Social Media Analytics: The Kosmix Story

Xiaoyong Chai[1], Omkar Deshpande[1], Nikesh Garera[1], Abhishek Gattani[1], Wang Lam[1], Digvijay S. Lamba[1], Lu Liu[1], Mitul Tiwari[2], Michel Tourn[3], Zoheb Vacheri[1], STS Prasad[1], Sri Subramaniam[1], Venky Harinarayan[4], Anand Rajaraman[4], Adel Ardalan[5], Sanjib Das[5], Paul Suganthan G.C.[5], AnHai Doan[1,5]

[1] @WalmartLabs, [2] LinkedIn, [3] Google, [4] Cambrian Ventures, [5] University of Wisconsin-Madison

1 Introduction

Kosmix was a Silicon Valley startup founded in 2005 by Anand Rajaraman and Venky Harinarayan. Initially targeting Deep Web search, in early 2010 Kosmix shifted its main focus to social media, and built a large infrastructure to perform social media analytics, for a variety of real-world applications.

In 2011 Kosmix was acquired by Walmart and converted into @WalmartLabs, the advanced research and development arm of Walmart. The goals of the acquisition were to provide a core of technical people in the Valley and attract more, to help improve traditional e-commerce for Walmart, and to explore the future of e-commerce. This future looks increasingly social, mobile, and local. Accordingly, @WalmartLabs continues to develop the social media analytics infrastructure pioneered by Kosmix, and uses it to explore a range of social e-commerce applications.

In this paper we describe social media analytics, as carried out at Kosmix. While our framework can handle many types of social media data, for concreteness we will focus mostly on tweets. Section 2 describes the analytics architecture, the applications, and the challenges. We describe in particular the Social Genome, a large real-time social knowledge base that lied at the heart of Kosmix and powered most of its applications. Section 3 describes how the Social Genome was built, using Wikipedia, a set of other data sources, and social media data. Section 4 describes how we classify and tag tweets, and extract entities from tweets and link them to a knowledge base. Section 5 describes how we detect and monitor events in the Twittersphere. Section 6 discusses how we process the high-speed Twitter stream using Muppet, a scalable distributed stream processing engine built in house [1]. Section 7 discusses lessons learned and related work, and Section 8 concludes. Parts of the work described here have been open sourced [1] and described in detail in recent papers [18, 23, 25, 32].

2 Architecture, Applications, and Challenges

The overall Kosmix architecture for social media analytics is shown in Figure 1.a. To start, we retrieve data from multiple sources, including Web sources (e.g., Wikipedia, Musicbrainz, Chrome, Yahoo Stocks) and social media sources (e.g., Twitter, Foursquare, YouTube, Facebook, Flickr). In particular, Kosmix had access to the full Twitter fire hose, which streamed in at about 3,000 tweets per second. Thus, fast and accurate processing of this fire hose became a major challenge.

In the next step, we process the retrieved data using a variety of techniques in information extraction, integration, entity matching and merging, schema matching, and event detection and monitoring, among

