Stream Clustering of Tweets

Sophie Baillargeon
Département de mathématiques
et de statistique
Université Laval
Québec (Québec), Canada G1V 0A6
Email: sophie.baillargeon@mat.ulaval.ca

Simon Hallé
Thales Canada
Research & Technology
Québec (Québec), Canada G1P 4P5
Email:
simon.halle@ca.thalesgroup.com

Christian Gagné
Département de génie électrique
et de génie informatique
Université Laval
Québec (Québec), Canada G1V 0A6
Email: christian.gagne@gel.ulaval.ca

***Abstract*—This paper proposes an approach to cluster social media posts. It aims at taking full advantage of this recent source of newsworthy information and at facilitating the work of users who need to monitor public events in real-time. The emphasis is on developing a stream clustering algorithm able to process incoming tweets. A first implementation of the algorithm, focusing on the tweets' text, was tuned and tested on a dataset of manually annotated messages. Results show that the algorithm produces a partition of tweets similar to the manual partition obtained from humans. In future work, we plan to extend this algorithm with additional features and integrate the resulting analytical capabilities to a real-time social media monitoring platform called CrowdStack.**

I. Introduction

Social media and news reporting platforms represent a valuable source of information to monitor events occurring around the world in real-time. Because of the size and speed at which this information is generated and the fact that this information is often produced in a free-text form, it can be an overwhelming challenge to fully benefit from such data sources.

In this paper, we propose a social media post stream clustering solution for addressing this problem. Our focus is on posts from the popular microblogging service Twitter, where we aim at facilitating the exploitation of tweets by grouping similar ones together.

The paper is organized as follows. We first briefly present a real-time monitoring and investigation platform of social media documents (posts) developed by Thales Canada. The motivation behind this project is to investigate how new analytical capabilities could be added to the latter platform. Then the proposed stream clustering algorithm is described, followed by a presentation of preliminary results obtained with the algorithm. The last section deals with future work.

II. Social media analytics platform

Thales Canada intends to exploit the stream clustering algorithm described in this paper through its social media analytic platform called CrowdStack™. This platform uses a big data processing framework to ingest and analyze large volumes of documents that are streamed in real-time. End users access the system through a Web interface (see Fig. 1) that provides various search and visualization features. Through this interface, they can monitor and compare situations evolving in real-time and investigate datasets of documents accumulated during a period of months or years of streaming. By ultimately adding the stream clustering algorithm to CrowdStack's document processing framework, users will have the ability to search and visualize documents by clusters, thus reducing the information they need to process, facilitating their comprehension and improving their search time. More importantly, they will be notified when large clusters of documents appear, thus providing a valuable alerting capability.

The work presented here is the beginning of a larger project in which we aspire to develop an algorithm for public event detection from social media posts that could be integrated to CrowdStack. Research in the field of event detection in social streams is quite recent (see [1], [2], and [3]).

A. Available Tweet Features

In this paper, we focused on analyzing tweets, which represent one of the best sources of information to cover broad subjects from all over the world in real-time. The CrowdStack API gives access to various features describing tweets, listed in Fig. 2. This figure includes metadata derived from the tweet's text using a Natural Language Processing (NLP) framework from the National Research Council of Canada (NRC).

Only the "text" feature is used in this paper, although we plan to generalize the approach to all available features in future work.

III. Stream Clustering Algorithm

We propose here a stream clustering algorithm to group similar tweets. In a stream of tweets, many tweets are in fact retweets of another tweet. Retweets of the same tweet should always belong to the same cluster. Therefore units to cluster are not individual tweets; instead they are groups of retweets.

Most tweets clustering algorithms we have seen in the literature process tweets, or groups of retweets, one after the other (see [2], [3], [4]). In our approach, we chose to process tweets in small batches formed over time, as illustrated in Fig. 3. By doing so, we wanted to facilitate the use of this algorithm inside the existing big data processing framework used by CrowdStack. Indeed, in this framework, data (tweet) streamed in real-time is immediately indexed into a database so it can be consumed by users in real-time. Afterward, this new data is processed by algorithms organized inside a batch layer to

