A Streaming Real-Time Web Observatory Architecture for Monitoring the Health of Social Machines

Ramine Tinati, Xin Wang, Ian Brown, Thanassis Tiropanis, Wendy Hall
University of Southampton
Web and Internet Science
{r.tinati,x.wang,ian.brown.tt2,wh}@ecs.soton.ac.uk

ABSTRACT

Over the past years, streaming Web services have become popular, with many of the top Web platforms now offering near real-time streams of user and machine activity. In light of this, Web Observatories now are faced with the challenge of being able to process and republish real-time, big data, Web streams, whilst maintaining access control and data consistency. In this paper we describe the architecture used in the Southampton Web Observatory to harvest, process, and serve real-time Web streams.

Categories and Subject Descriptors

H.3.5 [**INFORMATION STORAGE AND RETRIEVAL**]: Online Information Services—*Data sharing*

Keywords

Web Observatory; Real-time Processing; Data Security

1. INTRODUCTION

As the Web grows, the number of humans interacting with the Web, and the types of social machines are increasing, as are the devices used to engage with them. As a consequence of this changing technological and social landscape, there is a growing trend towards social machines emitting real-time streams of system and user activity. These real-time streams, which can be considered as big data due to their size, speed, and unstructured nature [15], offer Web Observatories with rich, timely resources for observation and analysis. Individually, these feeds provide a resource to measure the current state - or health - of a social machines, and combined, they have the potential to provide a collective pulse of the Web.

However, working with real-time big data sources introduces a number of new challenges for Web Observatory platforms which were initially built with technologies designed to process curated and structured resources.Many of these challenges are a result of the characteristics of big data, which is often large scale, high volume, and is not published in a standardised, (common) structured fashioned. Therefore, the challenge lies in being able to process them in a timely manner, which is computationally efficient, for both the publisher and subscriber. Aware of these challenges, the Web Observatory builders has already conducted substantial research into techniques for harvesting, processing, archiving, querying, and visualisation of data [9, 4, 6],

In addition to the challenges of processing real-time data, a Web Observatory introduces another layer of complexity: data control and sharing. A fundamental component of a Web Observatory is for data owners to control who has access to their data, and where their data is used. Therefore, Web Observatories need to extend the typical data processing pipeline of data collection-to-data visualisation, and append this with a layer of data sharing, security and access control, for *any* resource that it contains.

In this paper we describe the Southampton Web Observatory's approach to engineering a real-time streaming Web Observatory architecture to deliver resources via a custom developed API, which utilises common OAuth 2.0 protocols in order to provide access control to data owners. We also describe the type of metrics that real-time feeds can provide to monitor and measure the health of a social machine.

The remainder of this paper will be structured as follows, we first discuss the current state-of-the-art in terms of Web Observatories and real-time processing. We then introduce the Web Observatory concept and reference architecture, with particular focus on the streaming and API layers. We then describe our approach to harvesting and publishing via APIs. Finally we consider a number of scenarios for monitoring the health of social machines, and discuss the future areas of research still required.

2. RELATED WORK

In this section we describe existing and on-going research relevant to the development of Web Observatories and processing real-time Web data.

2.1 Web Observatory Development

The current body of Web Observatories are comprised of two main groups belonging either to systems conceived and developed specifically as Web Observatories [9] and those converging on Web Observatory status from other starting positions (such as analytics platforms or data repositories) [6, 4]. It should be noted that we do not include/exclude based on the specific designation "Observatory" for obvious reasons but focus rather on the functionality exhibited by the system (based on our developing taxonomy of Observatories [3].

Examples of the first class include the Southampton Web Observatory (SUWO) [9], the Singapore NUS NeXT Social Observatory [6], the Recorded Future Web Intelligence platform [1] and Quid [2] all

[1]`http://https://www.recordedfuture.com/`
[2]`http://http://quid.com/`

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2743977 .

of which are instances of systems gathering and preparing dynamic web-based external data for the purposes of decision support and insight.

The second class includes systems (whilst still comprising many of the features of WO) that are typic- ally orginally conceived with other purposes such as data repositories, data aggregation services or non-web research and anaytic tools. Examples here include the COSMOS Observatory, the ePrints document repository [3], Tamr [4], Import.IO [5] and the Datasift social network data platform [6].

