Clustering Microtext Streams for Event Identification

Jie Yin
Computational Informatics
CSIRO, Australia
Jie.Yin@csiro.au

Abstract

The popularity of microblogging systems has resulted in a new form of Web data – microtext – which is very different from conventional well-written text. Microtext often has the characteristics of informality, brevity, and varied grammar, which poses new challenges in applying traditional clustering algorithms to analyze microtext. In this paper, we propose a novel two-phase approach for clustering streaming microtext, in particular Twitter messages, into event-based clusters. In the online phase, an incremental process is applied to discover base clusters and maintain detailed summary statistics. Upon demand for any user-specified time horizons, an offline phase is triggered to merge related clusters together. We demonstrate that our proposed approach can achieve better clustering accuracy than state-of-the-art methods.

Introduction

Microtext is a newly emerging type of Web data which is generated in enormous volumes with the proliferation of online microblogging systems. These systems, such as Twitter and Facebook, provide a light-weight, easy form of communication that enables individuals around the globe to share information and express their opinions in fluid and less formal ways. Microtext streams generated from these sites offer a rich source of real-time information about a wide variety of real-world events, ranging from planned occurrences such as political campaigns or sports games, to unexpected incidents such as earthquakes or terrorist riots. To provide insight into user-generated content broadcast in microtext streams, clustering approaches have demonstrated great potential for identifying what topics people are talking about and tracking how events unfold over time.

Clustering microtext streams poses a number of new challenges, due to short, noisy and informal nature of microtext [Ellen, 2011]. First, clustering techniques should be scalable to the sheer volume of data generated in microblogging systems. Twitter, for example, generates over 400 million tweets per day in early 2013. Thus, it is crucial to develop efficient clustering algorithms that can handle such massive amounts of streaming data. Second, microtext often has the characteristic of informality, brevity, varied grammar, and free-style. Depending on various personal style or background knowledge, people tend to use different words to convey the same or similar meanings, when writing about a particular event. Therefore, it is highly desirable to design effective clustering algorithms that can discover event-based clusters over time.

To cope with the sparsity and brevity of microtext, different methods have been proposed for microtext clustering in recent years. The majority of previous work has primarily focused on clustering a static collection of short documents [Rangrej et al., 2011, Tsur et al., 2012], or on using surface features to compute pairwise similarity between microtext [Reuter et al., 2011, Li et al., 2012]. However, the challenge of how to effectively cluster microtext in dynamic data streams has not been well addressed.

In this paper, we propose a novel framework for automatically grouping streaming microtext, in particular Twitter messages, into a set of event-based clusters; it intelligently divides the clustering process into an online component which maintains summary statistics, and an offline component which uses these compact statistics to discover event-based clusters. In the online phase, an incremental process is applied to discover base clusters and maintain detailed summary statistics about the clusters. This process can be efficiently

International Joint Conference on Natural Language Processing, pages 719–725,
Nagoya, Japan, 14-18 October 2013.

performed for the purpose of online social media monitoring. The generated base clusters serve as an intermediate statistical representation of the stream. Upon request, an offline phase is thereafter utilized to perform more computational analyses which merge similar clusters together in a bottom-up manner within a given time horizon. Experimental results show that our proposed clustering algorithm improve the clustering quality of other state-of-the-art approaches.

Related Work

This section reviews two primary related research areas: first, short text clustering which deals with very short and informal text; and second, studies that address event identification in social media.

Short Text Clustering

Although document clustering is well studied in the past decade, clustering very short, noisy and informal text has remained a challenging task. Rosa et al. [2010] studied the problem of clustering tweets into several pre-specified categories. They used hashtags as indicators of topics and argued that the clusters produced by traditional unsupervised methods can often be incoherent from a topical perspective. Rangrej et al. [2011] compared the performance of three document clustering techniques on Twitter data, and found that graph-based approach using affinity propagation performs best in clustering tweets. To cope with the sparsity of tweets, Tsur et al. [2012] constructed a virtual document by concatenating all micro-messages having the same hashtag, and then applied k-means algorithm to cluster virtual documents. Existing research has primarily focused on clustering a static collection of short text, while the challenge of continuously clustering microtext streams has not been well addressed.