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

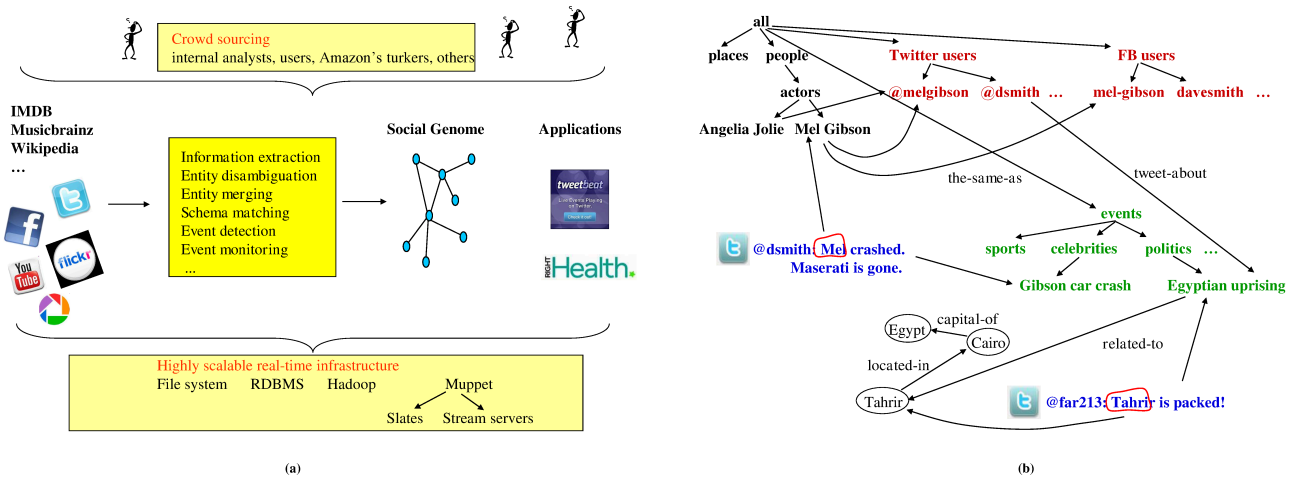


Figure 1: (a) The overall architecture for analytics, and (b) a sample fragment of the Social Genome.

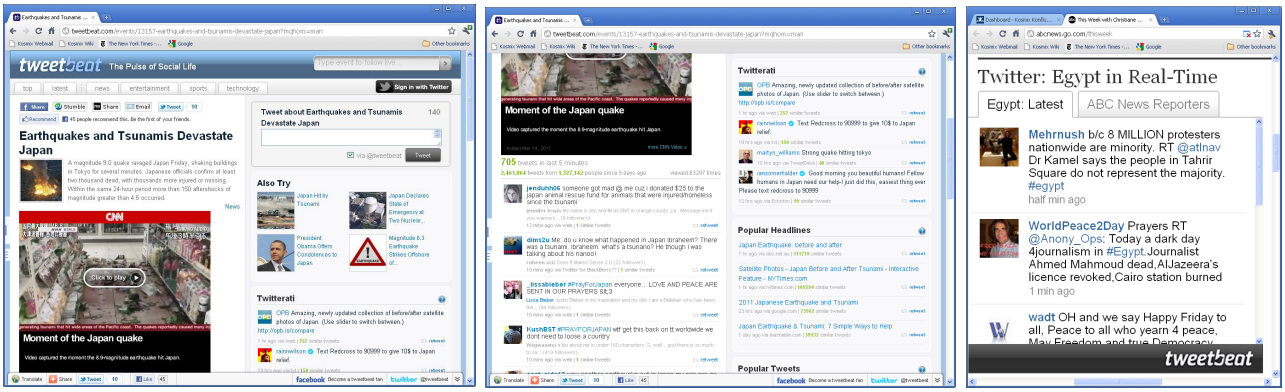
others. Our goal was to build a large real-time knowledge base called *Social Genome*, which captures all important entities, relationships, and events that are happening in real time in social media. Then we leverage the Social Genome to build a variety of applications, such as TweetBeat, Firsthand, Social Cube, and RightHealth. We run the above pipeline on a highly scalable real-time data processing infrastructure, which uses the file system, RDBMSs, Hadoop, and Muppet. Muppet is a distributed stream processing engine developed in house, and was used to process the high-speed stream of tweets flowing into Kosmix. Thus, it is similar to Storm at Twitter, but with important differences (see Section 7). Throughout the entire processing pipeline, we also employed crowdsourcing (using internal analysts, Amazon Mechanical Turk's workers, end users, etc.) to improve the accuracy of the processed data.

As described, the Social Genome knowledge base lies at the heart of the Kosmix analytics infrastructure. This knowledge base consists of the following (as illustrated in Figure 1.b):

- A Freebase-like knowledge base of popular concepts and instances on the Web, such as places, people, actors, politicians, Angelina Jolie, and Mel Gibson (see the top-left corner of the figure). This Web knowledge base, called *Kosmix KB*, was constructed by integrating Wikipedia with several other databases (see Section 3).
- Profiles of social media users: Twitter users such as @melgibson, @dsmith; Facebook users; Foursquare users, and so on (see the top-right corner of the figure).
- Popular events detected in the Twittersphere, such as Gibson car crash, Egyptian uprising, earthquakes (see the bottom-right corner).
- Tweets and other raw data such as Web pages (e.g., "Mel crashed. Maserati is gone." and "Tahrir is packed" in the figure).