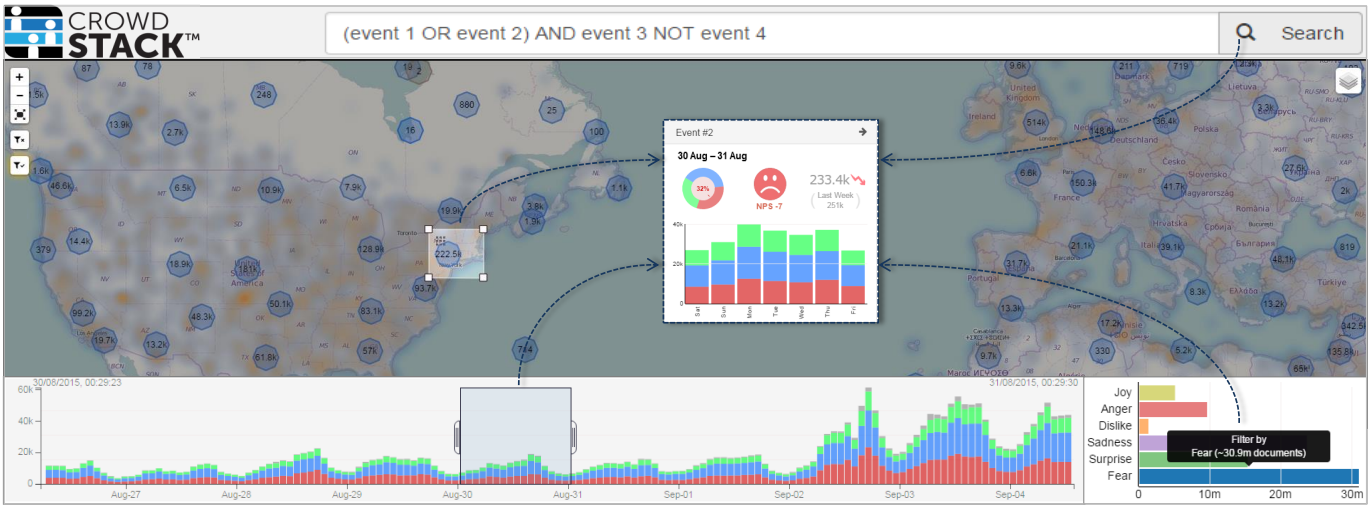

Fig. 1. Web widgets from the CrowdStack's user interface are used to search and filter particular subjects (top bar), locations (center map), time periods (bottom-left timeline), emotions (bottom-right widget) and much more. Customizable "cards", shown at the center of the figure as the result of multiple filters, illustrate the state of a subject of interest the user is following.

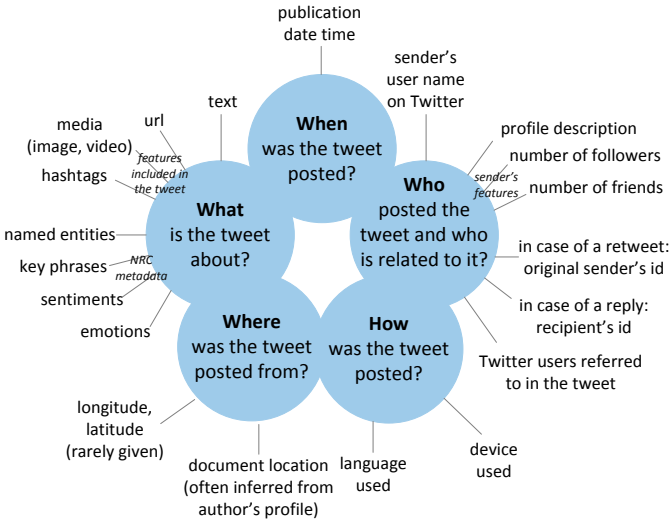

Fig. 2. Document features used to characterize tweets are categorized by the type of information they bring.

generate additional information and enhance this existing data after it has been streamed into the system. When the clustering algorithm will be integrated inside CrowdStack's batch layer, it will need to respect two non-functional requirements: 1) minimize the number of modifications to existing data used in CrowdStack's indexes; and 2) maximize its ability to be parallelized so it can run on multiple data nodes. Thus, by orchestrating the execution of the clustering algorithm using mini-batches, we: 1) minimize the number of modifications to data entries that define cluster groups in our databases; and 2) make it possible to parallelize the clustering algorithm, which could be considered as part of future works. In addition, the maximum batch size represents a configurable system parameter that allows us to achieve a trade-off between frequently updated cluster data (small batches) and good performance (large batches) of both the whole system and of specific system features relying on cluster data, e.g. alerts.