Event Identification in Social Media

In recent years, identifying events from social media has attracted much attention. Petrović et al. [2012] applied a k-nearest neighbor approach to detect the first message talking about an event in a stream of Twitter messages, and used locality-sensitive hashing to speed up the computational process. Reuter et al. [2011] formulated the event identification problem as a record linkage task, in which a blocking strategy was used to reduce the number of pairs of documents considered for computing pairwise similarity. Becker et al. [2011] proposed an incremental clustering approach to group Twitter messages into clusters, which was similar to the method developed for detecting events in streams of text documents [Allan et al., 1998]. This approach determines the assignment of a message based on its similarity to textual centroids of existing clusters. Li et al. [2012] proposed to first detect bursty tweet segments as event segments and then use graph-based clustering to cluster event segments into events. Most of these works have either relied on computing pairwise similarity between static messages, or considered only the textual features of messages. In our work, however, we focus on developing an efficient framework for clustering a continuous stream of microtext, which groups clusters in a single pass and has the flexibility to merge clusters upon demand to identify event-based clusters.

Microtext Stream Clustering

We aim to design an effective microtext stream clustering algorithm that can meet three requirements: (1) The ability to handle massive volumes of microtext (*i.e.*, tweets) under the one-pass constraint of streaming scenarios; (2) The ability to employ temporal information in the clustering process, because tweets published within a certain time interval are more likely to correspond to the same event in the stream; (3) The ability to merge related clusters together when necessary. To meet these needs, we propose a new clustering framework which works in two phases, i.e., an online discovery phase and an offline cluster merging phase. The basic idea is to carefully balance the computational load between the online component and the offline component. In the online phase, the Twitter stream is processed in a single pass to maintain sufficient summary statistics about the evolving stream. The offline phase provides the flexibility for an analyst to perform queries about clusters and retrieve event-based clusters upon demand over different time horizons.

Below, we detail the two phases in the following two subsections.

Online Discovery Phase

The main task of the online phase is to provide a one scan algorithm over the incoming Twitter stream for identifying base clusters, with each cluster consisting of a set of similar tweets. For

this purpose, we design an efficient single-pass clustering algorithm which clusters the stream of tweets in an incremental manner.

To represent textual information of tweets, we employ a traditional vector-space model which uses the bag-of-words representation. A tweet is represented using a vector of words (terms or features), which are weighted using the term frequency (TF) and the inverse document frequency (IDF) [Salton and Buckley, 1988]. Using this model, a tweet represents a data point in d-dimensional space, $\mathbf{m}_i = (v_1, v_2, \ldots, v_d)$, where d is the size of the word vocabulary and v_j is the TF-IDF weight of j^{th} word in tweet m_i. However, in a dynamic microtext stream, word vocabulary changes and the number of tweets increases over time, making it computationally expensive to recalibrate the inverse document frequency of TF-IDF. Therefore, we resort to using term frequency as the term weight and adopting a sparse matrix representation of tweets to deal with dynamically changing vocabulary in our clustering algorithm.

To discover meaningful clusters, one important factor is defining an effective similarity measure. In our work, we use cosine similarity to measure textual similarity between two tweets, which is defined as

$$sim_{text}(\mathbf{m}_i, \mathbf{m}_j) = \frac{\mathbf{m}_i \cdot \mathbf{m}_j}{||\mathbf{m}_i|| \times ||\mathbf{m}_j||}, \quad (1)$$

where $\mathbf{m}_i \cdot \mathbf{m}_j$ indicates the dot product of vectors $\mathbf{m}_i$ and $\mathbf{m}_j$. Besides, $||\mathbf{m}_i||$ and $||\mathbf{m}_j||$ denotes the norm of vectors $\mathbf{m}_i$ and $\mathbf{m}_j$, respectively.

Since real-world events typically span a limited time interval, tweets that largely differ on their publication times are much less likely to belong to the same event. Therefore, in order to cluster tweets into temporally-related groups, we also exploit a time similarity measure defined as

$$sim_{time}(\mathbf{m}_i, \mathbf{m}_j) = \exp(-\frac{|t_{\mathbf{m}_i} - t_{\mathbf{m}_j}|}{\lambda}), \quad (2)$$

which is based inversely on the distance between tweets' publication dates/times. $|t_{\mathbf{m}_i} - t_{\mathbf{m}_j}|$ indicates the time difference between tweets $\mathbf{m}_i$ and $\mathbf{m}_j$, represented as the number of days, and λ is the number of days of one month, whose value is application dependent. In our case, if $t_{\mathbf{m}_i}$ and $t_{\mathbf{m}_j}$ are more than one month apart, we consider time similarity between $\mathbf{m}_i$ and $\mathbf{m}_j$ to be very small.