In addition, the Social Genome also contains many relationships inferred by our algorithms. For example, we matched person instances in the Web knowledge base (i.e., the Kosmix KB) with social media user profiles, to create "the-same-as" relationship wherever appropriate. For example, we created a "the-same-as" relationship between the person instance "Mel Gibson" in the Kosmix KB and Twitter user @melgibson (see Figure 1.b), because they refer to the same real-world person.

As another example, when receiving the tweet "Mel crashed. Maserati is gone.", we established that "Mel" in the tweet is a person name, and that it refers to the person instance "Mel Gibson" in the Kosmix KB (see Figure 1.b). In other words, we perform entity extraction over tweets and link the discovered entities



(a) (b) (c)

Figure 2: (a)-(b) Screen shots of the TweetBeat homepage showing the 2011 Japanese earthquake event, and (c) a TweetBeat widget embedded in ABC news homepage showing the 2011 Egyptian uprising event.

to the same entities in the Kosmix KB. Other examples of relationships include a Twitter user tweeting about an event (e.g., @dsmith tweeting about Egyptian uprising), and an event (e.g., Egyptian uprising) is related to an instance in the Kosmix KB (e.g., Tahrir).

We used the Social Genome to build a variety of real-world applications. Our flagship application was TweetBeat, which monitors the Twittersphere to detect important emerging events (e.g., earthquake, uprising, stock crash), then displays the most important tweets of these events in real time. For example, when TweetBeat detected that the 2011 Japanese earthquake was happening, it created an entire page for this event. The top part of this page (Figure 2.a) names the event, and gives a brief description and a picture or video (if available). The bottom part of the page (Figure 2.b) shows important tweets about this event in real time, in a continuously scrolling fashion (see the left part of this figure). Figure 2.c shows a TweetBeat widget that was embedded in the ABC news homepage back in 2011. This widget shows important tweets for the event Egyptian uprising in real time. During the period 2010-2011, TweetBeat detected and monitored hundreds of important events per day.

Firsthand is another example application that used the Social Genome[1]. When a user is reading a news article, Firsthand (installed as a browser widget) detects and highlights entities (e.g., people, organizations) that appear the article and have Twitter accounts. If the user hovers the mouse over such an entity, Firsthand will retrieve and display the latest tweets from the corresponding account. This application makes use of "the-same-as" relationships in the Social Genome. Specifically, we extract and link entities that appear in the article to instances in the Kosmix KB, then follow "the-same-as" links to access the Twitter accounts of these entities.

SocialCube is another application in which we leveraged the Social Genome to build a real-time data cube with dimensions such as location, topics, and sentiment, then used it to answer questions such as "How many are tweeting about Barack Obama in New York, by the minute for the last hour?" and "How many Twitter users in Arizona feel positive about the new Medicare plan?".

As described, developing the analytics infrastructure and the applications on top of it raises difficult challenges. First, how do we build the various knowledge bases? Second, when a tweet comes in, how do we classify and tag the tweet, and extract and link entities, such as finding out that "Mel" in the tweet is a person name and refers to the person instance Mel Gibson in the Kosmix KB? Third, how do we detect emerging important events in the Twittersphere (e.g., earthquakes), and how do we monitor tweets of these events? Finally, how do we process social media data in real time? We discuss our solutions to these

[1] appscout.pcmag.com/social-networking/269719-tweetbeat-firsthand-read-someone-s-tweets-anywhere-they-re-mentioned

challenges in the subsequent sections.

3 Building the Social Genome Knowledge Base

We now describe building the Social Genome. But before that, we briefly describe the notion of a knowledge base. A knowledge base (KB) typically consists of a set of concepts organized into a taxonomy, instances for each concept, and relationships among the concepts. Figure 1.b shows a tiny KB (in the top-left corner), which illustrates the above notions. In this KB, for example, "actors" are a kind of "people", and Angelina Jolie and Mel Gibson are instances of "actors".