The challenges around gathering, storing and analysing data at Web scale are vast when considered from a technical perspective: from integrating publishing technologies, to querying across multiple sources, to harmonising meta-data schemes and access methods. Added to this the challenge of making such data/ analyses accessible to non-technical users, exacerbates the design challenges even further. The following are a few exemplars who have contributed variously to these challenges. The NeXT platform has developed methods to access extremely high volume image and micro-blog processing algorithms offering product/person/location based views. The SUWO portal has developed mechanisms to host data and analytics and perform database agnostic querying from historic data sources. The COSMOS platform enables drag-and-drop analytics of locally-based data sets for non-technical researchers.

2.2 Real-time Processing

Many of the key challenges faced for implementing real-time processing for Web Observatories is also situated in a wider community of research of involved in real-time stream processing [18, 2]. As Heinze et al [10] describe, there are several important aspects of (real-time) stream processing including designing scalable solutions which are able to both partition and query data efficiently, and systems which have an amount of fault tolerance, being able to actively respond to changes in stream conditions.

Research on these topics include the optimisation of hardware and software processes to improve run-time efficiency [19, 1] and improve hardware load-balancing [22]. There are also substantial efforts involved in data integration and aggregation, developing approaches to data integration using different publish-and-subscribe paradigms [5, 8]. Our research is also situated in the area of event detection and processing [13], which contain overlap with topics of stream processing such as data integration and aggregation [7, 12, 16] but have a particular focus on using these techniques in order to extract events [17].

The basis of our work draws upon the techniques described in the aforementioned literature in order to develop a solution which utilises a publish-and-subscribe approach in combination with the core principles of performing real-time stream processing as described by Stonebraker et al. [18].

3. SOUTHAMPTON WEB OBSERVATORY (SUWO)

The Southampton Web Observatory (SUWO) engages user communities with dataset and analytic resources via the SUWO portal [7]. In general, the SUWO portal provides access to the following types of resources: (1) Datasets: these can be historical or real-time, quantitative or qualitative, and are heterogeneous in content. (2) Applications: these represent analytical applications, often supported via visualisations, which are linked back to the datasets listed on the Web Observatory portal. (3) Tools: analytical toolkits which provide analytical methods for datasets listed - but not a requirement - on the portal.

[3]`http://twitter,eprints.soton.ac.uk`
[4]`http://http://www.tamr.com/`
[5]`http://import.io`
[6]`http://www.datasift.com`
[7]`http://webobservatory.soton.ac.uk`

3.1 Architectural Principles

Four fundamental principles were considered in the design of the SUWO, a summarised account of these is listed below:

Not all datasets or applications can be public. Access to some datasets needs to be restricted for licensing, privacy or other reasons. The Web Observatory allows its users to list or host datasets that are public or private. Access to private datasets is managed by the user who hosts them on the Web Observatory. Since access to datasets can be restricted, access to applications that make use of those datasets needs to be restricted as well.

Web Observatories list two main types of resources: datasets and analytic applications, including visualisations. The link between a listed analytic application and the datasets that it uses must always be made explicit, even if the used datasets are listed as private, with restricted access.

Not all listed resources need to be locally hosted. Listed datasets or analytic applications can be hosted in remote servers managed by third parties.

Metadata describing the listed resources and projects are published. This way, descriptions of resources can be harvested and listed in other Web Observatories or Web-based resources.

Based on these four principles and the requirements described above, the design of the SUWO architecture aims to abstract the types of features and requirements into three sets of sub-components, as shown in Figure 1. Each of the separate components contain their own individual architecture, workflows, and associated technologies. They interface with each other using open standards and protocols to ensure the highest level of modularity and reconfiguration. Each sub-component functions autonomously, thus can potentially interact with other Web Observatories, as described in [20].

For this paper we focus specifically on the real-time processing technology and workflows used within the dataset component, and the construction of the Web Observatory API.

