Open Challenges for Data Stream Mining Research

Georg Krempl
University Magdeburg, Germany
georg.krempl@iti.cs.uni-magdeburg.de

Indre Žliobaite
Aalto University and HIIT, Finland
indre.zliobaite@aalto.fi

Dariusz Brzeziński
Poznan U. of Technology, Poland
dariusz.brzezinski@cs.put.poznan.pl

Eyke Hüllermeier
University of Paderborn, Germany
eyke@upb.de

Mark Last
Ben-Gurion U. of the Negev, Israel
mlast@bgu.ac.il

Vincent Lemaire
Orange Labs, France
vincent.lemaire@orange.com

Tino Noack
TU Cottbus, Germany
noacktin@tu-cottbus.de

Ammar Shaker
University of Paderborn, Germany
ammar.shaker@upb.de

Sonja Sievi
Astrium Space Transportation, Germany
sonja.sievi@astrium.eads.net

Myra Spiliopoulou
University Magdeburg, Germany
myra@iti.cs.uni-magdeburg.de

Jerzy Stefanowski
Poznan U. of Technology, Poland
jerzy.stefanowski@cs.put.poznan.pl

ABSTRACT

Every day, huge volumes of sensory, transactional, and web data are continuously generated as streams, which need to be analyzed online as they arrive. Streaming data can be considered as one of the main sources of what is called big data. While predictive modeling for data streams and big data have received a lot of attention over the last decade, many research approaches are typically designed for well-behaved controlled problem settings, overlooking important challenges imposed by real-world applications. This article presents a discussion on eight open challenges for data stream mining. Our goal is to identify gaps between current research and meaningful applications, highlight open problems, and define new application-relevant research directions for data stream mining. The identified challenges cover the full cycle of knowledge discovery and involve such problems as: protecting data privacy, dealing with legacy systems, handling incomplete and delayed information, analysis of complex data, and evaluation of stream mining algorithms. The resulting analysis is illustrated by practical applications and provides general suggestions concerning lines of future research in data stream mining.

1. INTRODUCTION

The volumes of automatically generated data are constantly increasing. According to the Digital Universe Study [18], over 2.8ZB of data were created and processed in 2012, with a projected increase of 15 times by 2020. This growth in the production of digital data results from our surrounding environment being equipped with more and more sensors. People carrying smart phones produce data, database transactions are being counted and stored, streams of data are extracted from virtual environments in the form of logs or user generated content. A significant part of such data is volatile, which means it needs to be analyzed in real time as it arrives. Data stream mining is a research field that studies methods and algorithms for extracting knowledge from volatile streaming data [14; 5; 1]. Although data streams, online learning, big data, and adaptation to concept drift have become important research topics during the last decade, truly autonomous, self-maintaining, adaptive data mining systems are rarely reported. This paper identifies real-world challenges for data stream research that are important but yet unsolved. Our objective is to present to the community a position paper that could inspire and guide future research in data streams. This article builds upon discussions at the International Workshop on Real-World Challenges for Data Stream Mining (RealStream)[1] in September 2013, in Prague, Czech Republic.

Several related position papers are available. Dietterich [10] presents a discussion focused on predictive modeling techniques, that are applicable to streaming and non-streaming data. Fan and Bifet [12] concentrate on challenges presented by large volumes of data. Zliobaite et al. [48] focus on concept drift and adaptation of systems during online operation. Gaber et al. [13] discuss ubiquitous data mining with attention to collaborative data stream mining. In this paper, we focus on research challenges for streaming data inspired and required by real-world applications. In contrast to existing position papers, we raise issues connected not only with large volumes of data and concept drift, but also such practical problems as privacy constraints, availability of information, and dealing with legacy systems.

The scope of this paper is not restricted to algorithmic challenges, it aims at covering the full cycle of knowledge discovery from data (CRISP [40]), from understanding the context of the task, to data preparation, modeling, evaluation, and deployment. We discuss eight challenges: making models simpler, protecting privacy and confidentiality, dealing with legacy systems, stream preprocessing, timing and availability of information, relational stream mining, analyzing event data, and evaluation of stream mining algorithms. Figure 1 illustrates the positioning of these challenges in the CRISP cycle. Some of these apply to traditional (non-streaming) data mining as well, but they are critical in streaming environments. Along with further discussion of these challenges, we present our position where the forthcoming focus of research and development efforts should be directed to address these challenges.

In the remainder of the article, section 2 gives a brief introduction to data stream mining, sections 3–7 discuss each identified challenge, and section 8 highlights action points for future research.

[1] http://sites.google.com/site/realstream2013

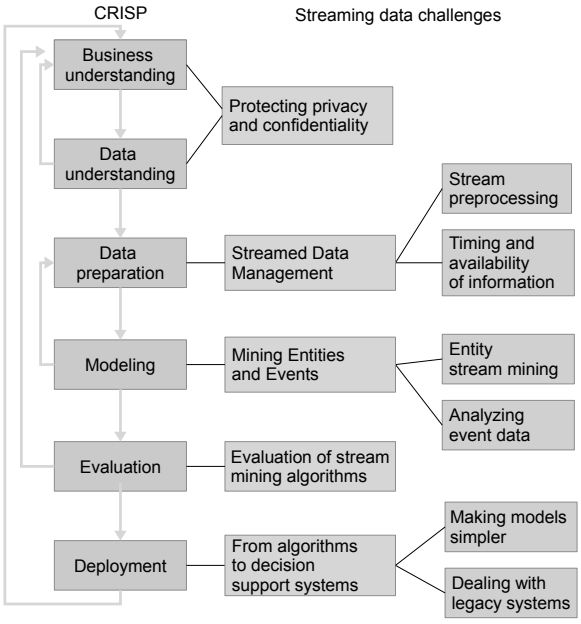


Figure 1: CRISP cycle with data stream research challenges.

2. DATA STREAM MINING

Mining big data streams faces three principal challenges: *volume*, *velocity*, and *volatility*. Volume and velocity require a high volume of data to be processed in limited time. Starting from the first arriving instance, the amount of available data constantly increases from zero to potentially infinity. This requires incremental approaches that incorporate information as it becomes available, and online processing if not all data can be kept [15]. Volatility, on the other hand, corresponds to a dynamic environment with ever-changing patterns. Here, old data is of limited use, even if it could be saved and processed again later. This is due to change, that can affect the induced data mining models in multiple ways: *change of the target variable*, *change in the available feature information*, and *drift*.

Changes of the target variable occur for example in credit scoring, when the definition of the classification target "default" versus "non-default" changes due to business or regulatory requirements. Changes in the available feature information arise when new features become available, e.g. due to a new sensor or instrument. Similarly, existing features might need to be excluded due to regulatory requirements, or a feature might change in its scale, if data from a more precise instrument becomes available. Finally, drift is a phenomenon that occurs when the distributions of features x and target variables y change in time. The challenge posed by drift has been subject to extensive research, thus we provide here solely a brief categorization and refer to recent surveys like [17].

