood* function against an affinity matrix and the $minPts$ parameter which defines how many neighbours are required for a point to be considered density-reachable. We describe how values for these parameters were selected which were optimal for the detection of social events in Section 4.2.

#### *3.2.2 Spectral Clustering*

Spectral clustering [24, 12] is an algorithm that was shown to achieve state-of-the-art performance for a range of tasks, from image segmentation [24] to community detection [25]. This technique treats the clustering problem as one of graph partitioning on the similarity graph between objects. The algorithm projects the objects via Singular Value Decomposition into a reduced dimension space which aims for maximal separation of clusters. Because of this, spectral clustering is useful when data dimensionality is high. Spectral clustering also has an inbuilt ability to detect the number of clusters in a given dataset making it useful when this cluster count cannot be easily predicted. The algorithm is also appealing because it is grounded by spectral graph theory [4].

The spectral clustering algorithm [27] works as follows: we start with the affinity matrix $W$, the diagonal matrix $D$ with elements equal to the row sums of $W$ and the graph Laplacian $L$. A few Laplacians exist depending on the partitioning problem we aim to solve. For example, the $L_{rw} = I - D^{-1}W$ Laplacian, is used to find a clustering such that a random walk in the graph rarely changes cluster memberships. The solution to this random walk cluster partition problem is NP-hard in the general case. The spectral clustering algorithm solves a relaxed version of this problem by finding the $k$ eigenvectors of $L$ with the smallest eigenvalues (ignoring 0 valued eigenvalues).

The selection of these $k$ small eigenvectors is simple when cluster boundaries are clear, but as datasets become more realistic the distribution of eigenvalues changes and the transition from a *small* eigenvalue to a *large* value becomes less well defined. In our work we choose to threshold the max-normalised eigenvalues and choose small eigenvalues up to some threshold, $eigenthresh$ relative to the normalised difference to the first non-zero eigenvalue, $\lambda_2$. Formally we choose the first (smallest) $k$ eigenvalues such that:

$$k = max_i(\frac{\lambda_i - \lambda_2}{\lambda_{max}} < eigenthresh) \quad (4)$$

where $\lambda_{max}$ is the value of the largest eigenvalue. The objects are clustered with a standard clustering algorithm in this reduced space, in our solution we choose DBSCAN with a $\epsilon$-*neighbourhood* function which uses a thresholded cosine distance between elements in the eigenvector space.

The parameters of our spectral clustering approach are the threshold of eigenvalue selection, the threshold of the cosine distance function for the DBSCAN $\epsilon$-*neighbourhood* check and the $minPts$ parameter for determining density-reachability (see Section 3.2.1). We describe how values for these parameters were selected which were optimal for the detection of social events in Section 4.2. Further details on spectral clustering can be found in the tutorial by Von Luxburg [27].

### 3.3 Incrementally Clustering the Media Stream

Spectral clustering requires the eigendecomposition of the Laplacian of the affinity matrix. The calculation quickly becomes intractable for datasets over $100,000$ items. Our sparse affinity matrix DBSCAN is a relatively efficient algorithm and can easily cluster the number of items in the ReSEED dataset in memory. However, even DBSCAN has limits in terms of performance. In its unmodified form it requires the affinity matrix of the entire space to be held in memory, which will eventually become intractable with large numbers of items.

As mentioned previously, social media data arrives as a stream, with items being uploaded over time. We make the explicit assumption that items which depict an event will start being uploaded at some point in time, continue being uploaded for some period of time, and then stop being uploaded. In this section we propose an incremental clustering technique that takes advantage of the streaming nature of social media data in order to both improve clustering performance as well as to respond to the scaling challenges.

Given our assumption about the temporal distribution of uploads belonging to an event, coupled with the data items appearing in stream, our incremental clustering algorithm proceeds as follows: Firstly, a small window of size $C_1$ of our whole dataset of size $N$ is clustered such that $|C_1| \leq N$ and $C_1$ is small enough to allow for easy clustering. The portion of data which is clustered represents some block of data uploaded to

the system sequentially in time. We then consume more data and perform the clustering again but this time on a window $C_2$ such that $|C_2| = 2 * |C_1|$. We might notice that certain clusters detected in the original $C_1$ remain *stable* when re-detected in $C_2$. This stability can be defined as a cluster whose members do not change whatsoever between the clustering of $C_1$ and $C_2$. A relaxed form defines stability as paired clusters with high overlap or similarity metrics (see Section 4.1). Regardless, once a set of clusters $c_i$ are identified as stable, items in that cluster are removed and not involved in future rounds of clustering. Therefore in future iterations, a data window $C_3$ might be clustered such that $|C_3| = |C_2| - \sum_i |c_i| + |C_1|$ — i.e. the window size is increased to include more elements, but data elements in the stable clusters $c_i$ are not clustered again. As illustrated in Figure 2, this results in the increase of the effective number of items being clustered as new items arrive, but will also result in a gradual decrease of items to be clustered as clusters are identified as stable. By using this incremental scheme we were able to successfully apply the spectral clustering algorithm to a large set of 300,000 items unlike Petkos et al [15] who also applied a spectral clustering technique to event detection, but on a comparatively small sample size of 40,000 items.