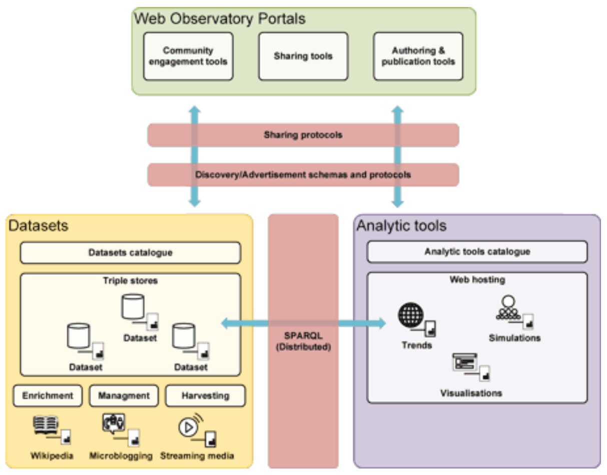


Figure 1: Architecture of the Southampton University Web Observatory

4. WEB OBSERVATORY: REAL-TIME PROCESSING AND PUBLISHING

In this section we introduce the architecture, technologies and configuration used to process and republish real-time data streams in the SUWO. We then describe the Web Observatory API and use of OAuth 2.0 as a security mechanism to provide data owners control over who has access to their data.

4.1 Harvesting Multiple Web Streams

Many Web services now provide programmatic access to platforms via APIs. Depending on the Web service, using the API can provide the complete collection of activities - the *firehose* connection - in real-time. Many of the social media platforms offer full-to-limited access to their social data, which typically contains information regarding their member's communications, interactions and activity.

4.1.1 Architecture and Configuration

As shown in Figure 2, the first stage of the real-time processing pipeline involves connecting to various external APIs in order. The *Pre-Processing Stage* involves the harvesting, enrichment and unification of various real-time streams from Web services. This process is achieved by using separate Web harvesters connected to the different external APIs, which are jointly controlled by the processing component. For each harvester, the External data streams are first collected, extracted from their own unique data schema, and then converted into the SUWO real-time stream JSON format [8]. As part of the conversion into a uniform schema we perform a lightweight enrichment process in order to ensure consistency across streams. This enrichment process, performed in real-time ensures that each record has at a minimum, a timestamp (ISO8068), source identifier (i.e. 'twitter_public_api') and unique identifier.

After the initial pre-processing stage, the enriched and uniformed data streams are then processed in the *Streaming Stage*, which uses the Advanced Message queueing protocol (AMQP) [21] to restream the incoming data sources. In our real-time processing pipeline, AMQP acts middleware for our publish-and-subscribe approach, it acts as a scalable technology which is designed for high-volume message processing and passing [11]. In order to take advantage of AMQP capabilities, we use RabbitMQ, an open source message broker and queueing service that supports advanced AMQP functionality, such as the in-memory queueing and exchange-based publish-and-subscribe services.

Many of the Web sources being harvested produce high volume feeds as a consequence of the rate at which incoming records are harvested and pre-processed, it is problematic to use a queueing approach to hold messages - in-memory or on disk - until clients connect and pop them off the queue. An alternative approach, which is far more suited to handling multiple, high volume data streams, is to take advantage of RabbitMQ's exchange mechanism, which effectively is a publish-and-subscribe service, which allows multiple clients to connect to a 'temporary' queue and retrieve the incoming messages. The advantage of this method is that it requires substantially less resources than a queueing approach, and that clients are receiving the *latest* data as soon as they connect. However, the disadvantage of this approach is that clients are offered no 'cache' when connecting to the exchange, thus if there is no incoming messages, the stream may appear to be inactive. For each pre-processed stream we instantiate a single exchange, and for combined streams, we use a recursive approach by connecting to the single stream exchanges, perform additional processing, and then publish this on a new exchange.

The final stage to the real-time stream processing workflow involves two components, internal consumption for archiving purposes, and a AMQP HTTP middleware in order to make the exchanges available via the Web Observatory portal. Although discussed in more detail in Section 4.2, the middleware used to connect to the AMQP exchanges is a lightweight service that does not process data, but provides the necessary handshaking and layer of security for clients to connect to the exchanges.

4.1.2 Challenges

As described in Section 2, there are various challenges associated with processing real-time streams of heterogeneous, unstructured data. One of the core challenges faced in developing this solution involved being able to to process the high volume of information in a timely fashion, which includes unifying, and in some cases enriching the data streams in order to obtain a consistent and usable stream. Many of the enrichment sub-processes required calls to external services (e.g. geographic location), which potentially slow down processing time. In order to overcome this, an internal cache of recently looked-up resources is used to ensure that processing of streams is performed in a timely manner.