In supervised learning, drift can affect the posterior $P(y|x)$, the conditional feature $P(x|y)$, the feature $P(x)$ and the class prior $P(y)$ distribution. The distinction based on which distribution is assumed to be affected, and which is assumed to be static, serves to assess the suitability of an approach for a particular task. It is worth noting, that the problem of changing distributions is also present in unsupervised learning from data streams.

A further categorization of drift can be made by:

- **smoothness of concept transition**: Transitions between concepts can be sudden or gradual. The former is sometimes also denoted in literature as shift or abrupt drift.
- **singular or recurring contexts**: In the former case, a model becomes obsolete once and for all when its context is replaced by a novel context. In the latter case, a model's context might reoccur at a later moment in time, for example due to a business cycle or seasonality, therefore, obsolete models might still regain value.
- **systematic or unsystematic**: In the former case, there are patterns in the way the distributions change that can be exploited to predict change and perform faster model adaptation. Examples are subpopulations that can be identified and show distinct, trackable evolutionary patterns. In the latter case, no such patterns exist and drift occurs seemingly at random. An example for the latter is fickle concept drift.
- **real or virtual**: While the former requires model adaptation, the latter corresponds to observing outliers or noise, which should not be incorporated into a model.

Stream mining approaches in general address the challenges posed by volume, velocity and volatility of data. However, in real-world applications these three challenges often coincide with other, to date insufficiently considered ones.

The next sections discuss eight identified challenges for data stream mining, providing illustrations with real world application examples, and formulating suggestions for forthcoming research.

3. PROTECTING PRIVACY AND CONFIDENTIALITY

Data streams present new challenges and opportunities with respect to protecting privacy and confidentiality in data mining. Privacy preserving data mining has been studied for over a decade (see. e.g. [3]). The main objective is to develop such data mining techniques that would not uncover information or patterns which compromise confidentiality and privacy obligations. Modeling can be done on original or anonymized data, but when the model is released, it should not contain information that may violate privacy or confidentiality. This is typically achieved by controlled distortion of sensitive data by modifying the values or adding noise.

Ensuring privacy and confidentiality is important for gaining trust of the users and the society in autonomous, stream data mining systems. While in offline data mining a human analyst working with the data can do a sanity check before releasing the model, in data stream mining privacy preservation needs to be done online. Several existing works relate to privacy preservation in publishing streaming data (e.g. [46]), but no systematic research in relation to broader data stream challenges exists.

We identify two main challenges for privacy preservation in mining data streams. The first challenge is incompleteness of information. Data arrives in portions and the model is updated online. Therefore, the model is never final and it is difficult to judge privacy preservation before seeing all the data. For example, suppose GPS traces of individuals are being collected for modeling traffic situation. Suppose person A at current time travels from the campus to the airport. The privacy of a person will be compromised, if there are no similar trips by other persons in the very near future. However, near future trips are unknown at the current time, when the model needs to be updated.

On the other hand, data stream mining algorithms may have some inherent privacy preservation properties due to the fact that they do not need to see all the modeling data at once, and can be incrementally updated with portions of data. Investigating privacy preservation properties of existing data stream algorithms makes another interesting direction for future research.

The second important challenge for privacy preservation is concept drift. As data may evolve over time, fixed privacy preservation rules may no longer hold. For example, suppose winter comes, snow falls, and much less people commute by bike. By knowing that a person comes to work by bike and having a set of GPS traces, it may not be possible to identify this person uniquely in summer, when there are many cyclists, but possible in winter. Hence, an important direction for future research is to develop adaptive privacy preservation mechanisms, that would diagnose such a situation and adapt themselves to preserve privacy in the new circumstances.

4. STREAMED DATA MANAGEMENT

Most of the data stream research concentrates on developing predictive models that address a simplified scenario, in which data is *already pre-processed*, *completely* and *immediately available* for *free*. However, successful business implementations depend strongly on the alignment of the used machine learning algorithms with both, the business objectives, and the available data. This section discusses often omitted challenges connected with streaming data.

4.1 Streamed Preprocessing

Data preprocessing is an important step in all real world data analysis applications, since data comes from complex environments, may be noisy, redundant, contain outliers and missing values. Many standard procedures for preprocessing offline data are available and well established, see e.g. [33]; however, the data stream setting introduces new challenges that have not received sufficient research attention yet.

While in traditional offline analysis data preprocessing is a once-off procedure, usually done by a human expert prior to modeling, in the streaming scenario manual processing is not feasible, as new data continuously arrives. Streaming data needs fully automated preprocessing methods, that can optimize the parameters and operate autonomously. Moreover, preprocessing models need to be able to update themselves automatically along with evolving data, in a similar way as predictive models for streaming data do. Furthermore, all updates of preprocessing procedures need to be synchronized with the subsequent predictive models, otherwise after an update in preprocessing the data representation may change and, as a result, the previously used predictive model may become useless.

Except for some studies, mainly focusing on feature construction over data streams, e.g. [49; 4], no systematic methodology for data stream preprocessing is currently available.

As an illustrative example for challenges related to data preprocessing, consider predicting traffic jams based on mobile sensing data. People using navigation services on mobile devices can opt to send anonymized data to the service provider. Service providers, such as Google, Yandex or Nokia, provide estimations and predictions of traffic jams based on this data. First, the data of each user is mapped to the road network, the speed of each user on each road segment of the trip is computed, data from multiple users is aggregated, and finally the current speed of the traffic is estimated.

There are a lot of data preprocessing challenges associated with this task. First, *noisiness* of GPS data might *vary* depending on location and load of the telecommunication network. There may be *outliers*, for instance, if somebody stopped in the middle of a segment to wait for a passenger, or a car broke. The number of pedestrians using mobile navigation may vary, and require *adaptive instance selection*. Moreover, road networks may change over time, leading to changes in average speeds, in the number of cars and even car types (e.g. heavy trucks might be banned, new optimal routes emerge). All these issues require automated preprocessing actions before feeding the newest data to the predictive models.

The problem of preprocessing for data streams is challenging due to the challenging nature of the data (continuously arriving and evolving). An analyst cannot know for sure, what kind of data to expect in the future, and cannot deterministically enumerate possible actions. Therefore, not only models, but also the procedure itself needs to be fully automated.

This research problem can be approached from several angles. One way is to look at existing predictive models for data streams, and try to integrate them with selected data preprocessing methods (e.g. feature selection, outlier definition and removal).

Another way is to systematically characterize the existing offline data preprocessing approaches, try to find a mapping between those approaches and problem settings in data streams, and extend preprocessing approaches for data streams in such a way as traditional predictive models have been extended for data stream settings.