## 4 Experimental Framework and Parameter Optimisation

In this section we highlight the procedures undertaken to calculate parameters for the various algorithms used to detect social events. Firstly, the quality metrics of the social events being detected are described, including the ground truth and the cluster quality metrics used. Secondly cluster parameter and feature weight optimisation is discussed, using the quality metrics to search for optimal parameters on a training set of data.

### 4.1 Evaluating Social Event Detection

Our approach to detecting social events is a complete subdivision of data into clusters, such that each cluster represents an event and all items in a given dataset belong to a single event. To measure the quality of these clusters, they are compared to some set of true data subdivisions. This ground-truth is created by detecting Flickr images which contain last.fm machine tags and using the *eventIds* of these tags to create a set of images with known event subdivisions [17].

Given this ground-truth information, evaluating the quality of a particular set of detected social events remains non-trivial. In the ReSEED dataset used in the MediaEval 2013 SED challenge, two main cluster quality metrics were used to judge whether items clustered by participant algorithms represented events identified in the ground truth. These were the Normalised Mutual Information (NMI) and the harmonic mean of precision and recall, or $F_1$ score. The divergence of $F_1$ and NMI from a random baseline [5] was also calculated.

In the following description of these metrics in the context of a clustering task, we consider two document assignment sets $X$ and $Y$, both holding partitions of the same set of $N$ documents. $X = \{x_1, \ldots, x_K\}$ is the clustered document assignments made by our algorithm; each $x_k$ represents the set of documents assigned to a specific cluster. $Y = \{y_1, \ldots, y_J\}$ is the set of true classes assigned from the ground truth; each $y_j$ represents the set of documents actually generated from each event.

#### *4.1.1 Normalised Mutual Information*

The NMI evaluation metric between $X$ and $Y$ is calculated as follows:

$$\mathrm{NMI}(X,Y) = \frac{I(X,Y)}{[H(X)+H(Y)]/2} \tag{5}$$

$$I(X,Y) = \sum_{x}^{X}\sum_{y}^{Y} \mathrm{P}(x,y)\log_2 \frac{\mathrm{P}(x,y)}{\mathrm{P}(x)\mathrm{P}(y)} \tag{6}$$

$$H(X) = -\sum_{x}^{X} \mathrm{P}(x)\log_2(\mathrm{P}(x)) \tag{7}$$

where $I(X,Y)$ calculates the mutual information assignment the interval $[0,1]$. Given that the $X$ and $Y$ share the same documents, $I(X,Y)$ is high when most documents grouped in cluster $x_k$ are likely to be also generated by event $y_j$. This metric fails when $|X| \approx N$, i.e. there are very many clusters each with few items. Although such an $X$ partitioning may not match the correct clusters of $Y$, $I(X,Y)$ will tend to be high regardless because $x_k$ clusters are likely to only contain items of a single event $y_j$ by virtue of having few items. To correct for this the $\mathrm{NMI}(X,Y)$ metric weights $I(X,Y)$ by the class and cluster entropy, $H(X)$ which is high when $|X| \approx N$.

#### *4.1.2 Harmonic mean of precision and recall*

The $F_1$ score, $\mathrm{F}_1(X,Y)$, takes a non-information theoretic approach, concentrating instead on the true-positives ($\mathrm{TP}(X,Y)$), false-positives ($\mathrm{FP}(X,Y)$), true-negatives ($\mathrm{TN}(X,Y)$) and false-negatives ($\mathrm{FN}(X,Y)$). $\mathrm{TP}(X,Y)$ is defined as the total number of times the pairs of documents in each $x_k$ exist as a pair in a given $y_j$. By extension, $\mathrm{FP}(X,Y)$ is the total number of times the pairs

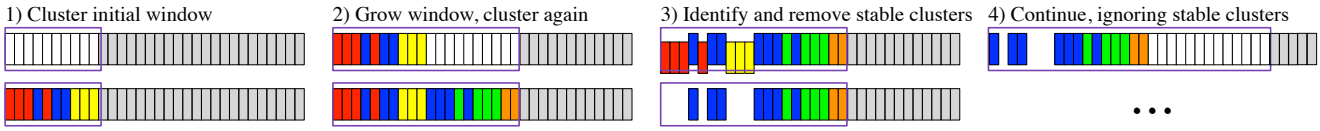



Fig. 2: Illustration of how the incremental clustering scheme works. The leading edge of the temporal window (denoted by the right-edge of the purple box) moves forwards at a different rate to the trailing edge (the left edge of the purple box) which moves forwards once stable clusters are identified.