Multi-stream integration is also an on-going area of research. Essentially, the integration of streams is achieved by finding a common field or 'pattern' between streams, which in many cases, is a topic, keyword, or simply, a timestamp. For the SUWO real-time stream integration, we are working on dynamic methods of integrating streams. One approach involves monitoring a the overall message rate of a given set of streams (i.e. posts per minute), and using fluctuations in stream volumes as an early indicator for combining streams undergoing similar changes.

4.2 Public Access - Web Observatory API

4.2.1 API Design

The SUWO maintains metadata of all resources, and those metadata are internally represented using the Data Catalog Vocabulary (DCAT) [14] (Figure 3). The SUWO API is built by mapping DCAT documents to a REST API in a way that preserves the semantics of DCAT. The SUWO API follows the Hypermedia as the Engine of Application State (HATEOAS) constraint of REST[9], that enables applications to explore the whole API from a single entry without referring to external documentations. A sample mapping of DCAT distribution to REST API is shown in Figure 4.

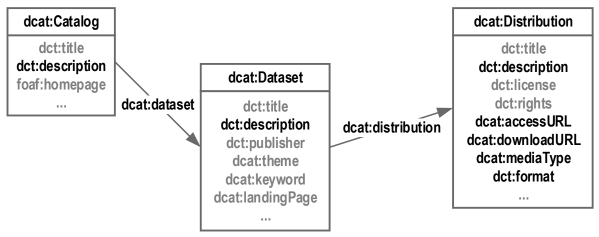


Figure 3: A simplified diagram demonstrating the three main classes of DCAT and their relationships. A complete diagram is given in [14].