Putting together, our clustering algorithm uses a combined similarity measure defined as:

$$sim(\mathbf{m}_i, \mathbf{m}_j) = sim_{text}(\mathbf{m}_i, \mathbf{m}_j) \cdot sim_{time}(\mathbf{m}_i, \mathbf{m}_j). \quad (3)$$

This similarity measure not only captures the similarity between the texual vectors of tweets, but also penalizes the similarity between tweets if their publication dates/times are far away.

To maintain sufficient information about clusters, we represent each cluster C_i using a cluster feature vector $\psi(C_i)$, defined as follows:

- Textual centroid C_i^w: which is a vector in which each element represents the average weight of the corresponding words for all tweets in cluster C_i.
- Time centroid C_i^t: which is the average publication time of all tweets that form cluster C_i.
- Cluster size $|C_i|$: which is defined as the number of tweets belonging to cluster C_i.

Now we describe the process of the incremental clustering algorithm. Given a Twitter stream in which the tweets are sorted according to their published times, the algorithm takes the first tweet from the stream, and uses it to form a cluster. As a new tweet $\mathbf{m}$ arrives, we calculate the similarity between tweet $\mathbf{m}$ and any existing clusters C_i as

$$sim(\mathbf{m}, C_i) = sim_{text}(\mathbf{m}, C_i^w) \cdot sim_{time}(t_{\mathbf{m}}, C_i^t). \quad (4)$$

Let C be the cluster that has the maximum similarity with $\mathbf{m}$. If $sim(\mathbf{m}, C)$ is less than a similarity threshold δ_{sim}, which is to be determined empirically, a new cluster is created to include $\mathbf{m}$; Otherwise, the tweet $\mathbf{m}$ is assigned to the closest cluster C. By adjusting the threshold δ_{sim}, we can obtain clusters at different levels of granularity. Once a new tweet $\mathbf{m}$ is added to cluster C_i, we update the corresponding cluster representatives $\psi(C_i)$ using the following equations:

$$\hat{C}_i^w = \frac{C_i^w \times |C_i| + \mathbf{m}}{|C_i| + 1}, \quad (5)$$

$$\hat{C}_i^t = \frac{C_i^t \times |C_i| + t_{\mathbf{m}}}{|C_i| + 1}, \quad (6)$$

$$|\hat{C}_i| = |C_i| + 1. \quad (7)$$

This incremental algorithm is efficient as it considers each tweet at once, and can thus scale to a growing amount of tweets. To further improve efficiency, we maintain a list of active clusters over

time in the online phase. If no more tweets are added to a cluster for a period of time, which is determined based on application needs, the cluster is considered inactive and it is removed from the active list. The algorithm considers only those clusters in the active list as candidates to which a new tweet can be added. The output of the algorithm is a list of clusters $C_1, \ldots, C_H$, together with their cluster representatives $\psi(C_1), \ldots, \psi(C_H)$.

Offline Cluster Merging Phase

The base clusters generated by the online phase serve as an intermediate statistical representation, which can be maintained in an efficient way even for a large volume of tweets. The subsequent offline phase is utilized to merge a list of clusters into event-based clusters. There is no need to process the voluminous microtext stream, but the compactly stored summary statistics of clusters.

For a particular event, since users tend to convey the same or a similar meaning using different words depending upon their own personal style, the online phase would organize the tweets that report the same event, but expressed using different words, into different base clusters. Therefore, we propose to merge together the clusters that are related with respect to the same event in the offline phase. Concretely, we calculate a cluster merge criterion, $link(C_i, C_j) = sim_{text}(C_i^w, C_j^w) \cdot sim_{time}(C_i^t, C_j^t)$, which captures the inter-similarity between two clusters C_i and C_j. The principle is to merge a pair of clusters that have a larger inter-cluster similarity. When two clusters are merged, we merge a smaller cluster into the larger one and in this way, larger clusters are retained which can better represent significant events of interest.