appearing in each $x_k$ do not appear in any given $y_j$. $\mathrm{FN}(X,Y)$ is the number of pairs found in each $y_j$ which do not appear in any $x_k$ and $\mathrm{TN}(X,Y)$ is the number of pairs of items which do appear in any $y_j$ which also do not appear in any $x_k$. Using these values we can define the precision ($\mathrm{Pre}(X,Y)$), recall ($\mathrm{Rec}(X,Y)$) and the $\mathrm{F}_1(X,Y)$ as:

$$\mathrm{Pre}(X,Y) = \frac{\mathrm{TP}(X,Y)}{\mathrm{TP}(X,Y)+\mathrm{FP}(X,Y)} \tag{8}$$

$$\mathrm{Rec}(X,Y) = \frac{\mathrm{TP}(X,Y)}{\mathrm{TP}(X,Y)+\mathrm{FN}(X,Y)} \tag{9}$$

$$\mathrm{F}_1(X,Y) = \frac{\mathrm{Pre}(X,Y)\mathrm{Rec}(X,Y)}{(\mathrm{Pre}(X,Y)+\mathrm{Rec}(X,Y))} \tag{10}$$

Precision measures, from all pairings in $X$ (both true and false pairings according to $Y$), what proportion of the pairings were correct pairings. Intuitively precision captures how likely a given cluster would be to only contain items from a single event. Serving as a counterpart, recall captures what proportion of the true pairings in $Y$ were correctly paired together in $X$. Intuitively, recall captures the likelihood of items being placed in the same cluster when they should have been. The $\mathrm{F}_1$ metric is the harmonic mean of these two factors, measuring a trade off between the two.

Although both these evaluation metrics have intuitive and theoretical backing, De Vries et al [5] showed that clusters with high $\mathrm{F}_1$ or NMI could still be poor representations of clusters for a given underlying use-case. They recommended a final divergence from random metric which could be used with any given cluster evaluation technique. The divergence from random score is calculated by computing a score for a given $X$ and comparing it to the score achieved for $X' = \{x'_1,\ldots,x'_K\}$ where $|X| = |X'|$ and $|x_k| = |x'_k|$ for all $k$, but the actual document assignment across $X'$ is randomised. If the resulting score of $X'$ is similar to that of $X$, or equivalently if $div_{\mathrm{score}} = score(X,Y) - score(X',Y)$ is small, then it can be better argued that a given cluster assignment $X$ is actually no better than random assignment and therefore poor.

Experimentally we found both NMI and $div_{\mathrm{NMI}}$ metrics were high and varied little in various feature weighting and cluster parameter configurations. The $\mathrm{F}_1$ metric however differed significantly from the divergence $div_{\mathrm{F}_1}$, demonstrating a high overall difference when cluster quality was low upon inspection. Therefore, for the selection of feature weightings, the tuning of parameters and the final experiments (as discussed in the following sections), we show the $\mathrm{F}_1$, $div_{\mathrm{F}_1}$ and NMI metrics.

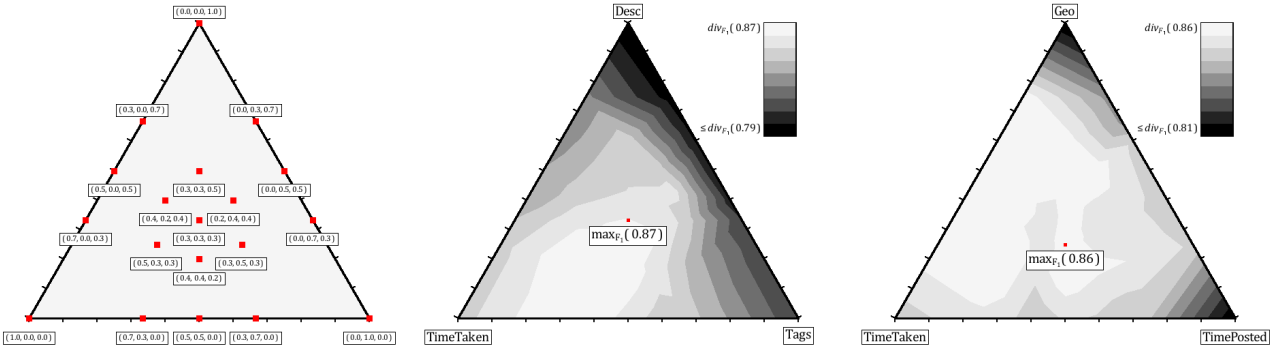
### 4.2 Feature Weightings and Clustering Parameters

In Section 3 we introduced the notion of feature weightings and cluster algorithm parameters. Here we discuss a search strategy for selecting values for those weightings and parameters which are optimal for the social event detection task.

