High-Throughput Continuous Clustering of Message Streams

Oisín Boydell, Marek Landowski, Guangyu Wu, and Pádraig Cunningham

Centre for Applied Data Analytics Research (CeADAR), University College Dublin, Belfield, Dublin 4, Ireland
{firstname.lastname}@ucd.ie

Abstract. Continuous streaming data is commonplace across a wide range of domains and industries, and analysing these often high volume data streams in an on-line, low latency and scalable way can be challenging. We present a continuous clustering system for topic detection and monitoring in high-throughput message streams. A parallel implementation of sequential leader clustering runs on the Storm[1] stream processing framework to achieve scalable performance for real world applications, which we demonstrate for live topic detection in Twitter message data and for spam identification in SMS data streams.

1 Introduction

Clustering is a core technique which is commonly used for many diverse data mining tasks across different domains. A relevant example for streaming data is the detection of emerging topics in message streams. The ability to monitor discussion topics in a social media message stream, or detect which types of content are flowing through a communications network can be highly useful for understanding public opinion and interest, detecting and filtering spam and enabling a service owner to understand what content is passing through their infrastructure.

Applying clustering to streaming data presents a number of challenges when compared with traditional off-line, batch clustering. Firstly, high clustering accuracy is harder to achieve due to the constraints of processing continuous data streams. Multiple passes over the data by a clustering algorithm and an iterative process of cluster refinement are not possible. Secondly, the performance of the clustering algorithm must be able to handle the required data throughput, as a live data stream cannot be 'paused' while the clustering algorithm catches up. In real world scenarios, throughput is not constant and scalability is a concern: both in terms of scaling up to accommodate increased throughput and also scaling back down to conserve resources. Thirdly, any overview of the cluster information needs to convey the continuous and evolving nature of the data stream, a static snapshot of the clusters cannot represent any temporal fluctuations.

Our continuous clustering system for topic detection and monitoring is designed for high-throughput real world streaming data with particular emphasis on low-latency performance and scalability. We have developed a parallel implementation of the Sequential Leader Clustering (SLC)[1] algorithm implemented

[1] http://storm-project.net/

on Storm, the open source scalable data stream processing framework. In the next section we describe our approach in detail along with an overview of Storm and locality sensitive hashing (LSH), a core technique underpinning our parallel SLC implementation.

2 System Description

Although there are a number of algorithms specifically designed for clustering streaming data [2–4], the majority have not been developed with high-throughput and scalability as a primary concern. We have focussed on SLC which is computationally efficient, requiring only a single pass over the data and thus is applicable for continuous streaming data. SLC is not as precise or accurate over complex data as some more advanced algorithms, however it performs well for clustering items where set similarity measures such as Jaccard similarity are effective, as is the case with short text message content which is represented as a ‘bag of words’. Furthermore, we are able to parallelise and distribute the SLC algorithm by using LSH.

In LSH, items are hashed so that similar items (according to some similarity measure) produce the same hash values. This differs from the conventional use of hash functions: LSH aims to maximise the probability of collisions between similar items rather than avoid collisions. There are a number of variations of LSH for approximating different similarity metrics, we employ minHashing [5] which approximates Jaccard similarity. We generate multiple hash values for each message which allows us to control the error rate of the approximation.

This allows us to process and cluster messages arriving in the data stream at different nodes in parallel in a distributed computing environment without having to share and synchronise a master view of the current clusters between nodes. Messages which are similar, i.e. share hash values, and hence belong to the same cluster, are routed to the same node where cluster membership and statistics can be reported independently of the other processing nodes. The system can be horizontally scaled by adding or removing processing nodes to increase or decrease message processing capacity to match the throughput of the incoming data stream.

Our parallel version of SLC is implemented in Storm as a topology of interconnected processing units that together execute an algorithm continuously over the incoming data stream. Each processing unit or *bolt*, can receive, transform and then emit data to other connected bolts downstream in the topology. Storm runs on a computing cluster of networked machines and topologies are automatically distributed across the underlying physical infrastructure in order to reduce network hops and maximise individual processor usage and a key strength is its high throughput, high volume processing capability which matches our continuous clustering requirements.