[8]`http://webobservatory.soton.ac.uk/tutorial/realtime`

[9]Strictly speaking an API is not RESTful without HATEOAS

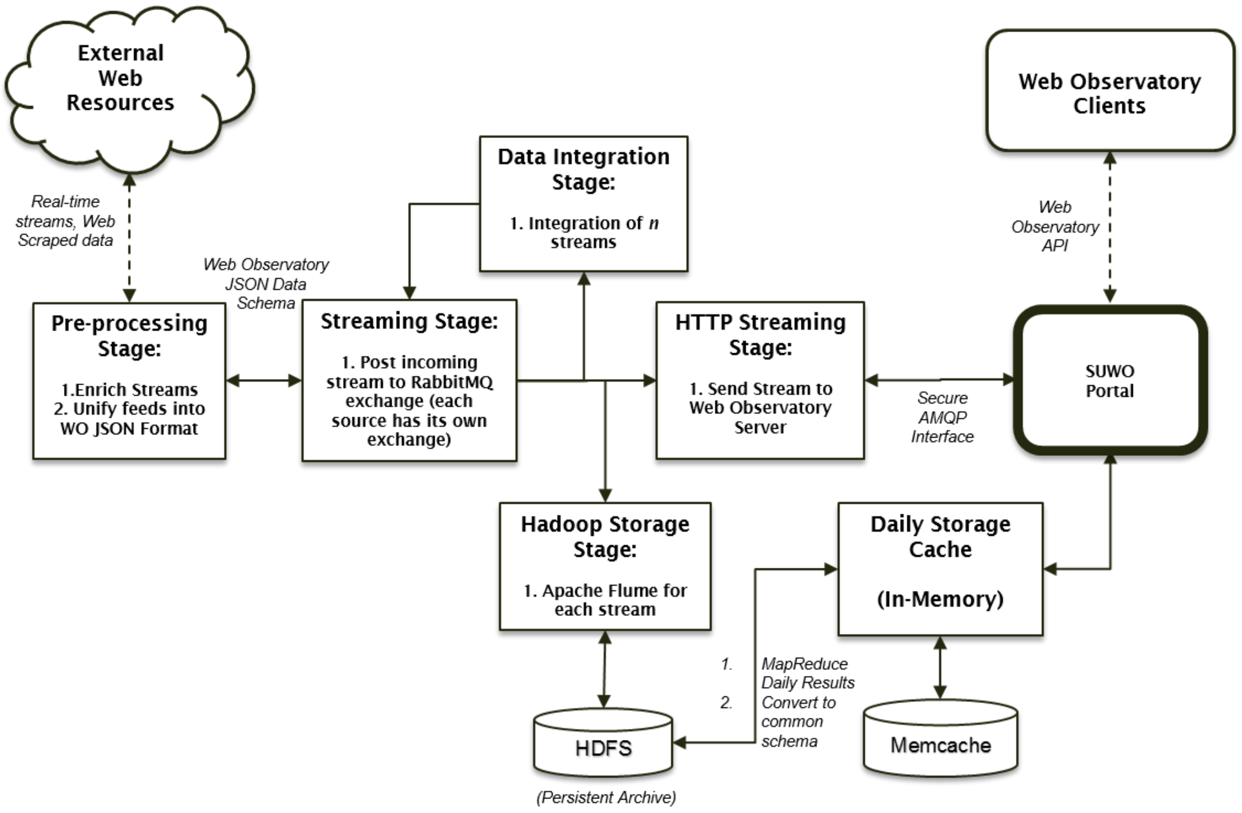
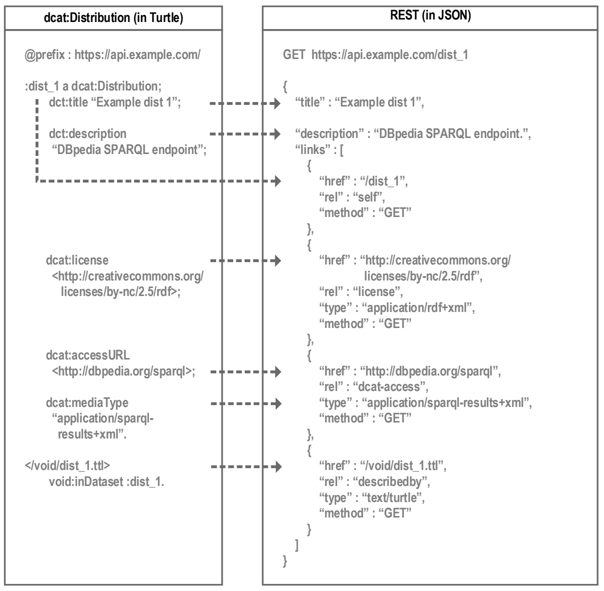

External Web Resources
Real-time streams, Web Scraped data
Web Observatory JSON Data Schema
Pre-processing Stage:
1.Enrich Streams
2. Unify feeds into WO JSON Format
Streaming Stage:
1. Post incoming stream to RabbitMQ exchange (each source has its own exchange)
Data Integration Stage:
1. Integration of n streams
HTTP Streaming Stage:
1. Send Stream to Web Observatory Server
Secure AMQP Interface
SUWO Portal
Web Observatory API
Web Observatory Clients
Hadoop Storage Stage:
1. Apache Flume for each stream
Daily Storage Cache
(In-Memory)
1. MapReduce Daily Results
2. Convert to common schema
HDFS
(Persistent Archive)
Memcache

Figure 2: SOWO Real-Time Streaming Architecture


dcat:Distribution (in Turtle)
REST (in JSON)

```
@prefix : https://api.example.com/

:dist_1 a dcat:Distribution;
    dct:title "Example dist 1";

    dct:description
     "DBpedia SPARQL endpoint";


   dcat:license
     <http://creativecommons.org/
       licenses/by-nc/2.5/rdf>;


   dcat:accessURL
     <http://dbpedia.org/sparql>;

   dcat:mediaType
     "application/sparql-
       results+xml".

</void/dist_1.ttl>
    void:inDataset :dist_1.
```

```
GET  https://api.example.com/dist_1

{
  "title" : "Example dist 1",

  "description" : "DBpedia SPARQL endpoint.",
  "links" : [
    {
      "href" : "/dist_1",
      "rel" : "self",
      "method" : "GET"
    },
    {
      "href" : "http://creativecommons.org/
                licenses/by-nc/2.5/rdf",
      "rel" : "license",
      "type" : "application/rdf+xml",
      "method" : "GET"
    },
    {
      "href" : "http://dbpedia.org/sparql",
      "rel" : "dcat-access",
      "type" : "application/sparql-results+xml",
      "method" : "GET"
    },
    {
      "href" : "/void/dist_1.ttl",
      "rel" : "describedby",
      "type" : "text/turtle",
      "method" : "GET"
    }
  ]
}
```

Figure 4: Mapping rules of a *dcat:Distribution* to its REST representation.