For feature weightings, once the affinity matrices for each modality are constructed (see Section 3), our goal becomes the calculation of a combined affinity matrix $W$ whose cells $w_{ij}$ represent the fused similarity of the $i^{th}$ image with the $j^{th}$ image. This is achieved by calculating a weighted sum of the feature affinities $W^{(f)}$ whose cells $w_{ij}^{(f)}$ hold the affinity of two images for all features, $f \in F_{ij}$, where $F_{ij}$ represents all features $f$ which have some non-zero value for both images $i$ and $j$. Therefore we calculate the combined affinity using weighted sum fusion:

$$w_{ij} = \sum_{f}^{F} p_f w_{ij}^{(f)}, \quad \sum_{f}^{F} p_f = 1 \tag{11}$$

where $p_f$ is the weight of a given feature $f$. The final affinity matrices produced by this process are used by the clustering techniques discussed in Section 3.2. Optimal values of $p_f$ were found using a search across the simplex of features. More concretely, we iterate over points on a regular grid as applied to an $|F|$-dimensional hypercube with 4 divisions per axis. We treat each component of the coordinates of each grid point on this hypercube as the un-normalised values for each weighting $p_f$. Each component of the point on the hypercube grid is then normalised such that that Equation 11 holds (obviously certain combinations of un-normalised values will result in the same normalised values; we just ignore these repeated combinations). In the case where $|F| = 3$ (i.e. when there are 3 features) the $p_f$ grid




(a) The 2 simplex search points (b) Time taken vs Tags vs Descriptions (c) Time Taken vs Geo vs Time Posted

Fig. 3: Grid search for learning feature weightings. (a) shows the search points on a 2-simplex (3 features). (b) and (c) show the $div_{\mathrm{F}_1}$ scores when 3 features are weighted differently and all other features are weighted as 0. The scores are averaged across random sets of 5000 training images.

points searched can be seen in Figure 3a. It should be noted that this search approach performs a denser sampling of weight combinations in the middle of the space (where weights are more evenly distributed across all features); we chose this strategy because we believed that a balance of features would likely help the clustering. However, the feature weighting learned for our experiments (described in the next section) indicate that this is not the case for all features, and that the bias towards the centre of the weighting-space might well be sub-optimal.

To find the best $p_f$ weighting combination, the following procedure was followed. For each grid point on the weightings hypercube, a combined affinity matrix was constructed for a randomly selected but contiguous range of 5000 data items from the ReSEED training set of 300,000 data items. This limited number of items was chosen to make the parameter search task tractable, though a similar strategy could be followed using incremental clustering to find parameters. A non-optimal configuration of the DBSCAN clustering technique was then applied and the clusters generated were scored using the $div_{\mathrm{F}_1}$. Further, for each search point, this procedure was repeated 10 times, each time selecting a different random range of 5000 data items. The average $div_{\mathrm{F}_1}$ across these random sets was used as the score for a particular weighting combination. Example $div_{\mathrm{F}_1}$ distributions across sets of 3 features can be seen in Figure 3b and Figure 3c. From these figures we can see that the space of scores based on feature weightings is non-convex.

To find optimal cluster parameters, a given feature weighting is first selected. For that feature weighting, using the affinity matrix constructed for a set of 5000 items, a linear search was applied to each of the cluster parameters which needed to be optimised for a particular experiment. This is repeated 10 times. The limits and step for the linear search for each cluster parameter is detailed in Table 1.

To reflect the non-linearity between, and within, feature weightings and cluster parameters the final configurations used for the experiments on the test set were chosen in two ways. The first approach optimised the cluster parameters using feature weightings derived from the average weightings of the top 1000 (in terms of $div_{\mathrm{F}_1}$) feature combinations. These are the "average-weight" experiments presented in Table 2. The second approach optimised the cluster parameters for each of the top 1000 best feature combinations, selecting the best weighting and cluster parameter pair. These are the "best-weight" results in Table 2. These results are discussed further in the next section.