A classical clustering method is used on the first batch to form the initial set of clusters. We call this phase "new clustering" because it creates new clusters. The following batches are treated differently since clusters already exist and groups of retweets in the batch could just as well belong to pre-existing clusters as to new ones. The clustering algorithm for every batch except the first one therefore comprises two phases: a first in which groups of retweets are added to pre-existing clusters, and a second one in which new clusters are formed with groups of retweets that did not join a cluster in the first phase. We call the first phase "join clustering". A new clustering is performed in the second phase, as was done with the first batch. Here follows a description of the two clustering phases.

A. *New clustering*

Even if we process a data stream, the new clustering phase involves a classical batch clustering method. Since features to process are of various types, the method has to allow non-numeric features. Consequently, we restricted our choice of clustering algorithm to methods able to work only from a dissimilarity or distance matrix between units. We could not use, for example, the K-means method. We chose agglomerative hierarchical clustering for its simplicity (see [5, Section 14.3.12]).

Agglomerative hierarchical clustering starts by considering that each unit is in a distinct cluster. At each step of the calculations, the two nearest clusters are merged. The process continues until all units are in a single big cluster. In this way, a hierarchy of clusters is built, which can be graphically represented by a dendrogram. Fig. 4 shows a simple example

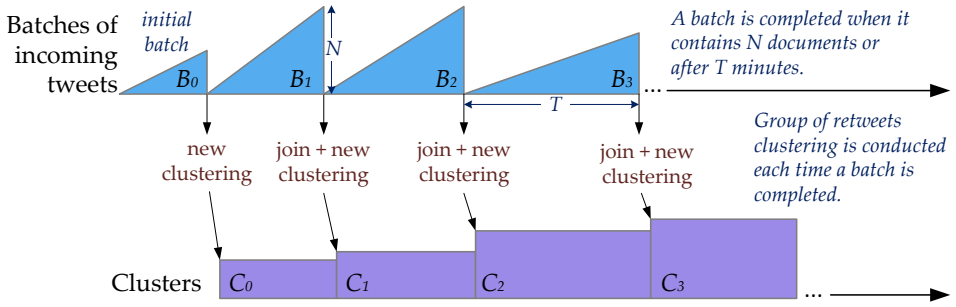

Fig. 3. Online mini-batch algorithm proposed to cluster groups of retweets: B_t is the batch containing tweets monitored during the t^{th} time period and C_t is the cluster set after clustering groups of retweets in B_t.

of a dendrogram, obtained from a distance matrix between 6 units.

To compute the distance between two clusters, we used the mean distances between units in those clusters, which is called "average link".

To obtain a partition of the units, one simply has to cut the dendrogram at a certain threshold height. This threshold is a tuning parameter of the method. We propose using annotated tweets in order to find the best value for this parameter.

K-medoids [5, Section 14.3.10] clustering would have been a possible alternative to hierarchical clustering here, because it can work from a distance matrix. However, choosing the number of clusters would have been a problem with that method. In the approach we propose, the choice of number of clusters is replaced by the choice of threshold parameter value.

B. Join clustering

The join clustering phase aims at deciding which groups of retweets will join a pre-existing cluster and which are not similar enough to pre-existing clusters to be added to one of them. Making this decision is simply a matter of calculating distances between groups of clusters in the processed batch and pre-existing clusters.

However, a newly formed batch almost always contains tweets belonging to groups of retweets already clustered, i.e. seen in previous batches. Therefore, we propose the following steps for the join clustering phase:

1) Identify the groups of tweets to cluster:
   B_t^* = subset of groups in B_t not already clustered.
2) For each group g_i^{*t} in B_t^*:
   a) identify the potential clusters:
   C_{t-1}^i = subset of clusters in C_{t-1} containing tweets posted close enough in time from the tweets in g_i^{*t};
   b) calculate distances between g_i^{*t} and clusters in C_{t-1}^i: group-cluster (or unit-cluster) distance = mean distance between group and the groups in the cluster (as in the hierarchical clustering);
   c) decide: add g_i^{*t} to its nearest cluster (if similar enough), or send g_i^{*t} to the new clustering phase.

