Real Time Event Monitoring with Trident

Igor Brigadir, Derek Greene, Pádraig Cunningham, and Gavin Sheridan

Clique Strategic Research Cluster, University College Dublin
igor.brigadir@ucdconnect.ie,{derek.greene,padraig.cunningham}@ucd.ie,
gavin.sheridan@storyful.com

Abstract. Building a scalable, fault-tolerant stream mining system that deals with realistic data volumes presents unique challenges. Considerable work is being done to make the development of such systems simpler, creating high level abstractions on top of existing systems. Many of the technical barriers can be eliminated by adopting a state-of-the-art interface, such as the Trident API for Storm. We describe a stream mining tool, based on Trident, for monitoring breaking news events on Twitter, which can be extended quickly and scaled easily.

Keywords: Stream Processing, Trident, Storm, Event Monitoring

1 Introduction

Recently there has been a significant shift online towards the task of content curation for online journalism. Media agencies such as Storyful[1] can now break or cover stories as they evolve by leveraging the content produced on social media platforms such as Twitter. However, given the massive volume of content produced by users of these platforms on a daily basis, the task of extracting content that is relevant to individual real-world news events presents a number of challenges. In particular, mining streams of this scale in real-time requires the adoption of new data processing frameworks and data mining algorithms.

In this paper, we present a new system for real-time monitoring streams of tweets to cluster relevant tweets around news events. In Section 3, we describe an initial application of this system in the context of breaking news on Twitter, and evaluate the usefulness of gathered tweets by allowing a journalist from Storyful to rate the relevancy of the resulting clusters with respect to six major news events from 2013. Based on this evaluation, we highlight specific issues that make the Twitter event monitoring task particularly difficult. We conclude in Section 4 with suggestions for future work to overcome these issues.

2 System Description

Our proposed stream monitoring system is built upon *Storm*[6], an open source framework for real-time distributed computation and data processing[2]. From an

[1] http://www.storyful.com
[2] http://storm-project.net

architectural perspective, the topology of a Storm system is formed from directed acyclic graphs containing two fundamental node types: “spouts” and “bolts”. Spouts produce tuples of data as their output, while bolts perform operations on tuples they receive as inputs.

“Trident” is a high-level abstraction framework for computation on Storm. As with the core Storm API, Trident uses spouts as the source of data streams. These streams are processed as batches of tuples partitioned among workers in a cluster. Trident provides means for transforming and persistently storing streams. The framework handles batching, transactions, acknowledgements, and failures internally. Tuple processing follows an “exactly once” semantic, making it easy to reason about processes, apply functions, filters and create aggregations from streams of tuples. Developing a fault tolerant, scalable stream mining system can be time consuming, but most implementation and deployment challenges can be avoided by using the Trident API and supporting software.

To cluster tweets, we have implemented two variations of a single pass algorithm suitable for streaming data. The first is the standard *sequential leader clustering algorithm* [3], a simple incremental approach that divides a dataset into k non-overlapping groups such that, for each group there is a “leader” data point and all other data points have similarity $\leq \tau$ to the leader. The second algorithm is a variation of this approach, which is described in Algorithm 1 and which we refer to as *moving leader clustering.* The moving leader approach attempts to capture new developments in a news story over time. Both algorithms use cosine similarity between tweets in the context of our proposed system.

```
Data: Stream of tweets; User assigned initial tweet leaders
Result: Sets of tweets attached to leaders
while A tweet Leader exists do
    Compare new tweet similarity to leaders;
    if $Sim > NewLeaderThreshold$ then Assign tweet as new Leader;
    else if $Sim > InclusionThreshold$ then Include tweet in Set;
    else Discard Tweet;
end
```

Algorithm 1: Moving Leader Clustering algorithm.

3 Evaluation

The public stream of tweets is generally useful for global trend detection and tracking, but is not so useful for tracking developing news stories, as it generally contains too many irrelevant messages for this task. As a solution, a set of about 20,000 “newsworthy” accounts has been curated and maintained by Storyful, which provides a useful filter. In our evaluation, we monitor a stream of tweets produced by this set of accounts. Filtering of this type vastly reduces the volume of spam tweets, and can help balance under represented groups of users.