To build the Social Genome, we first build the Kosmix KB, which is a knowledge base of popular concepts and instances on the Web, such as places, people, actors, politicians, Angelina Jolie, Mel Gibson, etc. In this sense it is similar to Freebase and Google's Knowledge Graph. Then we augment the Kosmix KB with social-media information, such as Twitter user profiles, events, and tweets. We now describe these two steps in more details.

To build the Kosmix KB (see [18] for a detailed description), we convert Wikipedia into a KB, then integrate it with additional data sources, such as Chrome (an automobile source), Adam (health), MusicBrainz, City DB, and Yahoo Stocks. Here we highlight several interesting aspects that have not commonly been discussed in the KB construction literature. First, we found that converting Wikipedia into a taxonomy is highly non-trivial, because each node in the Wikipedia graph can have multiple paths (i.e., lineages) to the root. We developed an efficient solution to this problem. Interestingly, it turned out that different applications may benefit from different lineages of the same node, so we convert Wikipedia into a taxonomy but preserve all lineages of all Wikipedia nodes.

Second, extracting precise relationships from Wikipedia (and indeed from any non-trivial text) is notoriously difficult. We developed a solution that sidesteps this problem and extracts "fuzzy relationships" instead, in the form of a relationship graph. Later we were able to use this fuzzy relationship graph in a variety of real-world applications. Third, we extracted meta information for the nodes in the KB, focusing in particular on social information such as Wikipedia traffic statistics and social contexts. For example, given the instance "Mel Gibson", we store the number of times people click on the Wikipedia page associated with it, the most important keywords associated with it in social media in the past 1 hour (e.g., "mel", "crash", "maserati"), and so on. Such meta information turns out to be critical for many of our applications. Finally, we added new data sources to the KB constructed out of Wikipedia. In doing so, we had to match external instances with those in the KB, and heavily used taxonomy matching and entity instance matching algorithms.

Building the initial KB is difficult, but is just the very first step. In the long run, maintaining and curating the KB pose the most challenges and incur most of the workload. We developed a solution to refresh the KB every day by rerunning most of it from the scratch. We also had to address a major technical challenge: how to curate the KB and preserve the curation after refreshing the KB. Our solution is to capture most of the human curation in terms of commands, and then apply these commands again when we refresh the KB.

Once we had built the Kosmix KB, we added social media data to it. Examples include adding profiles of social media users (e.g., Twitter users), events, and annotated tweets. This process raises two key challenges. First, how do we perform entity extraction, linking, classification, and tagging for tweets? And second, how do we detect and monitor events in the Twittersphere? In the next two sections we discuss these two challenges.

4 Entity Extraction, Linking, Classification, and Tagging

To augment the Kosmix KB with social media data, we need to perform entity extraction, linking, classification, and tagging for the incoming tweets. For example, given a tweet such as "Obama gave an immigration speech while on vacation in Hawaii", *entity extraction* determines that string "Obama" is a person name, and that "Hawaii" is a location. *Entity linking* goes one step further, inferring that "Obama" actually refers

to a particular entity in the Kosmix KB and that "Hawaii" refers to another entity. *Classification* assigns a set of predefined topics to the tweet, such as "politics" and "travel". Finally, *tagging* assigns descriptive tags to the tweet, such as "politics", "tourism", "vacation", "President Obama", "immigration", and "Hawaii", the way a person may tag a tweet today. Our applications heavily use the results of such extraction, linking, classification, and tagging.