Step 2 a) will be dealt with in future work. For now, let us focus on step 2 c). It involves a threshold parameter at which a group of retweets is declared too dissimilar to pre-existing clusters to join one of them. We chose the same definition of unit-cluster distance in the two phases so the threshold parameter has the exact same scale and meaning in the join and in the new clustering. We end up tuning only one parameter.

The join clustering phase is illustrated in Fig. 5 for groups of retweets in a fictive batch B_t. For every group g_i^t from B_t never seen before, distances are calculated between the group and clusters c_j^{t-1} in the set of clusters pre-existing C_{t-1} close enough in time from g_i^t. In this example, g_4^t has already been seen in a previous batch, so no distances are calculated for this group. Groups g_1^t, g_2^t and g_3^t have never been seen. However cluster c_2^{t-1} is discarded as a potential cluster for g_1^t, and so is c_3^{t-1} for g_2^t, because of tweet posting times too far apart. In the end, g_1^t is added to its nearest cluster c_1^{t-1}, so is g_3^t, also nearest to c_1^{t-1}. However, the smallest distance between g_2^t and a cluster in C_{t-1} is larger than the threshold distance (here 0.7) to join a pre-existing cluster. Therefore, g_2^t will have to be processed again, in the new clustering phase.

C. Clustering based on tweets' text

We began the implementation and test of our algorithm with the text feature. Following an exploratory analysis of annotated data and interviews with our annotators, we identified text as being the most useful feature for humans to group tweets together. To calculate distances between tweet texts, we chose to process as follows:

1) Text cleaning: we remove punctuation marks, common words in the language used (called stop words) and URLs; we also transform all letters to lower case.
2) Text to words: we split the texts into words.
3) Unit-unit distance: we calculate the Jaccard distance between pairs of texts, which are now sets of words.

IV. Preliminary results

We implemented the proposed stream clustering algorithm. This algorithm includes a configurable threshold parameter. To find the best value for such a tuning parameter, some choose to exploit manually annotated data [3], others base their choice

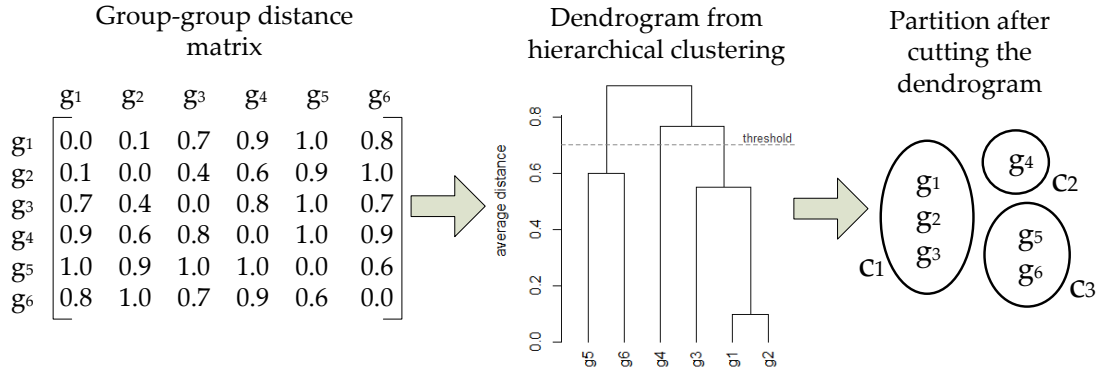

Fig. 4. Example of new clustering with 6 groups of retweets g_i, $i = 1, \ldots, 6$; where the dendrogram is cut at the threshold height of 0.7, resulting in a partition containing 3 clusters c_j, $j = 1, 2, 3$.

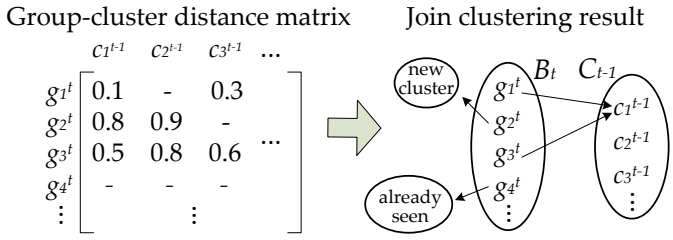

Fig. 5. Example of join clustering processing groups of tweets g_i^t, $i = 1, 2, \ldots$, in B_t. C_{t-1} is the set of pre-existing clusters c_j^{t-1}, $j = 1, 2, \ldots$.

on descriptive statistics [2]. We chose to use annotated data, even though properly annotating data requires time, because we believe it will lead to better performances.