In either case, developing individual methods and methodology for preprocessing of data streams would bridge an important gap in the practical applications of data stream mining.

4.2 Timing and Availability of Information

Most algorithms developed for evolving data streams make simplifying assumptions on the timing and availability of information. In particular, they assume that information is *complete*, *immediately available*, and *received passively and for free*. These assumptions often do not hold in real-world applications, e.g., patient monitoring, robot vision, or marketing [43]. This section is dedicated to the discussion of these assumptions and the challenges resulting from their absence. For some of these challenges, corresponding situations in offline, static data mining have already been addressed in literature. We will briefly point out where a mapping of such known solutions to the online, evolving stream setting is easily feasible, for example by applying windowing techniques. However, we will focus on problems for which no such simple mapping exists and which are therefore open challenges in stream mining.

4.2.1 Handling Incomplete Information

Completeness of information assumes that the true values of all variables, that is of features and of the target, are revealed eventually to the mining algorithm.

The problem of missing values, which corresponds to incompleteness of features, has been discussed extensively for the offline, static settings. A recent survey is given in [45]. However, only few works address data streams, and in particular *evolving* data streams. Thus several open challenges remain, some are pointed out in the review by [29]: how to address the problem that the frequency in which missing values occur is unpredictable, but largely affects the quality of imputations? How to (automatically) select the best imputation technique? How to proceed in the trade-off between speed and statistical accuracy?

Another problem is that of missing values of the target variable. It has been studied extensively in the static setting as semi-supervised learning (SSL, see [11]). A requirement for applying SSL techniques to streams is the availability of at least some labeled data from the most recent distribution. While first attempts to this problem have been made, e.g. the online manifold regularization approach in [19] and the ensembles-based approach suggested by [11], improvements in speed and the provision of performance guarantees remain open challenges. A special case of incomplete information is "censored data" in Event History Analysis (EHA), which is described in section 5.2. A related problem discussed below is active learning (AL, see [38]).

4.2.2 Dealing with Skewed Distributions

Class imbalance, where the class prior probability of the minority class is small compared to that of the majority class, is a frequent problem in real-world applications like fraud detection or credit scoring. This problem has been well studied in the offline setting (see e.g. [22] for a recent book on that subject), and has also been studied to some extent in the online, stream-based setting (see [23] for a recent survey). However, among the few existing stream-based approaches, most do not pay attention to drift of the minority class, and as [23] pointed out, a more rigorous evaluation of these algorithms on real-world data needs yet to be done.

4.2.3 Handling Delayed Information

Latency means information becomes available with significant delay. For example, in the case of so-called verification latency, the value of the preceding instance's target variable is not available before the subsequent instance has to be predicted. On *evolving* data streams, this is more than a mere problem of streaming data integration between feature and target streams, as due to concept drift patterns show temporal locality [2]. It means that feedback on the current prediction is not available to improve the subsequent predictions, but only eventually will become available for much later predictions. Thus, there is *no recent sample of labeled data at all* that would correspond to the most-recent unlabeled data, and semi-supervised learning approaches are not directly applicable.

A related problem in static, offline data mining is that addressed by unsupervised transductive transfer learning (or unsupervised domain adaptation): given labeled data from a source domain, a predictive model is sought for a related target domain in which no labeled data is available. In principle, ideas from transfer learning could be used to address latency in evolving data streams, for example by employing them in a chunk-based approach, as suggested in [43]. However, adapting them for use in evolving data streams has not been tried yet and constitutes a non-trivial, open task, as adaptation in streams must be fast and fully automated and thus cannot rely on iterated careful tuning by human experts.

Furthermore, consecutive chunks constitute several domains, thus the transitions between several subsequent chunks might provide exploitable patterns of systematic drift. This idea has been introduced in [27], and a few so-called *drift-mining* algorithms that identify and exploit such patterns have been proposed since then. However, the existing approaches cover only a very limited set of possible drift patterns and scenarios.

4.2.4 Active Selection from Costly Information

The challenge of intelligently selecting among costly pieces of information is the subject of active learning research. Active stream-based selective sampling [38] describes a scenario, in which instances arrive one-by-one. While the instances' feature vectors are provided for free, obtaining their true target values is costly, and the definitive decision whether or not to request this target value must be taken before proceeding to the next instance. This corresponds to a data stream, but *not necessarily* to an *evolving* one. As a result, only a small subset of stream-based selective sampling algorithms is suited for non-stationary environments. To make things worse, many contributions do not state explicitly whether they were designed for drift, neither do they provide experimental evaluations on such evolving data streams, thus leaving the reader the arduous task to assess their suitability for evolving streams. A first, recent attempt to provide an overview on the existing active learning strategies for evolving data streams is given in [43]. The challenges for active learning posed by evolving data streams are:

- **uncertainty regarding convergence**: in contrast to learning in static contexts, due to drift there is no guarantee that with additional labels the difference between model and reality narrows down. This leaves the formulation of suitable stop criteria a challenging open issue.
- **necessity of perpetual validation**: even if there has been convergence due to some temporary stability, the learned hypotheses can get invalidated at any time by subsequent drift. This can affect any part of the feature space and is not necessarily detectable from unlabeled data. Thus, without perpetual validation the mining algorithm might lock itself to a wrong hypothesis without ever noticing.
- **temporal budget allocation**: the necessity of perpetual validation raises the question of optimally allocating the labeling budget over time.
- **performance bounds**: in the case of drifting posteriors, no theoretical work exists that provides bounds for errors and label requests. However, deriving such bounds will also require assuming some type of systematic drift.

The task of active feature acquisition, where one has to actively select among costly features, constitutes another open challenge on evolving data streams: in contrast to the static, offline setting, the value of a feature is likely to change with its drifting distribution.

5. MINING ENTITIES AND EVENTS

Conventional stream mining algorithms learn over a single stream of arriving entities. In subsection 5.1, we introduce the paradigm of *entity stream mining*, where the entities constituting the stream are linked to instances (structured pieces of information) from further streams. Model learning in this paradigm involves the incorporation of the streaming information into the stream of entities; learning tasks include cluster evolution, migration of entities from one state to another, classifier adaptation as entities re-appear with another label than before.

Then, in subsection 5.2, we investigate the special case where entities are associated with the occurrence of events. Model learning then implies identifying the moment of occurrence of an event on an entity. This scenario might be seen as a special case of entity stream mining, since an event can be seen as a degenerate instance consisting of a single value (the event's occurrence).