Tweets have several characteristics that create challenges for traditional text analysis. Messages are very short (impacting term frequency), often contain misspellings or abbreviations, words are sometimes concatenated - especially in hash tags, punctuation is sparse (making tokenisation a challenge). Twitter specific

Event	*Tweets Streamed*	*Clustered*	*Leader Changes*
North Korean nuclear test	21,436,958	56,845	8,833
Chelyabinsk meteor	21,782,200	66,287	3,506
Pope Benedict to resign	17,260,343	19,036	2,654
Death of Hugo Chavez	11,276,989	45,197	3,356
Cyprus EU bailout	6,879,389	13,329	85
Canada exits UN convention	4,942,253	1,832	17

Table 1. Data for set of six news events from February and March 2013.

terms (@mentions, "RT", "via") and emoticons using unicode characters can also cause problems with automatic language detection. Prior to clustering, each tweet is stripped of these entities and English stop-words, and a corresponding term frequency vector is created. Tweets are also assigned a density score, which attempts to quantify the quality of a tweet, and promote longer, more informative tweets. Tweet density is the sum of term frequencies of non stopwords, divided by the number of stopwords they are apart, squared [1]. Tweets with a density score less than the query tweet were discarded. The density filter effectively removed short tweets that would otherwise be assigned very high similarity scores. For each event, 24 hours of tweets posted after the query tweet were streamed.

3.1 Results

To evaluate the two clustering algorithms described in Section 2, we examined their ability to find future relevant tweets for the six news events listed in Table 1, based on the provision of a single query "seed tweet". For each story, the first 30 tweets posted after the seed tweet and ranked by the system, were presented to a journalist. These tweets would typically represent the first few minutes of a breaking story, when need for information is greatest. A Storyful journalist manually provided relevancy judgements on a scale of 1 for a timely and useful tweet, 0 for a relevant tweet, and -1 for an irrelevant tweet. Quality of the results is based on the popular normalised discounted cumulative gain measure [4], as this measure reflects information usefulness for a journalist, discounting irrelevant and low ranking results. A summary of the scores achieved by both clustering techniques is shown in Fig. 1.

The moving leader approach was more suited to evolving stories, such as the bank bailout in Cyprus. However, constantly changing the leader in the cluster can impact results negatively, especially as users post more personal reactions after unexpected events, such as the resignation of the pope. The "Canada exits UN convention" story was interesting as it received very little attention overall. The scores for this story were low, however - the moving leader variant was able to retrieve the small number of relevant tweets.

The sequential leader was found to be more conservative, and better suited for gathering similar content for stories that do not develop beyond an initial announcement. There is also less chance of the cluster drifting off-topic, due to leader changes. Overall, the sequential leader clustering approach tended to capture many relevant, but near duplicate tweets, as was the case with the Meteor story - there were no tweets in this cluster that were useful for a journalist.

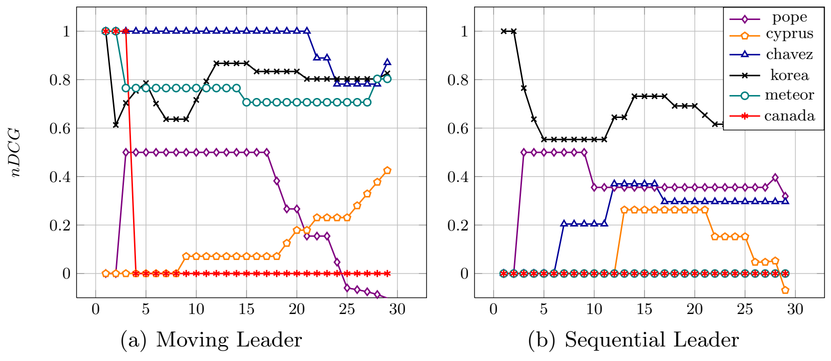

Fig. 1. nDCG at k tweets for both clustering methods, for six different news events.

4 Future Work

Specifics of tweets mean that term frequency-based similarity measures will often capture redundant information, and fail to explore related topics associated with a query. While the moving leader approach captured slightly more relevant tweets, performance on different types of events varied. We plan to investigate alternative ways of expanding tweet clusters around stories, and how topics associated with a news story change over much longer periods of time [2]. Other aspects of the data that are not fully explored involve the social connections between users. News stories tend to form unique collections of users that actively follow a story [5]. Following other interactions from these users could be a means to expand original queries, or to discover novel insights into an event. The next key component of our event monitoring system is an approach to summarise news events, based on the selection of a subset of key tweets from a given cluster, which serves to describe the evolution of an event as it unfolded on Twitter.

Acknowledgments. This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

References

1. R. Berendsen, M. Tsagkias, W. Weerkamp, and M. de Rijke. Pseudo test collections for training and tuning microblog rankers. In *Proc. SIGIR'13*. ACM, 2013.
2. F. Chong, T. Chua, and S. Asur. Automatic Summarization of Events from Social Media. *Proc. 7th Int. Conf. on Weblogs and Social Media*, 2013.
3. J. A. Hartigan. *Clustering algorithms.* John Wiley & Sons, Inc., 1975.
4. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.
5. J. Lehmann, C. Castillo, M. Lalmas, and E. Zuckerman. Transient News Crowds in Social Media. In *Proc. 7th Int. Conf. on Weblogs and Social Media*, 2013.
6. N. Marz. Storm: Distributed and fault-tolerant realtime computation, 2012.