To solve the above problems, we proceed as follows (see [23] for more details). Given a tweet, we preprocess it, e.g., detecting the language, tokenizing. Next, we use the Kosmix KB to extract mentions from the tweet, remove certain mentions, then score the remaining ones. Here a mention refers to a pair of (string, KB node), which states that the string refers to a particular node in the Kosmix KB. So we are effectively performing entity extraction and linking at the same time. Then in the next step we use these mentions to classify and tag the tweet. Next, we go back to processing the mentions, but do so in more depth. Specifically, we extract 30+ mention features, remove certain mentions using rules involving these features, disambiguate the mentions (e.g., linking "apple" to Apple the company not Apple the fruit), then score the mentions again. Next, we use the "clean" mentions to classify and tag the tweet again. Finally we apply hand-crafted editorial rules to filter mentions and classification and tagging results. Compared to current work, our solution is distinguished in several important aspects:

- *Using a Global and "Real-Time" Knowledge Base:* The Kosmix KB (which we use to find and link to entities mentioned in tweets) is built from Wikipedia. Wikipedia is global in that it contains most concepts and instances judged important in the world. Thus, it provides a good coverage for the tasks. More importantly, it is "real time" in that contributors continuously update it with new entities that just appear in real-world events. This "real time" nature makes it especially well-suited for processing social data. In contrast, many current solutions use knowledge bases that are updated less frequently.

- *Synergistic Combination of the Tasks:* Our system interleaves the four tasks of extraction, linking, classification, and tagging in a synergistic fashion. For example, given a tweet, we begin by performing a preliminary extraction and linking of entity mentions in that tweet. Suppose many such mentions link to many nodes under the subtree "Technology" in the Kosmix KB. Then we can infer that "Technology" is a likely topic for the tweet, thereby helping classification. In return, if we have determined that "Technology" is indeed a topic for the tweet, then we can infer that string "apple" in the tweet likely refers to the node "Apple Corp." in the KB, not the node "Apple (fruit)", thereby helping entity linking.

- *Using Contexts and Social Information:* Given a tweet such as "go giants!", without some context, such as knowing that this user often tweets about the New York Giants football team, it is virtually impossible to extract and link entities accurately. As another example, it is not possible to process the tweet "mel crashed, maserati gone" in isolation: we have no idea which person named Mel the user is referring to. However, if we know that in the past one hour, when people tweeted about Mel Gibson, they often mentioned the words "crash" and "maserati" (a car brand), then we can infer that "mel" likely refers to the node Mel Gibson in the KB. Our system exploits such intuitions. It collects contexts for tweets, Twitter users, hash tags, Web domains, and nodes in the Kosmix KB. It also collects a large number of social signals (e.g., traffic on Wikipedia and Pinterest pages). The system uses these contexts and signals to improve the accuracy of the tasks.

Other important features of our system include a minimal use of complex time-intensive techniques, to ensure that we can process tweets in real time (at the rate of up to 6000 tweets per second), and the use of hand-crafted rules at various places in the processing pipeline to exert fine-grained control and improve system accuracy.

5 Event Detection and Monitoring

As discussed earlier, we process the social media stream (e.g., Twitter fire hose, Foursquare checkins) to detect important emerging events, then monitor these events. Much work in academia and industry has addressed event detection. However, this work has been limited in three main ways. First, it typically exploits just one kind of heuristics, such as finding popular and strongly related keywords (e.g., Egypt, revolt). Second, it does not scale to the high volume of data streaming in, typically because the work does not exploit distributed and parallel processing on a cluster of machines. Finally, the work has not exploited crowdsourcing to improve the accuracy of event detection.

Our current event detection solution addresses these limitations. First, we employ many heuristics to detect event candidates. For example, a heuristic finds keywords that suddenly become hot and strongly related (e.g., "Haiti" suddenly became hot, and "Haiti" and "earthquake" suddenly co-occurred in many tweets). Another heuristic monitors Twitter accounts that are well known for broadcasting breaking news. Yet another heuristic checks to see if a large number of people (e.g., more than 15) check into the same location in Foursquare (potentially indicating that an event is taking place at that location), and so on. We evaluate these heuristics against the social media stream using Muppet, our in-house distributed stream processing engine, run over a cluster of machines. Finally, we employ crowdsourcing to remove false-positive events and to extract important meta data for the remaining events.