5.1 Entity Stream Mining

Let T be a stream of entities, e.g. customers of a company or patients of a hospital. We observe entities over time, e.g. on a company's website or at a hospital admission vicinity: an entity appears and re-appears at discrete time points, new entities show up. At a time point t, an entity $e \in T$ is linked with different pieces of information - the purchases and ratings performed by a customer, the anamnesis, the medical tests and the diagnosis recorded for the patient. Each of these information pieces $i_j(t)$ is a structured record or an unstructured text from a stream T_j, linked to e via the foreign key relation. Thus, the entities in T are in 1-to-1 or 1-to-n relation with entities from further streams $T_1, \ldots, T_m$ (stream of purchases, stream of ratings, stream of complaints etc). The schema describing the streams $T, T_1, \ldots, T_m$ can be perceived as a conventional relational schema, except that it describes streams instead of static sets.

In this relational setting, the *entity stream mining* task corresponds to learning a model ζ_T over T, thereby incorporating information from the adjoint streams $T_1, \ldots, T_m$ that "feed" the entities in T.

Albeit the members of each stream are entities, we use the term "entity" only for stream T – the target of learning, while we denote the entities in the other streams as "instances". In the unsupervised setting, entity stream clustering encompasses learning and adapting clusters over T, taking account the other streams that arrive at different speeds. In the supervised setting, entity stream classification involves learning and adapting a classifier, notwithstanding the fact that an entity's label may change from one time point to the next, as new instances referencing it arrive.

5.1.1 Challenges of Aggregation

The first challenge of entity stream mining task concerns information summarization: how to aggregate into each entity e at each time point t the information available on it from the other streams? What information should be stored for each entity? How to deal with differences in the speeds of the individual streams? How to learn over the streams efficiently? Answering these questions in a seamless way would allow us to deploy conventional stream mining methods for entity stream mining after aggregation.

The information referencing a relational entity cannot be held perpetually for learning, hence aggregation of the arriving streams is necessary. Information aggregation over time-stamped data is traditionally practiced in document stream mining, where the objective is to derive and adapt content summaries on learned topics. Content summarization on entities, which are referenced in the document stream, is studied by Kotov et al., who maintain for each entity the number of times it is mentioned in the news [26].

In such studies, summarization is a task by itself. Aggregation of information for subsequent learning is a bit more challenging, because summarization implies information loss - notably information about the evolution of an entity. Hassani and Seidl monitor health parameters of patients, modeling the stream of recordings on a patient as a sequence of events [21]: the learning task is then to predict forthcoming values. Aggregation with selective forgetting of past information is proposed in [25; 42] in the classification context: the former method [25] slides a window over the stream, while the latter [42] forgets entities that have not appeared for a while, and summarizes the information in frequent itemsets, which are then used as new features for learning.

5.1.2 Challenges of Learning

Even if information aggregation over the streams $T_1, \ldots, T_m$ is performed intelligently, entity stream mining still calls for more than conventional stream mining methods. The reason is that entities of stream T re-appear in the stream and evolve. In particular, in the unsupervised setting, an entity may be linked to conceptually different instances at each time point, e.g. reflecting a customer's change in preferences. In the supervised setting, an entity may change its label; for example, a customer's affinity to risk may change in response to market changes or to changes in family status. This corresponds to *entity drift*, i.e. a new type of drift beyond the conventional concept drift pertaining to model ζ_T. Hence, how should entity drift be traced, and how should the interplay between entity drift and model drift be captured?

In the unsupervised setting, Oliveira and Gama learn and monitor clusters as *states* of evolution [32], while [41] extend that work to learn Markov chains that mark the entities' evolution. As pointed out in [32], these states are not necessarily predefined – they must be subject of learning. In [43], we report on further solutions to the entity evolution problem and to the problem of learning with forgetting over multiple streams and over the entities referenced by them.

Conventional concept drift also occurs when learning a model over entities, thus the challenges pertinent to stream mining also apply here. One of these challenges, and one much discussed in the context of big data, is *volatility*. In relational stream mining, volatility refers to the entity itself, not only to the stream of instances that reference the entities. Finally, an entity is ultimately *big data* by itself, since it is described by multiple streams. Hence, next to the problem of dealing with new forms of learning and new aspects of drift, the subject of efficient learning and adaption in the Big Data context becomes paramount.

5.2 Analyzing Event Data

Events are an example for data that occurs often yet is rarely analyzed in the stream setting. In static environments, events are usually studied through *event history analysis* (EHA), a statistical method for modeling and analyzing the temporal distribution of events related to specific objects in the course of their lifetime [9]. More specifically, EHA is interested in the duration before the occurrence of an event or, in the *recurrent* case (where the same event can occur repeatedly), the duration between two events. The notion of an *event* is completely generic and may indicate, for example, the failure of an electrical device. The method is perhaps even better known as *survival analysis*, a term that originates from applications in medicine, in which an event is the death of a patient and *survival time* is the time period between the beginning of the study and the occurrence of this event. EHA can also be considered as a special case of entity stream mining described in section 5.1, because the basic statistical entities in EHA are monitored objects (or subjects), typically described in terms of feature vectors $\boldsymbol{x} \in \mathbb{R}^n$, together with their survival time s. Then, the goal is to model the dependence of s on $\boldsymbol{x}$. A corresponding model provides hints at possible cause-effect relationships (e.g., what properties tend to increase a patient's survival time) and, moreover, can be used for predictive purposes (e.g., what is the expected survival time of a patient).

Although one might be tempted to approach this modeling task as a standard regression problem with input (regressor) $\boldsymbol{x}$ and output (response) s, it is important to notice that the survival time s is normally not observed for all objects. Indeed, the problem of *censoring* plays an important role in EHA and occurs in different facets. In particular, it may happen that some of the objects survived till the end of the study at time t_{end} (also called the *cut-off point*). They are censored or, more specifically, *right censored*, since t_{event} has not been observed for them; instead, it is only known that $t_{event} > t_{end}$. In *snapshot monitoring* [28], the data stream may be sampled multiple times, resulting in a new cut-off point for each snapshot. Unlike standard regression analysis, EHA is specifically tailored for analyzing event data of that kind. It is built upon the *hazard function* as a basic mathematical tool.

5.2.1 Survival function and hazard rate

Suppose the time of occurrence of the next event (since the start or the last event) for an object $\boldsymbol{x}$ is modeled as a real-valued random variable T with probability density function $f(\cdot \,|\, \boldsymbol{x})$. The hazard function or hazard rate $h(\cdot \,|\, \boldsymbol{x})$ models the *propensity* of the occurrence of an event, that is, the marginal probability of an event to occur at time t, given that no event has occurred so far:

$$h(t \,|\, \boldsymbol{x}) = \frac{f(t \,|\, \boldsymbol{x})}{S(t \,|\, \boldsymbol{x})} = \frac{f(t \,|\, \boldsymbol{x})}{1 - F(t \,|\, \boldsymbol{x})} \ ,$$

where $S(\cdot \,|\, \boldsymbol{x})$ is the survival function and $F(\cdot \,|\, \boldsymbol{x})$ the cumulative distribution of $f(\cdot \,|\, \boldsymbol{x})$. Thus,

$$F(t \,|\, \boldsymbol{x}) \ = \ \mathbf{P}(T \leq t) = \int_0^t f(u \,|\, \boldsymbol{x}) \, du$$

is the probability of an event to occur before time t. Correspondingly, $S(t \mid \boldsymbol{x}) = 1 - F(t \mid \boldsymbol{x})$ is the probability that the event did not occur until time t (the survival probability). It can hence be used to model the probability of the right-censoring of the time for an event to occur.