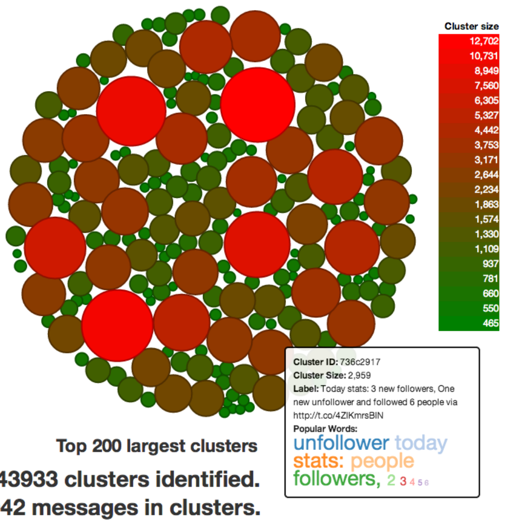
3 Applications and Performance

Detecting topics and trends in Twitter messages is a fertile area of research with many academic publications [6, 7] and companies[2] active in this area. Our

[2] http://analytics.topsy.com, http://socialmarketanalytics.com

continuous clustering approach is a good match for this task and is well suited to the high throughput, highly variable Twitter stream. It is reported that over 500 million tweets are posted on Twitter per day[3]. Although access to the raw Twitter firehose is not available to us we were able to gather a large dataset of randomly sampled tweets which were collected over a number of months. These tweets can then be replayed at an accelerated rate to replicate a much higher message throughput.

Our continuous clustering system is deployed on a Storm cluster comprising three worker machines (each a 12 core Intel processor with 64GB RAM) with the clustering output being written to a Redis in-memory key/value store cache. An animated live visualization of the clustering is dynamically generated from the Redis cache and an illustrative screen shot is shown in Figure 1. During performance testing we found that we could process, cluster and continually output the cluster results for over 3.4 million messages per minute (over 55,000 per second). Furthermore the system can be easily scaled by adding additional machines to the Storm cluster and we expect that further system tuning and configuration will achieve even greater throughput.



Cluster ID: 736c2917
Cluster Size: 2,959
Label: Today stats: 3 new followers, One new unfollower and followed 6 people via http://t.co/4ZIKmrsBIN

Top 200 largest clusters

543933 clusters identified.
2666942 messages in clusters.

Fig. 1. Screen shot of the Twitter topic detection application interface. Each circle's radius and colour represents the cluster size and mousing over a specific cluster shows additional information. Radius and colour could also be used to convey other information such as cluster growth rate or recency. The visualization continually updates as the data stream is processed.

Another application area is spam identification in SMS (text message) streams. As with other spam types, SMS spam tends to follow certain patterns such as

[3] http://www.telegraph.co.uk/technology/Twitter/9945505/Twitter-in-numbers.html

large volumes of similar or identical messages being sent to multiple subscribers over a short time period. The general message format and content may not have appeared in previous attacks and detecting an emerging attack while also being able to analyse the message content in order to block it at an early stage is of great benefit to network operators. Our real-time continuous clustering approach facilitates this early detection of spam by identifying emerging clusters of similar messages. The ability to report specific cluster behaviours such as fast growth rate can then be used to trigger whether further manual verification is required. Similarly to Twitter and other real world message streams, SMS streams in large mobile networks may carry thousands of messages per second and our parallel and scalable approach can handle these volumes in near real-time. We are currently evaluating the spam identification use case with a large real world SMS dataset provided by one of our industry partners and initial results show that spam attacks are being accurately clustered and detected by our system.

4 Conclusions and Future Work

We have presented a continuous clustering system for topic detection and monitoring in high-throughput message streams, and have given an overview of two specific applications. The research is still in its initial stages and a number of areas for future work have been identified including implementing other similarity metrics using locality sensitive hashing, clustering other types of data in addition to text, investigating alternative ways of reporting live clustering output for different tasks, undertaking further performance evaluation and exploring other potential application areas and domains.

Acknowledgements. We would like to acknowledge the support of our industry partner AdaptiveMobile, and to thank Enterprise Ireland and IDA Ireland for funding this research.

References

1. Hartigan, J.: Clustering Algorithms. John Wiley and Sons, New York (1975)
2. Guha, S., Meyerson, A., Mishra, N., Motwani, R.: Clustering data streams: Theory and practice. IEEE TKDE **15** (2003) 515–528
3. Fisher, D.H.: Iterative optimization and simplification of hierarchical clusterings. CoRR **cs.AI/9604103** (1996)
4. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th international conference on Very large data bases - Volume 29. VLDB '03, VLDB Endowment (2003) 81–92
5. Broder, A.Z.: On the resemblance and containment of documents. In: In Compression and Complexity of Sequences (SEQUENCES97, IEEE Computer Society (1997) 21–29
6. Mathioudakis, M., Koudas, N.: Twittermonitor: trend detection over the twitter stream. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. SIGMOD '10, New York, NY, USA, ACM (2010) 1155–1158
7. Cataldi, M., Di Caro, L., Schifanella, C.: Emerging topic detection on twitter based on temporal and social terms evaluation. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining. MDMKDD '10, New York, NY, USA, ACM (2010) 4:1–4:10