The offline clustering phase provides the flexibility to query the clustering results at any time horizon. Given a list of clusters generated during the online phase, we consider iteratively merging two clusters C_{j^*} and C_{i^*} such that $link(C_{i^*}, C_{j^*})$ is maximized. Accordingly, cluster representatives for cluster C_{i^*} are updated as follows:

$$\hat{C}_{i^*}^w = \frac{C_{i^*}^w \times |C_{i^*}| + C_{j^*}^w \times |C_{j^*}|}{|C_{i^*}| + |C_{j^*}|}, \quad (8)$$

$$\hat{C}_{i^*}^t = \frac{C_{i^*}^t \times |C_{i^*}| + C_{j^*}^t \times |C_{j^*}|}{|C_{i^*}| + |C_{j^*}|}, \quad (9)$$

$$|\hat{C_{i^*}}| = |C_{i^*}| + |C_{j^*}|. \quad (10)$$

To determine an optimal number of clusters, we use the notion of *separation* to measure the clustering quality, which is defined as the average inter-cluster similarity over all the clusters, that is, $S(k) = \frac{1}{N(N-1)} \sum_i \sum_j link(C_i, C_j)$, where $C_1, \ldots, C_N$ are the clusters obtained at step k. The smaller value this metric has, the better clusters are separated from each other. Based on this metric, we design a criterion to decide whether or not to stop the merging process. At each step k, given two candidate clusters to be merged, we compute a validation index as

$$\Delta_k = \frac{S(k+1) - S(k)}{S(k)}, \quad (11)$$

which represents the relative change in inter-cluster similarity after a merge is made. If $\Delta_k < 0$, that means a cluster merge can improve the separation of clusters. We thus proceed with merging the two clusters. Otherwise, if $\Delta_k \geq 0$, we stop the cluster merging process. In this way, the optimal number of clusters can be automatically determined during the cluster merging process.

Experiments

We carry out experiments to evaluate the effectiveness of our proposed algorithm, and compare its performance with other baseline methods.

Dataset

The dataset we used is an annotated corpus of tweets collected from the beginning of July 2011 to September 2011 [Petrović et al., 2012]. The corpus was distributed as a set of tweet IDs, together with their annotations. We re-retrieved the tweets using Twitter search API[1] and obtained a set of 2,633 tweets. Each tweet was annotated as one out of 27 events, which cover a variety of real-world events, such as London riots, terrorist attacks in Norway, Earthquake in Virginia, and NASA's announcement about discovery of water on Mars. The annotations are used as the ground truth for evaluating the clustering algorithms.

We preprocessed the tweets by removing stopwords, user mentions (@username), and embedded links, because such elements in tweets may not be useful for indicating the topics. We compiled a list of stopwords that specifically suited Twitter content. It includes formal English stopwords such as *is*, *am*, informal English stopwords

[1] https://dev.twitter.com/docs/using-search

such as *gonna*, *arent*, and Twitter specific stopwords such as *RT* that indicates a retweet. We also performed a shallow lexical normalization on tweets and stemmed words using Porter Stemmer. For lexical normalization, we only considered words that were emphasized by repeating one or more letters. If a letter was repeated more than three times, it was normalized to one instance of that letter. For example, the word *crazyyyyy* was turned to *crazy*.

For our clustering task, we constructed a Twitter stream by sorting all tweets according to their publication times. The stream was taken as input to the clustering algorithms. For each tweet, we mainly used bag-of-words and specific hashtags (words preceded with a # sign) as features to construct a vector model.

Baselines

Our proposed algorithm is referred to as **MSC** (Microtext Stream Clustering). For comparison, we use two other methods as baselines:

- **IC**: which is a standard incremental clustering algorithm adopted by Becker et al. [2011]. It determines the assignment of a message solely based on its similarity to the textual centroids of existing clusters.
- **IC-Time**: which differs from our proposed algorithm in that it only uses the first online phase to discover clusters. By comparing with this baseline, we show how much gain in clustering quality can be achieved with the offline cluster merging.

In our experiments, we set parameter λ in Eq.(2) to be 30. In addition, we set the similarity threshold $\delta_{sim} = 0.2$ for all the algorithms.