A simple example is the Cox proportional hazard model [9], in which the hazard rate is constant over time; thus, it does depend on the feature vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ but not on time t. More specifically, the hazard rate is modeled as a log-linear function of the features x_i:

$$h(t \mid \boldsymbol{x}) = \lambda(\boldsymbol{x}) = \exp\left(\boldsymbol{x}^\top \beta\right)$$

The model is proportional in the sense that increasing x_i by one unit increases the hazard rate $\lambda(\boldsymbol{x})$ by a factor of $\alpha_i = \exp(\beta_i)$. For this model, one easily derives the survival function $S(t \mid \boldsymbol{x}) = 1 - \exp(-\lambda(\boldsymbol{x}) \cdot t)$ and an expected survival time of $1/\lambda(\boldsymbol{x})$.

5.2.2 EHA on data streams

Although the temporal nature of event data naturally fits the data stream model and, moreover, event data is naturally produced by many data sources, EHA has been considered in the data stream scenario only very recently. In [39], the authors propose a method for analyzing earthquake and Twitter data, namely an extension of the above Cox model based on a sliding window approach. The authors of [28] modify standard classification algorithms, such as decision trees, so that they can be trained on a snapshot stream of both censored and non-censored data.

Like in the case of clustering [35], where one distinguishes between clustering observations and clustering data sources, two different settings can be envisioned for EHA on data streams:

1. In the first setting, events are generated by multiple data sources (representing monitored objects), and the features pertain to these sources; thus, each data source is characterized by a feature vector $\boldsymbol{x}$ and produces a stream of (recurrent) events. For example, data sources could be users in a computer network, and an event occurs whenever a user sends an email.

2. In the second setting, events are produced by a single data source, but now the events themselves are characterized by features. For example, events might be emails sent by an email server, and each email is represented by a certain set of properties.

Statistical event models on data streams can be used in much the same way as in the case of static data. For example, they can serve predictive purposes, i.e., to answer questions such as "How much time will elapse before the next email arrives?" or "What is the probability to receive more than 100 emails within the next hour?". What is specifically interesting, however, and indeed distinguishes the data stream setting from the static case, is the fact that the model may change over time. This is a subtle aspect, because the hazard model $h(t \mid \boldsymbol{x})$ itself may already be time-dependent; here, however, t is not the absolute time but the duration time, i.e., the time elapsed since the last event. A change of the model is comparable to concept drift in classification, and means that the way in which the hazard rate depends on time t and on the features x_i changes over time. For example, consider the event "increase of a stock rate" and suppose that $\beta_i = \log(2)$ for the binary feature $x_i = \texttt{energy sector}$ in the above Cox model (which, as already mentioned, does not depend on t). Thus, this feature doubles the hazard rate and hence halves the expected duration between two events. Needless to say, however, this influence may change over time, depending on how well the energy sector is doing.

Dealing with model changes of that kind is clearly an important challenge for event analysis on data streams. Although the problem is to some extent addressed by the works mentioned above, there is certainly scope for further improvement, and for using these approaches to derive predictive models from censored data. Besides, there are many other directions for future work. For example, since the *detection* of events is a main prerequisite for analyzing them, the combination of EHA with methods for *event detection* [36] is an important challenge. Indeed, this problem is often far from trivial, and in many cases, events (such as frauds, for example) can only be detected with a certain time delay; dealing with delayed events is therefore another important topic, which was also discussed in section 4.2.

6. EVALUATION OF DATA STREAM ALGORITHMS

All of the aforementioned challenges are milestones on the road to better algorithms for real-world data stream mining systems. To verify if these challenges are met, practitioners need tools capable of evaluating newly proposed solutions. Although in the field of static classification such tools exist, they are insufficient in data stream environments due to such problems as: concept drift, limited processing time, verification latency, multiple stream structures, evolving class skew, censored data, and changing misclassification costs. In fact, the myriad of additional complexities posed by data streams makes algorithm evaluation a highly multi-criterial task, in which optimal trade-offs may change over time.

Recent developments in applied machine learning [6] emphasize the importance of understanding the data one is working with and using evaluation metrics which reflect its difficulties. As mentioned before, data streams set new requirements compared to traditional data mining and researchers are beginning to acknowledge the shortcomings of existing evaluation metrics. For example, Gama et al. [16] proposed a way of calculating classification accuracy using only the most recent stream examples, therefore allowing for time-oriented evaluation and aiding concept drift detection. Methods which test the classifier's robustness to drifts and noise on a practical, experimental level are also starting to arise [34; 47]. However, all these evaluation techniques focus on single criteria such as prediction accuracy or robustness to drifts, even though data streams make evaluation a constant trade-off between several criteria [7]. Moreover, in data stream environments there is a need for more advanced tools for visualizing changes in algorithm predictions with time.

The problem of creating complex evaluation methods for stream mining algorithms lies mainly in the size and evolving nature of data streams. It is much more difficult to estimate and visualize, for example, prediction accuracy if evaluation must be done online, using limited resources, and the classification task changes with time. In fact, the algorithm's ability to adapt is another aspect which needs to be evaluated, although information needed to perform such evaluation is not always available. Concept drifts are known in advance mainly when using synthetic or benchmark data, while in more practical scenarios occurrences and types of concepts are not directly known and only the label of each arriving instance is known. Moreover, in many cases the task is more complicated, as labeling information is not instantly available. Other difficulties in evaluation include processing complex relational streams and coping with class imbalance when class distributions evolve with time. Finally, not only do we need measures for evaluating single aspects of stream mining algorithms, but also ways of combining several of these aspects into global evaluation models, which would take into

account expert knowledge and user preferences.