## 5 Experimental Results

Our experiments are performed using the ReSEED dataset. This allows comparative evaluation against other state-of-the-art techniques. ReSEED is a good dataset to test with because it contains a sample of a range of

Table 1: The ranges of linear search for each cluster parameter

| Parameter | start | stop | step |
|---|---|---|---|
| DBSCAN $\epsilon$ | 0.3 | 0.55 | 0.05 |
| DBSCAN $minPts$ | 1 | 5 | 1 |
| SPECTRAL $eigenthresh$ | 0.6 | 0.75 | 0.05 |
| SPECTRAL COS-DBSCAN $\epsilon$ | -1 | -0.4 | 0.1 |
| SPECTRAL COS-DBSCAN $minPts$ | 1 | 5 | 1 |

highly diverse event types, which helps ensure that the techniques being evaluated have general applicability to real-world data. As with any real-world dataset, we have to be aware that the data is highly noisy and that the ground-truth does have errors. Also, we have to be aware that the ReSEED dataset in particular has a sampling bias as only photos belonging to events are included (although this is actually not such a problem, because there is a high proportion of single-item events). This is discussed in more detail in Section 7.

Using the search scheme described in the previous section we generated 4 parameter configurations which were applied to the ReSEED test set. The *average-weight* feature weighting was $\{w^{(\text{taken})} = 0.257, w^{(\text{posted})} = 0.217, w^{(\text{geo})} = 0.183, w^{(\text{desc})} = 0.089, w^{(\text{tags})} = 0.207, w^{(\text{title})} = 0.046\}$. For DBSCAN (see Section 3.2.1) the average weight configuration used an $\epsilon = 0.45$ and a $minPts = 3$ while for spectral clustering (see Section 3.2.2) the configuration used $eigenthresh = 0.7$ and the cosine-similarity DBSCAN applied to the spectral clustering data used $\epsilon = -0.65$ and $minPts = 3$. The *best-weight* feature weighting was $\{w^{(\text{taken})} = 0.375, w^{(\text{posted})} = 0.0, w^{(\text{geo})} = 0.125, w^{(\text{desc})} = 0.125, w^{(\text{tags})} = 0.375, w^{(\text{title})} = 0.0\}$. In this *best-weight* configuration DBSCAN used $\epsilon = 0.5$ and a $minPts = 3$ and spectral clustering used $eigenthresh = 0.75$ and cosine-similarity DBSCAN with $\epsilon = -0.6$ and $minPts = 3$. All our submitted configurations used the incremental clustering technique discussed in Section 3.3 with a window increment size of 3000. The results of these configurations is presented in Table 2 and compared with all the other MediaEval SED 2013 participants. We show better results than all participants and a substantially better result than most.

Table 2: Official results from MediaEval 2013 SED task using the ReSEED dataset. Our 4 submitted runs are presented last and the names are highlighted in bold. The best result for each cluster quality metric over the runs is also highlighted.

| **Group/Technique** | $\mathbf{F}_1$ | **NMI** | $div_{\mathbf{F}_1}$ |
|---|---|---|---|
| CERTH-ITI(1) [16] | 0.5698 | 0.8743 | 0.5049 |
| CERTH-ITI(2) [23] | 0.7031 | 0.9131 | 0.6367 |
| UPC [10] | 0.8833 | 0.9731 | 0.8316 |
| UNITN [13] | 0.9320 | 0.9849 | 0.8793 |
| TUWIEN [32] | 0.78 | 0.94 | - |
| ADMRG [26] | 0.812 | 0.954 | 0.758 |
| ISMLL [28] | 0.8784 | 0.9655 | - |
| NTUA [14] | 0.2364 | 0.6644 | - |
| VIT [7] | 0.1426 | 0.1802 | 0.0724 |
| QM [2] | 0.78 | 0.94 | - |
| **DBSCAN (best-weight)** | 0.9454 | 0.9851 | **0.8865** |
| **Spectral (best-weight)** | 0.9114 | 0.9765 | 0.8534 |
| **DBSCAN (average-weight)** | **0.9461** | **0.9852** | 0.8864 |
| **Spectral (average-weight)** | 0.9024 | 0.9737 | 0.8455 |

## 6 Discussion

The results shown in Table 2 illustrate that our technique is very strong when applied to the ReSEED dataset. The performance is also very consistent across the three cluster metrics presented; many of the other techniques seem to have reasonable NMI scores, but relatively poor $F_1$ and $div_{F_1}$ scores. The Spectral clustering algorithm has a strong theoretical grounding, however, our results indicate that DBSCAN outperforms the Spectral clustering by a few percent across all metrics. The overall high scores for DBSCAN indicate that the feature similarities must by themselves provide a high amount of separability between items belonging to different events. It is difficult to justify the reduction in performance for the Spectral approach, however, it could simply be down to parameter choice, which is made harder by the additional parameters that the Spectral clustering algorithm introduces over plain DBSCAN.

In terms of the features used in our approach, by looking at the performance of the different feature weighting combinations we can make the following observations:

- Time taken appears to be the most important feature.
- The time posted and geographical features seem to hold a lot of the same information.
- The tags hold more useful information for clustering than the titles and descriptions.