Once we have detected an event, we monitor the Twitter sphere to find tweets related to this event, then display the most important tweets. This is often called *event monitoring* or *tracking*. The simplest, and most common, solution for event monitoring is to manually write rules to match tweets to events. For example, if a tweet contains certain keywords or user IDs, then it is flagged as positive. This solution is conceptually simple, easy to implement, and often achieves high initial precision. But it suffers from three limitations. First, manually writing rules is labor intensive and does not scale to hundreds or thousands of events per day. Second, manually writing good rules can be quite hard for many events. Finally, and most importantly, rules often become invalid or inadequate over time. For example, when a shooting happened in Baltimore in 2011, initially Twitter users referred to it using the keywords "Baltimore" and "shooting". A few hours later, however, when it was clear that the shooting happened on the John Hopkins campus, most Twitter users referred to it as the "John Hopkins shooting" instead of the "Baltimore shooting", thus rendering ineffective any rules that mention "Baltimore" and "shooting". To address the above limitations, our solution uses machine learning to evolve the profile of an event over time, then uses the learned profile to find tweets related to the event.

6 Scalable Processing of Fast Data

We run most of the social media analytics pipeline on Muppet, an in-house system that processes the incoming social data streams (e.g., Twitter fire hose, Foursquare checkins) in a distributed fashion, over a cluster of machines. Muppet was motivated by the need to process such streams with minimal latency and high scalability. For example, an application that monitors the Twitter fire hose for an ongoing earthquake may want to report relevant information within a few seconds of when a tweet appears, and must handle drastic spikes in the tweet volumes. The key idea behind Muppet is to provide a MapReduce-like framework for fast data (i.e., high-speed data streams), so that developers can quickly write and execute such applications on large clusters of machines.

To realize the above idea, we first developed MapUpdate, a framework to process fast data. Like MapReduce, in MapUpdate the developer only has to write a few functions, specifically *map* and *update* ones. The system automatically executes these functions over a cluster of machines. MapUpdate however differs from MapReduce in several important aspects. First, MapUpdate operates on data streams. So map and update functions must be defined with respect to streams. For example, mappers map streams to streams, split streams, or merge streams, while updaters process events flowing in from one or multiple streams.

Second, streams may never end. So updaters use storage called *slates* to summarize the data that they have seen so far. The notion of slates does not arise in MapReduce, nor in many recently proposed stream processing systems (see the related work section). In MapUpdate, slates are in effect the "memories" of updaters, distributed across multiple map/update machines, as well as persisted in a key-value store for later processing. Making such "memory pieces" explicit and managing them as "first-class citizens", in a real-time fashion, is a key distinguishing aspect of the MapUpdate framework. Finally, a MapUpdate application often involves not just a mapper followed by an updater, but many of these, in an elaborate workflow that consume and generate data streams.

We then developed Muppet, a MapUpdate implementation. In [25] we discuss the key challenges of Muppet in terms of distributed execution, managing slates, handling failures, reading slates, and sketch our solutions. Since mid-2010, we have used Muppet extensively to develop many social media and e-commerce applications.

7 Lessons Learned & Related Work

Our work on social media analytics suggests several important lessons. First, analyzing social data is fundamentally much harder than analyzing "traditional" data, due to a lack of context, dynamic environment (concepts appear and disappear quickly), quality issues (lots of spam), quick spread of information, and fast data. Second, context is vital to analyzing social data. Given a tweet such as "go giants!", without some context, such as knowing that this user often tweets about the New York Giants football team, it is virtually impossible to extract and link entities accurately. As another example, it is not possible to process the tweet "mel crashed, maserati gone" in isolation: we have no idea which person named Mel the user is referring to. Third, it is important to use a knowledge base to help classify and tag tweets, and to extract and link entities in tweets. Finally, crowdsourcing is indispensable (e.g., in building knowledge bases, evaluating detected events), but raises many interesting challenges.

In terms of related work, in the past few years a wealth of work has addressed the problem of social media analytics, in both academia and industry (e.g., Topsy, Stocktwits, [9–11,14,16,33]). But this work has mostly analyzed the data at the keyword level, to answer questions such as "how many tweets mention the word 'Obama' today?". In contrast, Kosmix aimed to analyze at the *semantic* level, to answer questions such as "how many tweets mention President Obama today?". To do this, we need to recognize that "Obama", "the pres", "BO", and "the messiah" for example can all refer to the same person.