Clearly, evaluation of data stream algorithms is a fertile ground for novel theoretical and algorithmic solutions. In terms of prediction measures, data stream mining still requires evaluation tools that would be immune to class imbalance and robust to noise. In our opinion, solutions to this problem should involve not only metrics based on relative performance to baseline (chance) classifiers, but also graphical measures similar to PR-curves or cost curves. Furthermore, there is a need for integrating information about concept drifts in the evaluation process. As mentioned earlier, possible ways of considering concept drifts will depend on the information that is available. If true concepts are known, algorithms could be evaluated based on: how often they detect drift, how early they detect it, how they react to it, and how quickly they recover from it. Moreover, in this scenario, evaluation of an algorithm should be dependent on whether it takes place during drift or during times of concept stability. A possible way of tackling this problem would be the proposal of graphical methods, similar to ROC analysis, which would work online and visualize concept drift measures alongside prediction measures. Additionally, these graphical measures could take into account the state of the stream, for example, its speed, number of missing values, or class distribution. Similar methods could be proposed for scenarios where concepts are not known in advance, however, in these cases measures should be based on drift detectors or label-independent stream statistics. Above all, due to the number of aspects which need to be measured, we believe that the evaluation of data stream algorithms requires a multi-criterial view. This could be done by using inspirations from multiple criteria decision analysis, where trade-offs between criteria are achieved using user-feedback. In particular, a user could showcase his/her criteria preferences (for example, between memory consumption, accuracy, reactivity, self-tuning, and adaptability) by deciding between alternative algorithms for a given data stream. It is worth noticing that such a multi-criterial view on evaluation is difficult to encapsulate in a single number, as it is usually done in traditional offline learning. This might suggest that researchers in this area should turn towards semi-qualitative and semi-quantitative evaluation, for which systematic methodologies should be developed.

Finally, a separate research direction involves rethinking the way we test data stream mining algorithms. The traditional train, cross-validate, test workflow in classification is not applicable for sequential data, which makes, for instance, parameter tuning much more difficult. Similarly, ground truth verification in unsupervised learning is practically impossible in data stream environments. With these problems in mind, it is worth stating that there is still a shortage of real and synthetic benchmark datasets. Such a situation might be a result of non-uniform standards for testing algorithms on streaming data. As community, we should decide on such matters as: What characteristics should benchmark datasets have? Should they have prediction tasks attached? Should we move towards online evaluation tools rather than datasets? These questions should be answered in order to solve evaluation issues in controlled environments before we create measures for real-world scenarios.

7. FROM ALGORITHMS TO DECISION SUPPORT SYSTEMS

While a lot of algorithmic methods for data streams are already available, their deployment in real applications with real streaming data presents a new dimension of challenges. This section points out two such challenges: making models simpler and dealing with legacy systems.

7.1 Making models simpler, more reactive, and more specialized

In this subsection, we discuss aspects like the simplicity of a model, its proper combination of offline and online components, and its customization to the requirements of the application domain. As an application example, consider the French Orange Portal[2], which registers millions of visits daily. Most of these visitors are only known through anonymous cookie IDs. For all of these visitors, the portal has the ambition to provide specific and relevant contents as well as printing ads for targeted audiences. Using information about visits on the portal the questions are: what part of the portal does each cookie visit, and when and which contents did it consult, what advertisement was sent, when (if) was it clicked. All this information generates hundreds of gigabytes of data each week. A user profiling system needs to have a back end part to preprocess the information required at the input of a front end part, which will compute appetency to advertising (for example) using stream mining techniques (in this case a supervised classifier). Since the ads to print change regularly, based on marketing campaigns, the extensive parameter tuning is infeasible as one has to react quickly to change. Currently, these tasks are either solved using bandit methods from game theory [8], which impairs adaptation to drift, or done offline in big data systems, resulting in slow reactivity.

7.1.1 Minimizing parameter dependence

Adaptive predictive systems are intrinsically parametrized. In most of the cases, setting these parameters, or tuning them is a difficult task, which in turn negatively affects the usability of these systems. Therefore, it is strongly desired for the system to have as few user adjustable parameters as possible. Unfortunately, the state of the art does not produce methods with trustworthy or easily adjustable parameters. Moreover, many predictive modeling methods use a lot of parameters, rendering them particularly impractical for data stream applications, where models are allowed to evolve over time, and input parameters often need to evolve as well.

The process of predictive modeling encompasses fitting of parameters on a training dataset and subsequently selecting the best model, either by heuristics or principled methods. Recently, model selection methods have been proposed that do not require internal cross-validation, but rather use the Bayesian machinery to design regularizers with data dependent priors [20]. However, they are not yet applicable in data streams, as their computational time complexity is too high and they require all examples to be kept in memory.

7.1.2 Combining offline and online models

Online and offline learning are mostly considered as mutually exclusive, but it is their combination that might enhance the value of data the most. Online learning, which processes instances one-by-one and builds models incrementally, has the virtue of being fast, both in the processing of data and in the adaptation of models. Offline (or batch) learning has the advantage of allowing the use of more sophisticated mining techniques, which might be more time-consuming or require a human expert. While the first allows the processing of "*fast* data" that requires real-time processing and adaptivity, the second allows processing of "*big* data" that requires longer processing time and larger abstraction.

Their combination can take place in many steps of the mining process, such as the data preparation and the preprocessing steps. For example, offline learning on big data could extract fundamental and sustainable trends from data using batch processing and massive parallelism. Online learning could then take real-time decisions

[2] www.orange.fr

from online events to optimize an immediate pay-off. In the online advertisement application mentioned above, the user-click prediction is done within a context, defined for example by the currently viewed page and the profile of the cookie. The decision which banner to display is done online, but the context can be preprocessed offline. By deriving meta-information such as "the profile is a young male, the page is from the sport cluster", the offline component can ease the online decision task.

7.1.3 Solving the right problem

Domain knowledge may help to solve many issues raised in this paper, by systematically exploiting particularities of application domains. However, this is seldom considered, as typical data stream methods are created to deal with a large variety of domains. For instance, in some domains the learning algorithm receives only partial feedback upon its prediction, i.e. a single bit of right-or-wrong, rather than the true label. In the user-click prediction example, if a user does not click on a banner, we do not know which one would have been correct, but solely that the displayed one was wrong. This is related to the issues on timing and availability of information discussed in section 4.2.

However, building predictive models that systematically incorporate domain knowledge or domain specific information requires to choose the right optimization criteria. As mentioned in section 6, the data stream setting requires optimizing multiple criteria simultaneously, as optimizing only predictive performance is not sufficient. We need to develop learning algorithms, which minimize an objective function including intrinsically and simultaneously: memory consumption, predictive performance, reactivity, self monitoring and tuning, and (explainable) auto-adaptivity. Data streams research is lacking methodologies for forming and optimizing such criteria.

Therefore, models should be simple so that they do not depend on a set of carefully tuned parameters. Additionally, they should combine offline and online techniques to address challenges of big and fast data, and they should solve the right problem, which might consist in solving a multi-criteria optimization task. Finally, they have to be able to learn from a small amount of data and with low variance [37], to react quickly to drift.