In terms of overall performance, it is instructive to look beyond the NMI (and other metrics), and dig a little deeper into the how the clusters extracted by our approach differ from the ground-truth. Table 3 shows the key first order statistics of the clusters from our approach (specifically the *DBSCAN (average-weight)* configuration) and the ground truth. Our approach clearly tends to overestimate the number of clusters in the data and creates almost 4 times as many single-item clusters than the ground truth would suggest should exist. This is confirmed by looking at Figure 4, which shows the distribution of cluster sizes for clusters with 1 through 50 items. We hypothesise that the biggest single reason for the large number of single-item clusters is a lack of overlap between features. In-particular, analysis shows that a large number of these items have no tags — of the items assigned to a single cluster, only 36.0% have a tag, whereas the for complete dataset 95.6% have tags.

The plots in Figure 4 also show that our technique completely fails to create any clusters with exactly two

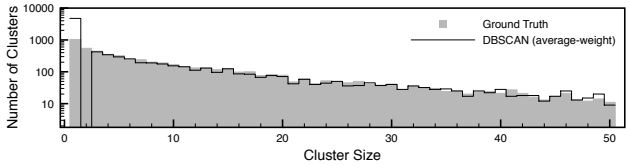



Fig. 4: Histograms of the distribution of cluster sizes for clusters of between 1 and 50 items for both the ground-truth and our approach. Note the logarithmic scale on the y-axis.

items; this is purely an artefact of the parameters of the DBSCAN algorithm (specifically $minPts$) and similarity measures used to create the sparse affinity matrix. Beyond clusters with two items, the overall distribution of cluster sizes between our approach and the ground-truth is similar; although not shown in the figure, the tail-ends of the distribution are also similar, right up to the largest cluster (which as shown in Table 3 has a remarkably similar number of items).

### 6.1 Computational complexity

DBSCAN is much faster than spectral clustering; in our experiments in the previous section running DBSCAN with precomputed affinities using the incremental approach resulted in run times of little over 2 minutes. Spectral clustering (implemented using ARPACK) on the other-hand took several hours. As mentioned previously, our DBSCAN implementation can scale to the size of the ReSEED test set, and so it is possible to compare the non-incremental DBSCAN against the incremental version. Interestingly, the non-incremental version is actually slightly faster (around 1m40s) in this case, although the performance (F1=0.945; NMI=0.985; $div_{F_1}$=0.887 for the *average-weight* configuration) is not significantly different. The difference in speed is due to the additional overheads of the incremental clustering relative to the (very fast) clustering algorithm. With larger datasets this would swing to favour incremental clustering, especially if the affinity matrix becomes so large that it could not be held in memory. Affinity matrix construction is faster with the incremental approach as fewer items have to be indexed and compared.

For spectral clustering, the time and space requirements of the algorithm meant that it was infeasible to cluster the dataset in a reasonable amount of time, and that the incremental approach was the only option. The insignificant difference in clustering performance between the incremental and non-incremental DBSCAN algorithms is a strong additional indicator that our assumptions about the clustered temporal nature of uploads belonging to an event is reasonable.

Table 3: Comparison of clusters formed using our technique (DBSCAN (average-weight)) against the ReSEED ground-truth.

| | **Ground Truth** | **Our approach** |
|---|---|---|
| No. clusters | 6287 | 9215 |
| Min. cluster size | 1 | 1 |
| Max. cluster size | 1409 | 1454 |
| Mean cluster size | 20.87 | 14.24 |
| Median cluster size | 7 | 1 |
| No. 1-item clusters | 1036 | 4748 |

### 6.2 Comparisons with other similar techniques

Of the work outside of MediaEval 2013, the technique by Becker et al [1] follows a methodology closest to the one described in this work. A key difference exists in the formulations of the feature distance metrics used. Their feature distances are more direct comparisons, whereas ours try to highlight the importance of graph sparsity by encoding quick dropoffs in the feature distance functions. Also, while both the clustering techniques described are incremental, ours re-performs clustering on overlapping blocks while forgetting previously discovered clusters, never allowing additions to be made to them if they are considered stable. Our explicit forgetting of completed clusters matches the nature of multimedia social streams in that it is common for an event to have items generated which relate to it over some set period of time, and then never have items posted to it again. By not holding previously clustered items at all, whether in their raw form or in an aggregated form, our technique exploits an efficiency which is unattainable if previous clusters are held and checked for every future multimedia item in the stream. In terms of performance, Becker et al report NMI scores of up to 0.94 on their test data (which is different from ReSEED, but has many similarities). We cannot draw any further conclusions about relative performance to our technique due to the different datasets and because, as shown in Table 2, a high NMI does not necessarily indicate effectiveness with respect to other cluster metrics, nor does it necessarily indicate a good clustering [5].