Evaluation Metrics

Let $\mathcal{C} = \{C_1, \ldots, C_K\}$ denote the clustering result produced by one clustering algorithm, and $\mathcal{G} = \{G_1, \ldots, G_L\}$ denote the desired ground truth. We use two evaluation metrics: F-measure [Yin and Yang, 2005] and normalized mutual information (NMI) [Strehl and Ghosh, 2003], to validate the effectiveness of the clustering algorithms. we observe that the results are strongly correlated on the two metrics.

Experimental Results

We first performed experiments to evaluate the performance of three clustering algorithms on the entire stream. Since hashtags are considered as good indicators of topics in the tweets, we investigated two different ways of using hashtags as features: first, considering hashtags in the same way as words, and second, removing the # symbol and treating hashtags as normal words. Table 1 reports the clustering accuracy using the three algorithms on the two settings.

	F-measure	NMI
Hashtags		
IC	0.892	0.897
IC-Time	0.905	0.907
MSC	**0.958**	**0.955**
Hashtags without #		
IC	0.899	0.907
IC-Time	0.910	0.913
MSC	**0.966**	**0.962**

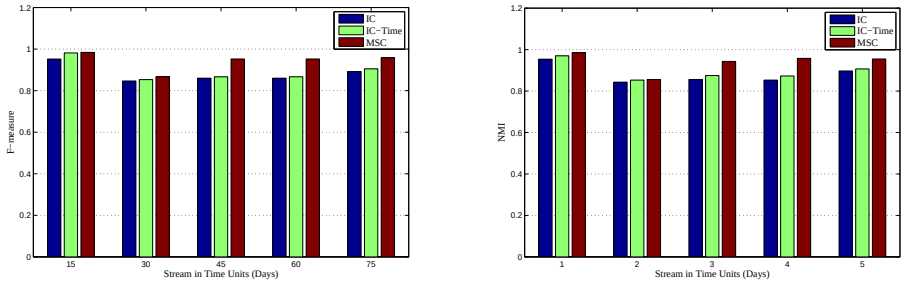
Table 1: Comparison of clustering algorithms on F-measure and NMI metrics

The top part of the table compares the performance of the three algorithms using bag-of-words and original hashtags as features. We can see that, our proposed MSC algorithm is superior to the other two baselines, while IC-time performs slightly better than IC. This is because, IC only relies on the cosine similarity between textual features of tweets to form clusters, while IC-Time enforces a time constraint in the similarity measure to reflect the time locality of events, which thus leads to better clustering accuracy. By explicitly merging related clusters, MSC achieves the highest accuracy on both two metrics.

The bottom part of the table shows the clustering results by removing the # symbol and treating hashtags as normal words. We can observe that, this improves the clustering accuracy for all three algorithms. We believe that this improvement is because removing the # symbol contributes to increasing the term frequency of the same topic word in the tweets. It thus translates to yielding better clustering accuracy. This can be illustrated using the examples as follows.

> *Bold move as Google Buys Motorola for* 12.5 *Billion, and paid cash #google #motorola.*

> *5.8 earthquake happened in Virginia just moments ago. #Earthquake #Virginia.*



(a) F-measure (b) NMI

Figure 1: Clustering accuracy over different time horizons

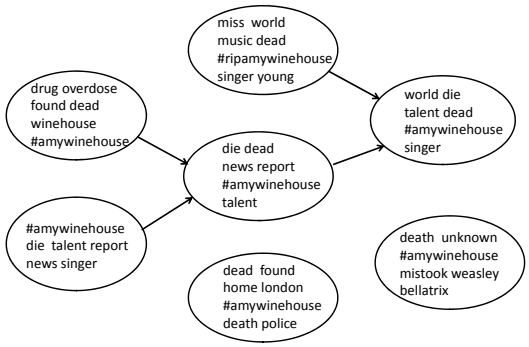


Figure 2: Illustration of the cluster merge process

If we remove the # symbol, hashtags *#google* and *#motorola* are turned into words *google* and *motorola*, in the first tweet, and *#Earthquake* and *#Virginia* are into *Earthquake* and *Virginia*, in the second tweet. In both cases, this increases the term frequencies of the topic words or main entities of events, thus highlighting their contributions to forming the clusters.