7.2 Dealing with Legacy Systems

In many application environments, such as financial services or health care systems, business critical applications are in operation for decades. Since these applications produce massive amounts of data, it becomes very promising to process these amounts of data by real-time stream mining approaches. However, it is often impossible to change existing infrastructures in order to introduce fully fledged stream mining systems. Rather than changing existing infrastructures, approaches are required that integrate stream mining techniques into legacy systems. In general, problems concerning legacy systems are domain-specific and encompass both technical and procedural issues. In this section, we analyze challenges posed by a specific real-world application with legacy issues — the ISS Columbus spacecraft module.

7.2.1 ISS Columbus

Spacecrafts are very complex systems, exposed to very different physical environments (e.g. space), and associated to ground stations. These systems are under constant and remote monitoring by means of telemetry and commands. The ISS Columbus module has been in operation for more than 5 years. For some time, it is pointed out that the monitoring process is not as efficient as previously expected [30]. However, we assume that data stream mining can make a decisive contribution to enhance and facilitate the required monitoring tasks. Recently, we are planning to use the ISS Columbus module as a technology demonstrator for integrating data stream processing and mining into the existing monitoring processes [31]. Figure 2 exemplifies the failure management system (FMS) of the ISS Columbus module. While it is impossible to simply redesign the FMS from scratch, we can outline the following challenges.

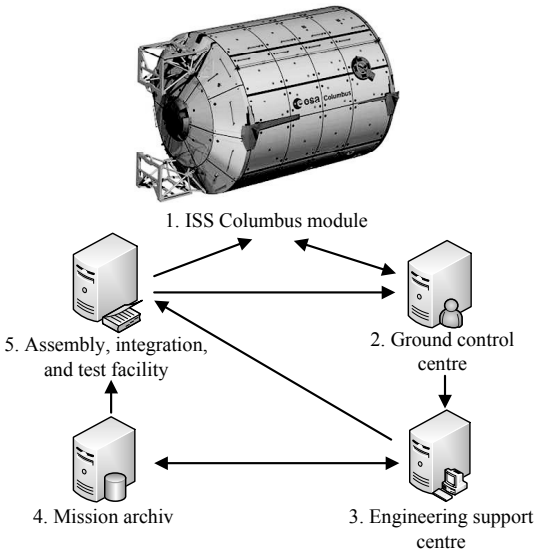


Figure 2: ISS Columbus FMS

7.2.2 Complexity

Even though spacecraft monitoring is very challenging by itself, it becomes increasingly difficult and complex due to the integration of data stream mining into such legacy systems. However, it was assumed to enhance and facilitate current monitoring processes. Thus, appropriate mechanism are required to integrate data stream mining into the current processes to decrease complexity.

7.2.3 Interlocking

As depicted in Figure 2, the ISS Columbus module is connected to ground instances. Real-time monitoring must be applied aboard where computational resources are restricted (e.g. processor speed and memory or power consumption). Near real-time monitoring or long-term analysis must be applied on-ground where the downlink suffers from latencies because of a long transmission distance, is subject to bandwidth limitations, and continuously interrupted due to loss of signal. Consequently, new data stream mining mechanisms are necessary which ensure a smooth interlocking functionality of aboard and ground instances.

7.2.4 Reliability and Balance

The reliability of spacecrafts is indispensable for astronauts' health and mission success. Accordingly, spacecrafts pass very long and expensive planning and testing phases. Hence, potential data stream mining algorithms must ensure reliability and the integration of such algorithms into legacy systems must not cause critical side effects. Furthermore, data stream mining is an automatic process which neglects interactions with human experts, while spacecraft monitoring is a semi-automatic process and human experts (e.g.

the flight control team) are responsible for decisions and consequent actions. This problem poses the following question: How to integrate data stream mining into legacy systems when automation needs to be increased but the human expert needs to be maintained in the loop? Abstract discussions on this topic are provided by expert systems [44] and the MAPE-K reference model [24]. Expert systems aim to combine human expertise with artificial expertise and the MAPE-K reference model aims to provide an autonomic control loop. A balance must be struck which considers both aforementioned aspects appropriately.

Overall, the Columbus study has shown that extending legacy systems with real time data stream mining technologies is feasible and it is an important area for further stream-mining research.

8. CONCLUDING REMARKS

In this paper, we discussed research challenges for data streams, originating from real-world applications. We analyzed issues concerning privacy, availability of information, relational and event streams, preprocessing, model complexity, evaluation, and legacy systems. The discussed issues were illustrated by practical applications including GPS systems, Twitter analysis, earthquake predictions, customer profiling, and spacecraft monitoring. The study of real-world problems highlighted shortcomings of existing methodologies and showcased previously unaddressed research issues.

Consequently, we call the data stream mining community to consider the following action points for data stream research:

- developing methods for ensuring privacy with incomplete information as data arrives, while taking into account the evolving nature of data;
- considering the availability of information by developing models that handle incomplete, delayed and/or costly feedback;
- taking advantage of relations between streaming entities;
- developing event detection methods and predictive models for censored data;
- developing a systematic methodology for streamed preprocessing;
- creating simpler models through multi-objective optimization criteria, which consider not only accuracy, but also computational resources, diagnostics, reactivity, interpretability;
- establishing a multi-criteria view towards evaluation, dealing with absence of the ground truth about how data changes;
- developing online monitoring systems, ensuring reliability of any updates, and balancing the distribution of resources.

As our study shows, there are challenges in every step of the CRISP data mining process. To date, modeling over data streams has been viewed and approached as an extension of traditional methods. However, our discussion and application examples show that in many cases it would be beneficial to step aside from building upon existing offline approaches, and start blank considering what is required in the stream setting.

Acknowledgments

We would like to thank the participants of the RealStream2013 workshop at ECMLPKDD2013 in Prague, and in particular Bernhard Pfahringer and George Forman, for suggestions and discussions on the challenges in stream mining. Part of this work was funded by the German Research Foundation, projects SP 572/11-1 (IMPRINT) and HU 1284/5-1, the Academy of Finland grant 118653 (ALGODAN), and the Polish National Science Center grants DEC-2011/03/N/ST6/00360 and DEC-2013/11/B/ST6/00963.

9. REFERENCES

[1] C. Aggarwal, editor. *Data Streams: Models and Algorithms*. Springer, 2007.

[2] C. Aggarwal and D. Turaga. Mining data streams: Systems and algorithms. In *Machine Learning and Knowledge Discovery for Engineering Systems Health Management*, pages 4–32. Chapman and Hall, 2012.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. *SIGMOD Rec.*, 29(2):439–450, 2000.

[4] C. Anagnostopoulos, N. Adams, and D. Hand. Deciding what to observe next: Adaptive variable selection for regression in multivariate data streams. In *Proc. of the 2008 ACM Symp. on Applied Computing*, SAC, pages 961–965, 2008.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS, pages 1–16, 2002.