Accessing a resource requires two pieces of information: the *access URL* and its *media type*[10]. The access URL gives the location where the resource is available; the media type indicates the protocol and procedure for accessing the resource. The media type should be standard if possible, or a definition of the media type should be provided. For example, the resource shown in Figure 4 is a SPARQL endpoint (indicated by the media type "application/sparql-results+xml"[11]). Then applications will know that they can send SPARQL queries to the given URL. For AMQP streams we use a custom media type "application/amqp"[12]. Applications built on multiple streams can select resources of the type "application/amqp" and filter relevant ones based on the resource descriptions and keywords. It is also possible to combine static datasets and live streams in the same way. There are several advantages of the DCAT-to-REST approach:

Rich semantics: DCAT provides the information required to discover and access a resource. The mapping preserves all such information in the API.

Interoperability and automation: DCAT is an interoperable vocabulary, and HATEOAS enables applications to automatically traverse the whole API from a single start point. Combining both gives the opportunity of automatic resource discovery and retrieving.

Reversible mapping: The API has all the semantics of DCAT documents, it is straightforward to construct the source DCAT documents from the API structure.

Easy federation: DCAT documents can be combined into a bigger documents and mapped to a REST API. Therefore different in-

[10]A comprehensive list of media types is available at `http://www.iana.org/assignments/media-types/media-types.xhtml`

[11]`http://www.w3.org/TR/rdf-sparql-XMLres/`

[12]There is a media type for streams, "application/octet-stream". However in our case the media type has to be specific enough to enable applications to automatically access the resource.

stances of SUWO can be federated by simply merging their DCAT documents.

4.2.2 *Protecting resources using OAuth 2.0*

Not all resources are open to the public. The SUWO API adopts OAuth 2.0 to provide comprehensive yet flexible protection for both public and private resources. As shown in Figure 5, the SUWO acts as a reverse proxy of registered resources. That is, all requests for access are controlled by the SUWO before accessing the resources. Using OAuth users can authorise applications (either their own or third-party) to act on behalf of the users, and the authorised applications gain the same permission as the users. To access resources, authorised applications firstly authenticate themselves against OAuth 2.0, and the SUWO verifies whether the applications - the users who are viewing the application - are allowed to access the resources.

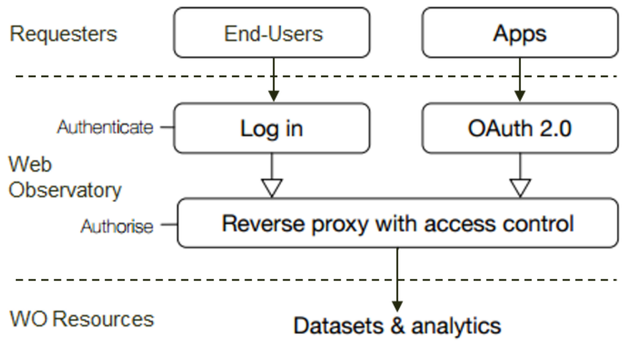


Figure 5: Requesters are users or applications initiating requests to access resources; SUWO authenticates requesters and authorises their requests, and resources, that are datasets and analytics shared on the web observatory.

5. MEASURING SOCIAL MACHINES USING SUWO

In this section we will address the purpose of designing a system capable of ingesting and republishing multiple Web streams in real-time, and discuss the type of metrics that provide an indication of a social machines state of operation.

5.1 Measuring Social Machines

A critical question for a Web observatory is what insights can such a platform provide, and how does real-time analysis provide additional value? In response to this, we argue that in order to support, improve, or even re-engineer a system, it is critical that it is first understood. The fundamental purpose of a Web Observatory such as SUWO, is to provide observational and analytical understanding of current state of the Web at various levels of granularity and types of interaction. A Web Observatory becomes the intermediary tool to measure a social machine using various different metrics in order to understand its functionality and state of operation.