A. Annotated dataset

Tweets monitored on August 30, 2015, Eastern Daylight Time (UTC-4), have been annotated. Annotating consisted in tagging tweets according to their subject. Tweets about the same subject were given the same tag. Since tags can be seen has cluster membership, tweets were clustered manually through these annotations.

The complete dataset includes 3102 tweets, divided into 1411 groups of retweets. 1169 of these groups include only 1 tweet, but other groups contain up to 152 tweets.

The dataset has been split into a validation (811 groups of retweets) and a test (600 groups of retweets) set. Note that we intend to annotate another dataset (on another time period), which could become an even more suitable test set. The validation set has been used to optimize the threshold parameters, whereas the test set has been used to evaluate the performance of the algorithm.

In addition to these utilities, the annotated dataset is of great value to better understand tweet features (see Fig. 2). By studying the variations within and between clusters in this dataset, we can get some insight into which features are the most relevant for the clustering of tweets.

B. Threshold parameter optimization

The tuning parameter common to the new and the join clustering phases of our algorithm is a threshold on the distance above which two clusters are too dissimilar to be merged. Here we search for the best threshold parameter value (i.e., the one giving the most meaningful partition).

The term "automatic partition" is used to refer to the result obtained from processing the tweets in the annotated dataset with our algorithm. The partition obtained from the annotation is called "manual partition". We aim at an automatic partition as similar as possible to the manual partition. The indexes used here to measure the similarity (or dissimilarity) between the two partitions are the Rand index, adjusted or not [6], and the variation of information distance [7].

Fig. 6 shows the results of the threshold parameter optimization. Parameter values between 0.7 and 0.92 (by leaps of 0.005) were considered. Indexes to compare automatic and manual partitions, as well as the number of clusters formed, are reported in this figure. The Rand index is maximized at a threshold value of 0.85, the adjusted Rand index is maximized at 0.87 and the variation of information (vi) distance is minimized at 0.84. We selected the value 0.87 since it optimizes the widely used adjusted Rand index and also because it leads to the number of relevant clusters closest to 69, which is the number of relevant clusters in the annotated dataset.

Clusters are considered irrelevant if the tweets they contain relate, for instance, to non-public issues or to entertainment. Such tweets are abundant in any stream of tweets. But since our ultimate objective is to detect public events, we are not interested in clustering tweets irrelevant for that goal. We rather wish to filter them out.

C. Algorithm performance

With the chosen threshold value of 0.87, we processed the test set with our algorithm to obtain an automatic partition. Table I shows a comparison of this partition with the manual partition of the test set, considered to be the truth.

Rand index and specificity are excellent whereas sensitivity is acceptable. However, we obtain a poor false discovery rate.