Within MediaEval 2013, the approach by Schinas et al [23] has many similarities to our own, although it does not take into account the incremental/streaming nature of the problem. The initial indexing stage has some similarities with our Lucene index step described in Section 3.1, and the resultant graph of image adjacencies is conceptually similar to our aggregated affinity matrix (see Section 3.1.1). The overall performance

of this approach is quite a lot worse than ours, especially with respect to the $F_1$ and $div_{F_1}$ metrics. Possible reasons for this could include poor choice of feature weightings and the lack of consideration of the temporal nature of the data.

Although our approach achieves the best score against all metrics, the approach presented by Nguyen et al [13] shows results that are competitive. The relative closeness of the scores to our approach indicates that high quality social event detection on the ReSEED dataset can indeed be achieved in many different ways. We do however believe that our attention to the streaming nature of the problem gives us an advantage.

## 7 Conclusions and Future Work

We have presented and evaluated an end-to-end system for multi-modal social event clustering of social multimedia streams. By posing the problem as a streaming one, and developing an incremental approach to clustering, coupled with custom feature comparison metrics and weighted feature-fusion techniques, we are able to detect events in a dataset of Flickr images in a manner that is both highly accurate and scalable. Our results suggest that the weighting of features plays an important role in achieving high performance. To this end, exploring non-grid-search based schemes for optimal feature weighting detection would be an interesting future avenue of research. Beyond this, a feature weight selection scheme which could be learnt online might also be able to better deal with the potential heteroscedasticity present in social media streams. Another possible avenue of research would be to consider more advanced affinity aggregation techniques (e.g. [3]).

### 7.1 Is event detection solved?

Evaluation of our technique using the ReSEED dataset indicates that we are very close to reaching a perfect segmentation of events, with the mutual information clustering measure reaching over 98%. However, we must be wary of over-fitting and also be aware that the dataset is likely to contain some mistakes as a result of the original human-provided event annotations that were used to construct it. The nature of the ReSEED dataset represents a scenario where every social media item was relevant, and every item belonged to one and only one event. This represents a significant bias in the dataset and is an important indicator that there is still further work to be done. An important next step in future work should explore other incarnations of social event detection, including query-led approaches where only certain items in the stream are relevant, and hierarchical approaches where individual social media items might belong to multiple events at different granularities of time and space. To do this, we need to consider how a dataset exhibiting these features might be constructed that will allow for effective comparative experimentation.

### 7.2 What about visual features?

Our brief experiments with SIFT features in the early phases of this work was largely unsuccessful and led to us concentrating on features extracted from the metadata. From our results it is clear that the metadata features are particularly powerful in the context of social event detection, however, this isn't a good reason not to explore visual features further in the future. In particular, we believe that visual features could be used to help in situations where the metadata is poor or sparse (such as for images with zero tags). Our particular embodiment of affinity graph construction using hashed SIFT features undoubtedly worked well for cases where there were near identical images belonging to an event. However, it would also very easily confuse images as belonging to the same event if they contained a visual similarity characterised by a few matching SIFT features (regardless of their spatial arrangement). The key to using visual features is likely to be to use (combinations of) features that are suitable for the task. In particular, we should consider image similarity at a number of levels: global similarity (for example using GIST features or a multiscale spatial pyramid of dense features like PHOW) might help capture the context of an event; local features (like SIFT) could identify (with suitable spatial constraints) common objects in the scene; and facial descriptors and similarity metrics could identify common individuals across images. Even with this layered approach to visual features it is likely that these features would have to be carefully moderated with respect to the available metadata, and that a much more dynamic technique (than weighted sum fusion) for combining the affinity matrices would be required.

## Acknowledgments

The authors would like to acknowledge the financial support of the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreements 270239 (ARCOMEM) and 287863 (TrendMiner). The effort of the 2013 MediaEval SED challenge organisers in creating the ReSEED social event dataset used in this paper is also acknowledged.