In addition to this, real-time analysis of social machines turns the process of observation, analysis, and reporting into something far more responsive and reactive. By being able to monitor how a system is functioning in real-time offers the potential to react to sudden changes in types of activity, and from this, learning strategies can be built which can automatically reach depending of the fulfilment of certain criteria. Therefore, it is important to define a set of macro and micro level measurements and metrics that can be used to measure different characteristics of a social machine. The following section provides a number of metrics that a real-time stream can provide and that we are using in the SUWO.

5.1.1 *Social Machine Metrics*

Described below are a number of metrics we are currently examining in order to characterise the activity of a social machine. Many of these measurements can be used in combination with each other to provide a more detailed view of a social machines operation.

Macro-level system activity. A metric associated with the principle type of activity that a social machine is designed for. In many cases there are a number of measurements that can be considered as a primary indicator of activity. In such cases, these can all be used to represent system activity. For instance, considering a microblogging platform such as Twitter may be measured as the number of messages posted per second, as well as the number of re-shares between messages. Other classes of social machines may be measured as the number of tasks performed per second, or the number of active users in a given time period.

Community Engagement activity. Many systems involve the interaction between users, which may be communications, or some model of a friendship social network graph (i.e. Facebook, LinkedIn). The connectivity of this network can act as a measure of a social machine's structure, which can act as a proxy for the underlying community structure. For instance, measuring a social machines social graph may reveal distinct communities of users operating in isolation from each other, which, depending on the purpose of the system (i.e. a community-crowdsourced platform such as Wikipedia) , may reduce its performance, or effectiveness.

User-Centric Activity. This can be considered as a micro-level measurement as it represents a measure of activity based on a - potentially seeded - list of users of a system. This may monitor their frequency of activity (i.e. active session durations), or their communication patterns (i.e. who they interact with). Such measures may be useful for those monitoring a social machines that contains different types of users. For instance, in crowdsourcing platforms, these measures could be used to examine the speed to which new users engage with community or gain the necessary skills to complete an activity.

Topic-Centric Activity. This type of metric measures emerging topics within a social machine. This measurement may involve studying the frequency of activity associated with a specific object, such as text, multimedia, or data. These measurements provide an indicator of emergent *hot* or *trending* topics which may be useful as an alert for initiating a stream integration process with other social machines under observation.

Content/Sentiment Measure. This provides a system-level view of the current content, topics, or sentiment of a social machine. This measurement is of particular relevance to communication-based social machines such as microblogging platforms, as it provides an indicator of the current topics of discussion, or the current sentiment of its user base.

Information Flow. The flow of information of a resource (e.g. text, image, video, URL) can be used as a measurement to understand various characteristics of a social machine. For instance, tracing the flow of a word, phrase, or hashtag, in a user generated message-based social machine (e.g. Twitter), could provide insight into the current topics of interest, and how diverse such topic has become (topic virality).

6. CONCLUDING REMARKS

In this paper we describe an architecture for ingesting, processing, and publishing real-time big data for Web Observatories. We also describe how the application of a real-time Web Observatory can be used to measure several different characteristics of a social machine.

The architecture presented in this paper provides a scalable and secure solution for measuring multiple social machines using a common schema. From a Web Observatory perspective, being able to provide a level of access control over datasets and streams is a fundamental principle in order to allow data owners to retain control of their data. From a social machines researchers perspective, the ability to access unified, and in certain circumstances, integrated real-time streams of activity is a essential resource to understand, analyse, and possibly make predictions about the current state of a social machines health.

An immediate challenge for SUWO, and the network of Web Observatories, is to improve the current approach of integrating real-time streams, and also to be able to query incoming streams in order to filter the data for specific purposes. Future work will also include a wider analysis of the current and proposed metrics for measuring social machine activity, and how they contribute to understanding different classes of social machines. We also wish to explore the combination of metrics as a means to measure social machine activity.

7. ACKNOWLEDGEMENTS

Support for this study was provided by the *SOCIAM: The Theory and Practice of Social Machines* (EP/J017728/1) grant from the UK Engineering and Physical Sciences Research Council (EPSRC).

8. REFERENCES

[1] Aniello, L., Baldoni, R., and Querzoni, L. Adaptive online scheduling in storm. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, ACM (2013), 207–218.