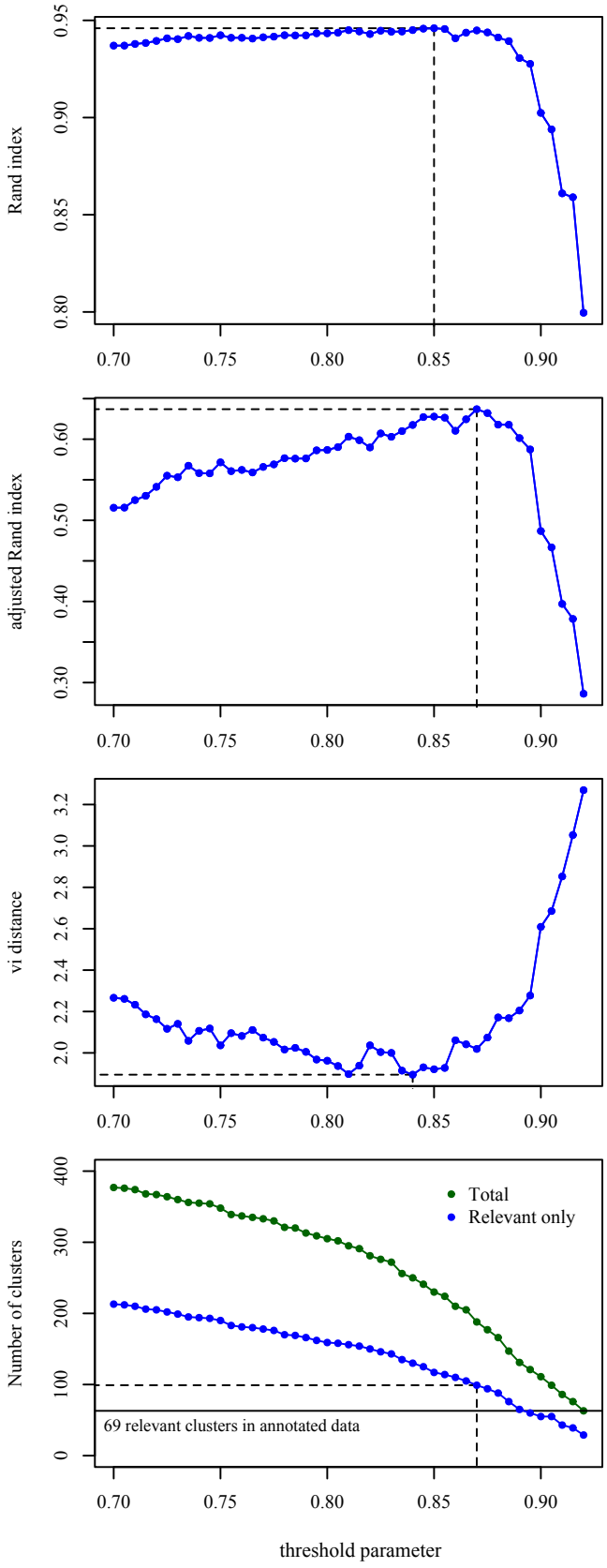

Fig. 6. Results of the threshold parameter optimization on the validation set: dashed lines highlight the performances of the best value for each index.

TABLE I
COMPARISON BY PAIRS OF GROUPS OF RETWEETS BETWEEN AUTOMATIC AND MANUAL PARTITIONS FOR THE TEST SET (THRESHOLD = 0.87)

Automatic partition → **Manual partition ↓**	Pair not in same cluster	Pair in same cluster
Pair not in same cluster	85616	3657
Pair in same cluster	1607	3950

Index	**Value**
Rand index or proportion of agreement (larger is better)	0.94
Sensitivity or true positive rate (larger is better)	0.71
Specificity or 1 - false negative rate (larger is better)	0.96
False discovery rate (smaller is better)	0.48

About half of the pairs automatically placed in the same cluster are not in the same cluster in the manual partition. We think that exploiting features other than the text will help to improve this statistic.

V. FUTURE WORK

This research project is at an early stage. We have developed, implemented, and tested a stream clustering algorithm to group similar tweets. There is still much work to do in order to reach the larger objective of developing an event detection approach and to integrate it into CrowdStack. We plan to:

- simultaneously exploit as many features as possible through ensemble clustering;
- filter clusters using an ensemble classification method to tag them as related or not to an event of interest;
- scale the algorithm to large streams of tweets by introducing cluster management.

So far, the novelty of our approach resides mostly in the mini-batch processing. We adapted a classical clustering algorithm, agglomerative hierarchical clustering, to that setting. Future work will bring more novelty, for instance by using a certain number of heterogeneous features in the clustering. We intend to perform a clustering per feature/dissimilarity measure combination, than to aggregate the clusterings using a chosen consensus function in order to get a single final clustering. This ensemble clustering approach (see [8] and [9]) is expected to produce a meaningful grouping of the tweets while bringing valuable information about features importance in the clustering.

To achieve the goal of performing event detection, a layer has to be added to our approach. Fig. 7 shows the proposed algorithm with that layer added. We plan on using an ensemble classification method to identify events of interest. We could use a technique as simple as majority vote from k-nearest neighbors classifier per feature/dissimilarity measure combination. We will conduct experiments to find an ensemble classification method performing well on our data. Annotated data will again be used, this time to train the classifiers. The annotated dataset needed here should contain clusters along with tags differentiating them between events and non-events.