Regarding knowledge bases, recent work has utilized Wikipedia (and other data sources) to build global ontology-like KBs. Well-known examples include Freebase [13], DBpedia [8,12], YAGO [37,38], and WolframAlpha [41]. These works however have not described the end-to-end process of building, maintaining, and using these KBs. In particular, they have not discussed converting the Wikipedia graph into a taxonomy (as we do here and in [18]). Finally, as far as we know, no work has addressed the problem of building a large real-time social KB, such as the Social Genome.

Entity extraction and classification of formal text has been widely studied for more than two decades (e.g., [4,5,17,24,26,28]), with competitions (e.g., CoNLL [35,40], MUC [39] and ACE [20]) and off-the-shelf tools (e.g., Stanford NER, OpenNLP, GATE, LingPipe). Entity extraction and classification for tweets, on the other hand, has been a less studied problem. Liu et al. [27] extracts only person, organization and location entities, while we do it for a large number of entity types with links to a knowledge base. Finn et al. [21] use crowdsourcing to annotate a large corpus of tweets. Recently Ritter et al. [34] have developed a NLP pipeline spanning POS tagging, chunking and named entity recognition and classification for tweets. SemTag and Seeker [19] perform automatic semantic tagging of a large Web corpus using an ontology. Industrial systems for entity extraction and classification include OpenCalais [31], AlchemyAPI [6], and Semantria [36]. Semantria does not have support for linked data whereas OpenCalais and AlchemyAPI do. OpenCalais additionally extracts relations, facts and events. The paper [23] empirically shows that our system outperforms OpenCalais in many aspects. Event and trend detection in social media have been

actively studied (e.g., [11, 33]). However, they typically do not use multiple heuristics, as we do, and few of them have considered scaling to the Twitter fire hose. Research has also addressed event monitoring but mostly in news stories (e.g., [7]). Our event monitoring work is perhaps most similar to Twitcident [3], but that work uses deep semantic techniques (based on named entity recognition) that tend to be error prone, and hence achieves limited accuracy.

Regarding scalable stream processing, recent works (e.g., [15, 30]) have extended MapReduce to incremental batch processing. However, they retain the MapReduce model, where a Reducer is a "blocking" operator, in that it still has to see *all* the necessary data from the Mappers before it can "reduce" and emit the final result. So this data has to be stored in the system. In contrast, MapUpdate uses slates to *summarize* the past data, so an updater can immediately process each event as the event comes in. This allows us to stream events through the system with millisecond to second latencies. Numerous stream processing systems have been developed in the database community [22] (e.g., Borealis; STREAM; Telegraph; SPADE for System S, commercialized as IBM InfoSphere Streams; Aurora, commercialized as StreamBase Systems; and Truviso). These systems often employ declarative query languages over data with schemas. In contrast, we make few assumptions about the data structure, and adopt a MapReduce style in which applications are decomposed into a procedural workflow of custom code. Second, much work has focused on optimizing query processing over data streams in an RDBMS style. In contrast, we focus on how to efficiently execute Maps and Updates over a cluster of machines, to achieve ultra-low-latency and high scalability. Our work is similar to S4 [29] and Storm [2]. These systems, however, leave it to the application to implement and manage its own state. Our experience suggests that this is highly non-trivial in many cases. In contrast, Muppet transparently manages application storage, which are slates in our case.

8 Concluding Remarks

In this paper we have described how Kosmix performed semantic analysis of social media, by extracting the most important entities, relationships, and events from the data. We believe that such semantic analysis will become increasingly important to a variety of applications, and that a key to perform them effectively is to leverage large-scale knowledge bases (such as the Kosmix KB), crowdsourcing (e.g., to clean up the knowledge bases and process the discovered events), as well as distributed stream processing infrastructure (such as Muppet).

References

[1] Muppet. Available as the open source system called Mupd8 github.com/walmartlabs/mupd8 (renamed to Mupd8 for legal reasons).

[2] Storm. `https://github.com/nathanmarz/storm`, 2011.