To better understand how our MSC algorithm performs cluster merges, Figure 2 illustrates the cluster merging process for the topic talking about the death of Amy Winehouse[2]. There are seven clusters generated from the online phase, each of which is represented using top-ranked keywords in the figure. In the offline phase, the clusters are merged based on their similarity and relatedness in a bottom-up manner, and finally three clusters remain after two rounds of cluster merges.

The other important feature of our proposed MSC algorithm is that it can merge related clusters upon demand for any user-specified horizon. Therefore, we carried out experiments to compare the clustering quality of the three algorithms at different time horizons. Figure 1 shows the clustering accuracy with respect to F-measure and NMI at different time units in the stream. We can see that, our proposed MSC algorithm consistently outperforms the other two baselines over time. This indicates that, MSC has the ability to retain sufficient statistics required for effective cluster merging in the offline phase.

[2] http://en.wikipedia.org/wiki/Amy_Winehouse

Conclusions and Future Work

In this paper, we proposed a new approach for clustering microtext streams into event-based clusters. Our proposed approach intelligently divides the clustering process into an online component which maintains summary statistics, and an offline component which uses these compact statistics to discover event-based clusters. Therefore, it has the advantage of processing and scaling to large volumes of microtext streams. Experiments and comparisons demonstrated that our proposed approach achieves better clustering accuracy than state-of-the-art methods, and merging similar clusters can improve the performance of short text clustering.

This work can be extended in several directions. We will further evaluate the effectiveness of our clustering algorithm in the ESA (Emergency Situation Awareness) system [Yin et al., 2012] in larger-scale datasets. In particular, we will test its performance together with the burst detection module for identifying significant event-based clusters from the real-time Twitter stream. Moreover, since short, informal microtext has high degree of lexical variations, we will explore paragraphing techniques to uncover hidden semantic relatedness between microtext. Such information can be leveraged to group clusters that talk about the same event, but expressed using different words, and thus improve the clustering quality.

References

J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–45, Melbourne, Australia, 1998.

H. Becker, M. Naaman, and L. Gravano. Beyond trending topics: Real-world event identification on Twitter. In *Proceedings of the Fifth International Conference on Weblogs and Social Media*, pages 438–441, Barcelona, Catalonia, Spain, 2011.

J. Ellen. All about microtext: A working definition and a survey of current microtext research within artificial intelligence and natural language processing. In *Proceedings of the Third International Conference on Agents and Artificial Intelligence*, pages 329–336, Rome, Italy, 2011.

C. Li, A. Sun, and A. Datta. Twevent: Segment-based event detection from tweets. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 155–164, Maui, HI, USA, 2012.

S. Petrović, M. Osborne, and V. Lavrenko. Using paraphrases for improving first story detection in news and Twitter. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–346, Montreal, Canada, 2012.

A. Rangrej, S. Kulkarni, and A.V. Tendulkar. Comparative study of clustering techniques for short text documents. In *Proceedings of the International World Wide Web Conference*, pages 111–112, Hyderabad, India, 2011.

T. Reuter, P. Cimiano, L. Drumond, K. Buza, and L. Schmidt-Thieme. Scalable event-based clustering of social media via record linkage techniques. In *Proceedings of the Fifth International Conference on Weblogs and Social Media*, pages 313–320, Barcelona, Catalonia, Spain, 2011.

K.D. Rosa, R. Shah, B. Lin, A. Gershman, and R. Frederking. Topical clustering of tweets. In *Proceedings of the SIGIR Workshop on Social Web Search and Mining*, Beijing, China, 2010.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.

O. Tsur, A. Littman, and A. Rappoport. Scalable multi stage clustering of tagged micromessages. In *Proceedings of the 21st World Wide Web Conference*, pages 621–622, Lyon, France, 2012.

J. Yin and Q. Yang. Integrating hidden Markov models and spectral analysis for sensory time series clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 506–513, Houston, Texas, USA, 2005.

J. Yin, S. Karimi, B. Robinson, and M. Cameron. Esa: Emergency situation awareness via microbloggers. In *Proceedings of the Twenty-First ACM International Conference on Information and Knowledge Management*, pages 2701–2703, Maui, Hawaii, USA, 2012.