## References


1. Becker H, Naaman M, Gravano L (2010) Learning similarity metrics for event identification in social media. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, ACM, WSDM '10, pp 291–300
2. Brenner M, Izquierdo E (2013) MediaEval 2013: Social Event Detection, Retrieval and Classification in Collaborative Photo Collections. In: [9]
3. Chuang YY (2012) Affinity aggregation for spectral clustering. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Washington, DC, USA, CVPR '12, pp 773–780, URL http://dl.acm.org/citation.cfm?id=2354409.2355074
4. Chung F (1997) Spectral Graph Theory, vol 92. Amer Mathematical Society
5. De Vries CM, Geva S, Trotman A (2012) Document clustering evaluation: Divergence from a random baseline. CoRR
6. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDDM, AAAI Press, pp 226–231
7. Gupta I, Gautam K, Chandramouli K (2013) VIT@MediaEval 2013 Social Event Detection Task: Semantic Structuring of Complementary Information for Clustering Events. In: [9]
8. Hare JS, Samangooei S, Dupplaw DP, Lewis PH (2013) Twitter's visual pulse. In: ICMR'13, ACM, pp 297–298
9. Larson M, Anguera X, Reuter T, Jones GJ, Ionescu B, Schedl M, Piatrik T, Hauff C, Soleymani M (eds) (2013) Working Notes Proceedings of the MediaEval 2013 Workshop
10. Manchon-Vizuete D, Giro-I-Nieto X (2013) UPC at MediaEval 2013 Social Event Detection Task. In: [9]
11. Mayurathan B, Pinidiyaarachchi U, Niranjan M (2013) Compact codebook design for visual scene recognition by sequential input space carving. In: MLSP, pp 1–6
12. Ng A, Jordan M, Weiss Y, et al (2002) On spectral clustering: Analysis and an algorithm. NIPS 2:849–856
13. Nguyen TVT, Dao MS, Mattivi R, Sansone E, Natale FGD, Boato G (2013) Event Clustering and Classification from Social Media: Watershed-based and Kernel Methods. In: [9]
14. Papaoikonomou A, Konstantinos Tserpes MK, Varvarigou T (2013) A Similarity-based Chinese Restaurant Process for Social Event Detection. In: [9]
15. Petkos G, Papadopoulos S, Kompatsiaris Y (2012) Social event detection using multimodal clustering and integrating supervisory signals. In: Proc. ICMR
16. Rafailidis D, Semertzidis T, Lazaridis M, Strintzis MG, Daras P (2013) A Data-Driven Approach for Social Event Detection. In: [9]
17. Reuter T, Cimiano P (2012) Event-based classification of social media streams. In: In Proc. ICMR
18. Reuter T, Papadopoulos S, Mezaris V, Cimiano P, de Vries C, Geva S (2013) Social Event Detection at MediaEval 2013: Challenges, datasets, and evaluation. In: MediaEval 2013 Workshop
19. Reuter T, Papadopoulos S, Mezaris V, Cimiano P (2014) Reseed: Social event detection dataset. In: Proceedings of the 5th ACM Multimedia Systems Conference, ACM, New York, NY, USA, MMSys '14, pp 35–40, DOI 10.1145/2557642.2563674, URL http://doi.acm.org/10.1145/2557642.2563674
20. Risse T, Peters W (2012) Arcomem: From collect-all archives to community memories. In: Proceedings of the 21st International Conference Companion on World Wide Web, ACM, New York, NY, USA, WWW '12 Companion, pp 275–278, DOI 10.1145/2187980.2188027, URL http://doi.acm.org/10.1145/2187980.2188027
21. Samangooei S, Hare J, Dupplaw D, Niranjan M, Gibbins N, Lewis P, Davies J, Jai N, Preston J (2013) Social Event Detection Via Sparse Multimodal Feature Selection and Incremental Density Based Clustering. In: [9]
22. Scherp A, Jain R, Kankanhalli M, Mezaris V (2010) Modeling, detecting, and processing events in multimedia. In: Proceedings of the International Conference on Multimedia, ACM, MM '10, pp 1739–1740
23. Schinas M, Mantziou E, Papadopoulos S, Petkos G, Kompatsiaris Y (2013) CERTH @ MediaEval 2013 Social Event Detection Task. In: [9]
24. Shi J, Malik J (2000) Normalized cuts and image segmentation. PAMI 22(8):888–905
25. Smyth S, White S (2005) A spectral clustering approach to finding communities in graphs. In: Proceedings of the 5th SIAM International Conference on Data Mining, pp 76–84
26. Sutanto T, Nayak R (2013) ADMRG @ MediaEval 2013 Social Event Detection. In: [9]
27. Von Luxburg U (2007) A tutorial on spectral clustering. Statistics and computing 17(4):395–416
28. Wistuba M, Schmidt-Thieme L (2013) Supervised Clustering of Social Media Streams. In: [9]
29. Xu X, Yuruk N, Feng Z, Schweiger TAJ (2007) Scan: A structural clustering algorithm for net-

works. In: Proc. SIGKDD, ACM, KDD '07, pp 824–833

30. Zaharieva M, Zeppelzauer M, Breiteneder C (2013) Automated social event detection in large photo collections. In: Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval, ACM, ICMR '13, pp 167–174
31. Zandbergen PA, Barbeau SJ (2011) Positional accuracy of assisted gps data from high-sensitivity gps-enabled mobile phones. The Journal of Navigation 64:381–399, DOI 10.1017/S0373463311000051, URL `http://journals.cambridge.org/article_S0373463311000051`
32. Zeppelzauer M, Zaharieva M, Fabro MD (2013) Unsupervised Clustering of Social Events. In: [9]