[3] F. Abel, C. Hauff, G. Houben, R. Stronkman, and K. Tao. Semantics + filtering + search = twitcident. exploring information in social web streams. In *HT*, 2012.

[4] E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *SIGKDD*, 2004.

[5] J. S. Aitken. Learning information extraction rules: An inductive logic programming approach. In *ECAI*, 2002.

[6] AlchemyAPI. http://www.alchemyapi.com/.

[7] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR*, 1998.

[8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *The Semantic Web*, 2007.

[9] N. Bansal, F. Chiang, N. Koudas, and F. Wm. Tompa. Seeking stable clusters in the blogosphere. In *VLDB*, 2007.

[10] N. Bansal and N. Koudas. Blogscope: A system for online analysis of high volume text streams. In *VLDB*, 2007.

[11] H. Becker, M. Naaman, and L. Gravano. Beyond trending topics: Real-world event identification on twitter. In *ICWSM*, 2011.

[12] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia- a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[13] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.
[14] C. Budak, D. Agrawal, and A. El Abbadi. Structural trend analysis for online social networks. *PVLDB*, 4(10):646–656, 2011.
[15] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. Mapreduce online. In *NSDI '10: 7th USENIX Symposium on Networked Systems Design and Implementation*, 2010.
[16] M. Das, S. Thirumuruganathan, S. Amer-Yahia, G. Das, and C. Yu. Who tags what? an analysis framework. *PVLDB*, 5(11):1567–1578, 2012.
[17] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. In *VLDB*, 2007.
[18] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Building, maintaining, and using knowledge bases: a report from the trenches. In *SIGMOD*, 2013.
[19] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *WWW*, 2003.
[20] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The automatic content extraction (ACE) program–tasks, data, and evaluation. In *LREC*, 2004.
[21] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze. Annotating named entities in Twitter data with crowdsourcing. In *NAACL HLT Workshop*, 2010.
[22] M. Garofalakis, J. Gehrke, and R. Rastogi (editors). *Data Stream Management*. Springer, 2009.
[23] A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Entity extraction, linking, classification, and tagging for social media: A wikipedia-based approach. In *VLDB*, 2013.
[24] J. Lafferty, A. McCallum, and F. CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.
[25] W. Lam, L. Liu, S. Prasad, A. Rajaraman, Z. Vacheri, and A. Doan. Muppet: Mapreduce-style processing of fast data. *PVLDB*, 5(12):1814–1825, 2012.
[26] W. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng, C. Dolan, and S. Goldman. UMass/Hughes: Description of the CIRCUS system used for MUC-5. In *MUC-5*, 1993.
[27] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *ACL HLT*, 2011.
[28] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *ICML*, 2000.
[29] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *ICDMW 2010, The 10th IEEE Int. Conf. on Data Mining Workshops*, 2010.
[30] C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V. B. N. Rao, V. Sankarasubramanian, S. Seth, C. Tian, T. ZiCornell, and X. Wang. Nova: Continuous pig/hadoop workflows. In *Proc. of SIGMOD '11*, 2011.
[31] OpenCalais. http://www.opencalais.com/.
[32] Y. Pavlidis, M. Mathihalli, I. Chakravarty, A. Batra, R. Benson, R. Raj, R. Yau, M. McKiernan, V. Harinarayan, and A. Rajaraman. Anatomy of a gift recommendation engine powered by social media. In *SIGMOD*, 2012.
[33] A. Popescu, M. Pennacchiotti, and D. Paranjpe. Extracting events and event descriptions from twitter. In *WWW (Companion Volume)*, 2011.
[34] A. Ritter, S. Clark, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *EMNLP*, 2011.
[35] T. K. Sang and F. Erik. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *NAACL*, 2002.
[36] Semantria. https://semantria.com/.
[37] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.
[38] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from Wikipedia and WordNet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217, 2008.
[39] B. M. Sundheim and N. A. Chinchor. Survey of the message understanding conferences. In *ACL HLT*, 1993.
[40] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *NAACL HLT*, 2003.
[41] S. Wolfram. Wolfram| Alpha. *http://www.wolframalpha.com*, 2009.