Finally, with time, the number of clusters will keep growing. At some point, it will not be possible to calculate distances between incoming tweets and every clusters while continuing

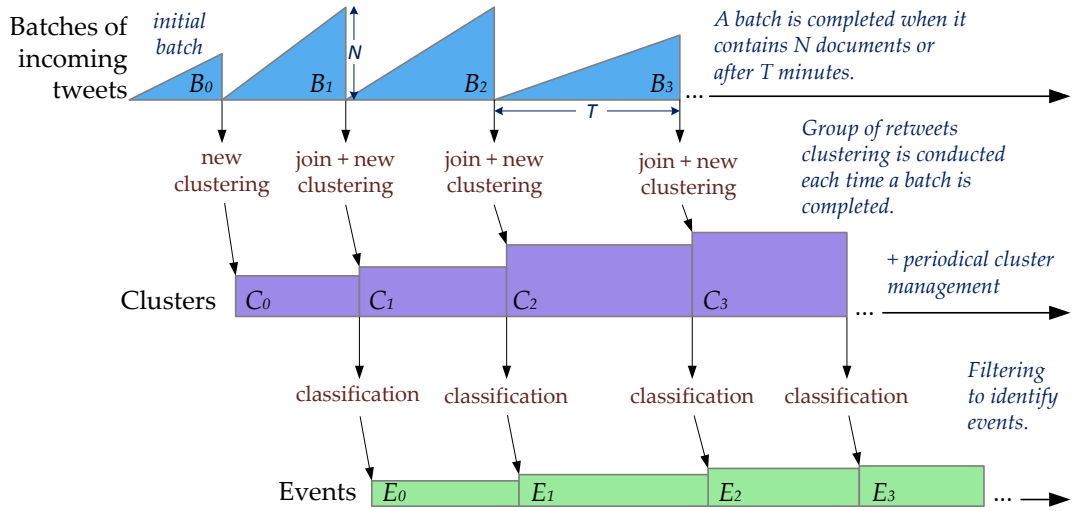

Fig. 7. Complete online mini-batch algorithm we plan to develop. B_t is the batch containing tweets monitored during the t^{th} time period and C_t is the cluster set after clustering groups of retweets in B_t. In the new part of the diagram, compared to figure 3, E_t is the set of clusters from C_t tagged as referring to an event of interest after filtering through classification.

to process data in real-time. We intend to develop a model to estimate the probability for a new group of retweets to join a cluster. When that probability becomes too small, a cluster will be considered inactive and no more distances will be calculated between that cluster and new groups of retweets. Periodically, we could also consider merging or splitting clusters, in order to manage situations where many clusters would become similar to one another, or a cluster would become too diverse to be considered homogeneous.

VI. Conclusion

Data found on social medias is abundant, it is continuously generated by its users and it comes with features of various types. By creating a clustering algorithm that targets this data source, we wanted to address the requirements involved in dealing with data in large volume, high velocity and variety, which are common characteristics of big data sources. In this initial phase of our work, we proved that our algorithm has the potential to handle such big data sources. In subsequent phases, we wish to prove it is successful in achieving meaningful results in an operational context with large quantities of data over a long period of time.

Acknowledgements

The authors would like to thank Thales Canada and Mitacs for funding a Mitacs Accelerate Internship and the Natural Sciences and Engineering Research Council of Canada (NSERC) for a PhD graduate scholarships granted to Sophie Baillargeon. They are also grateful to Annette Schwerdtfeger for proofreading this manuscript.

References

[1] C. Aggarwal and K. Subbian, "Event Detection in Social Streams," in *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2012, pp. 624–635.

[2] S. Kumar, H. Liu, S. Mehta, and L. V. Subramaniam, "From tweets to events: Exploring a scalable solution for Twitter streams," *arXiv preprint arXiv:1405.1392*, 2014.

[3] H. Becker, M. Naaman, and L. Gravano, "Beyond Trending Topics: Real-World Event Identification on Twitter," in *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011, pp. 1–17.

[4] S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming First Story Detection with application to Twitter," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 181–189.

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009.

[6] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[7] M. Meila, "Comparing clusterings - an information based distance," *Journal of Multivariate Analysis*, vol. 98, pp. 873–895, 2007.

[8] L. Zheng, T. Li, and C. Ding, "Hierarchical ensemble clustering," in *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 2010, pp. 1199–1204.

[9] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, 2007.