[2] Artikis, A., Etzion, O., Feldman, Z., and Fournier, F. Event processing under uncertainty. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, ACM (2012), 32–43.

[3] Brown, I. C., Hall, W., and Harris, L. Towards a taxonomy for web observatories. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, International World Wide Web Conferences Steering Committee (Republic and Canton of Geneva, Switzerland, 2014), 1067–1072.

[4] Burnap, P., Rana, O., Williams, M., Housley, W., Edwards, A., Morgan, J., Sloan, L., and Conejero, J. Cosmos: Towards an integrated and scalable service for analysing social media on demand. *International Journal of Parallel, Emergent and Distributed Systems*, ahead-of-print (2014), 1–21.

[5] Chen, J., Ramaswamy, L., and Lowenthal, D. Towards efficient event aggregation in a decentralized publish-subscribe system. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, ACM (New York, NY, USA, 2009), 18:1–18:11.

[6] Chua, T.-S., Luan, H., Sun, M., and Yang, S. Next: Nus-tsinghua center for extreme search of user-generated content. *MultiMedia, IEEE 19*, 3 (July 2012), 81–87.

[7] Fawcett, T., and Provost, F. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, ACM (New York, NY, USA, 1999), 53–62.

[8] Frischbier, S., Margara, A., Freudenreich, T., Eugster, P., Eyers, D., and Pietzuch, P. Asia: Application-specific integrated aggregation for publish/subscribe middleware. In *Proceedings of the Posters and Demo Track*, Middleware '12, ACM (New York, NY, USA, 2012), 6:1–6:2.

[9] Hall, W., Tiropanis, T., Tinati, R., Wang, X., Luczak-Rosch, M., and Simperl, E. The web science observatory-the challenges of analytics over distributed linked data infrastructures. *ERCIM News*, 96 (2014), 29–30.

[10] Heinze, T., Aniello, L., Querzoni, L., and Jerzak, Z. Cloud-based data stream processing. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, ACM (New York, NY, USA, 2014), 238–245.

[11] Houston, P. Building distributed applications with message queuing middleware, 1998.

[12] Krishnamurthy, S., Wu, C., and Franklin, M. On-the-fly sharing for streamed aggregation. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, ACM (New York, NY, USA, 2006), 623–634.

[13] Luckham, D. C. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[14] Maali, F., and John Erickson. Data Catalog Vocabulary (DCAT), 2014.

[15] Manyika, J., Chui, Michael Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Hung Byers, A. Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute* (2011).

[16] Pandey, N. K., Zhang, K., Weiss, S., Jacobsen, H.-A., and Vitenberg, R. Distributed event aggregation for content-based publish/subscribe systems. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, ACM (New York, NY, USA, 2014), 95–106.

[17] Reuter, T., and Cimiano, P. Event-based classification of social media streams. In *Proceedings of the 2Nd ACM International Conference on Multimedia Retrieval*, ICMR '12, ACM (New York, NY, USA, 2012), 22:1–22:8.

[18] Stonebraker, M., Çetintemel, U., and Zdonik, S. The 8 requirements of real-time stream processing. *SIGMOD Rec. 34*, 4 (Dec. 2005), 42–47.

[19] Su, H., Rundensteiner, E. A., and Mani, M. Automaton in or out: Run-time plan optimization for xml stream processing. In *Proceedings of the 2Nd International Workshop on Scalable Stream Processing System*, SSPS '08, ACM (New York, NY, USA, 2008), 38–47.

[20] Tiropanis, T., Wang, X., Tinati, R., and Hall, W. Building a connected web observatory: architecture and challenges. In *2nd International Workshop on Building Web Observatories (B-WOW14), ACM Web Science Conference 2014* (June 2014).

[21] Vinoski, S. Advanced message queuing protocol. *IEEE Internet Computing 10*, 6 (Nov. 2006), 87–89.

[22] Wang, W., Sharaf, M. A., Guo, S., and Özsu, M. T. Potential-driven load distribution for distributed data stream processing. In *Proceedings of the 2Nd International Workshop on Scalable Stream Processing System*, SSPS '08, ACM (New York, NY, USA, 2008), 13–22.