[6] C. Brodley, U. Rebbapragada, K. Small, and B. Wallace. Challenges and opportunities in applied machine learning. *AI Magazine*, 33(1):11–24, 2012.

[7] D. Brzezinski and J. Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Trans. on Neural Networks and Learning Systems.*, 25:81–94, 2014.

[8] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal. Mortal multi-armed bandits. In *Proc. of the 22nd Conf. on Neural Information Processing Systems*, NIPS, pages 273–280, 2008.

[9] D. Cox and D. Oakes. *Analysis of Survival Data*. Chapman & Hall, London, 1984.

[10] T. Dietterich. Machine-learning research. *AI Magazine*, 18(4):97–136, 1997.

[11] G. Ditzler and R. Polikar. Semi-supervised learning in non-stationary environments. In *Proc. of the 2011 Int. Joint Conf. on Neural Networks*, IJCNN, pages 2741 – 2748, 2011.

[12] W. Fan and A. Bifet. Mining big data: current status, and forecast to the future. *SIGKDD Explorations*, 14(2):1–5, 2012.

[13] M. Gaber, J. Gama, S. Krishnaswamy, J. Gomes, and F. Stahl. Data stream mining in ubiquitous environments: state-of-the-art and current directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(2):116 – 138, 2014.

[14] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. *SIGMOD Rec.*, 34(2):18–26, 2005.

[15] J. Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 2010.

[16] J. Gama, R. Sebastiao, and P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[17] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept-drift adaptation. *ACM Computing Surveys*, 46(4), 2014.

[18] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, December 2012.

[19] A. Goldberg, M. Li, and X. Zhu. Online manifold regularization: A new learning setting and empirical study. In *Proc. of the European Conf. on Machine Learning and Principles of Knowledge Discovery in Databases*, ECMLPKDD, pages 393–407, 2008.

[20] I. Guyon, A. Saffari, G. Dror, and G. Cawley. Model selection: Beyond the bayesian/frequentist divide. *Journal of Machine Learning Research*, 11:61–87, 2010.

[21] M. Hassani and T. Seidl. Towards a mobile health context prediction: Sequential pattern mining in multiple streams. In *Proc. of , IEEE Int. Conf. on Mobile Data Management*, MDM, pages 55–57, 2011.

[22] H. He and Y. Ma, editors. *Imbalanced Learning: Foundations, Algorithms, and Applications*. IEEE, 2013.

[23] T. Hoens, R. Polikar, and N. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012.

[24] IBM. An architectural blueprint for autonomic computing. Technical report, IBM, 2003.

[25] E. Ikonomovska, K. Driessens, S. Dzeroski, and J. Gama. Adaptive windowing for online learning from multiple interrelated data streams. In *Proc. of the 11th IEEE Int. Conf. on Data Mining Workshops*, ICDMW, pages 697–704, 2011.

[26] A. Kotov, C. Zhai, and R. Sproat. Mining named entities with temporally correlated bursts from multilingual web news streams. In *Proc. of the 4th ACM Int. Conf. on Web Search and Data Mining*, WSDM, pages 237–246, 2011.

[27] G. Krempl. The algorithm APT to classify in concurrence of latency and drift. In *Proc. of the 10th Int. Conf. on Advances in Intelligent Data Analysis*, IDA, pages 222–233, 2011.

[28] M. Last and H. Halpert. Survival analysis meets data stream mining. In *Proc. of the 1st Worksh. on Real-World Challenges for Data Stream Mining*, RealStream, pages 26–29, 2013.

[29] F. Nelwamondo and T. Marwala. Key issues on computational intelligence techniques for missing data imputation - a review. In *Proc. of World Multi Conf. on Systemics, Cybernetics and Informatics*, volume 4, pages 35–40, 2008.

[30] E. Noack, W. Belau, R. Wohlgemuth, R. Müller, S. Palumberi, P. Parodi, and F. Burzagli. Efficiency of the columbus failure management system. In *Proc. of the AIAA 40th Int. Conf. on Environmental Systems*, 2010.

[31] E. Noack, A. Luedtke, I. Schmitt, T. Noack, E. Schaumlöffel, E. Hauke, J. Stamminger, and E. Frisk. The columbus module as a technology demonstrator for innovative failure management. In *German Air and Space Travel Congress*, 2012.

[32] M. Oliveira and J. Gama. A framework to monitor clusters evolution applied to economy and finance problems. *Intelligent Data Analysis*, 16(1):93–111, 2012.

[33] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers Inc., 1999.

[34] T. Raeder and N. Chawla. Model monitor (m^2): Evaluating, comparing, and monitoring models. *Journal of Machine Learning Research*, 10:1387–1390, 2009.

[35] P. Rodrigues and J. Gama. Distributed clustering of ubiquitous data streams. *WIREs Data Mining and Knowledge Discovery*, pages 38–54, 2013.

[36] T. Sakaki, M. Okazaki, and Y. Matsuo. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Trans. on Knowledge and Data Engineering*, 25(4):919–931, 2013.

[37] C. Salperwyck and V. Lemaire. Learning with few examples: An empirical study on leading classifiers. In *Proc. of the 2011 Int. Joint Conf. on Neural Networks*, IJCNN, pages 1010–1019, 2011.

[38] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[39] A. Shaker and E. Hüllermeier. Survival analysis on data streams: Analyzing temporal events in dynamically changing environments. *Int. Journal of Applied Mathematics and Computer Science*, 24(1):199–212, 2014.

[40] C. Shearer. The CRISP-DM model: the new blueprint for data mining. *J Data Warehousing*, 2000.

[41] Z. Siddiqui, M. Oliveira, J. Gama, and M. Spiliopoulou. Where are we going? predicting the evolution of individuals. In *Proc. of the 11th Int. Conf. on Advances in Intelligent Data Analysis*, IDA, pages 357–368, 2012.

[42] Z. Siddiqui and M. Spiliopoulou. Classification rule mining for a stream of perennial objects. In *Proc. of the 5th Int. Conf. on Rule-based Reasoning, Programming, and Applications*, RuleML, pages 281–296, 2011.

[43] M. Spiliopoulou and G. Krempl. Tutorial "mining multiple threads of streaming data". In *Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, PAKDD, 2013.

[44] D. Waterman. *A Guide to Expert Systems*. Addison-Wesley, 1986.

[45] W. Young, G. Weckman, and W. Holland. A survey of methodologies for the treatment of missing values within datasets: limitations and benefits. *Theoretical Issues in Ergonomics Science*, 12, January 2011.

[46] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia. Continuous privacy preserving publishing of data streams. In *Proc. of the 12th Int. Conf. on Extending Database Technology*, EDBT, pages 648–659, 2009.

[47] I. Zliobaite. Controlled permutations for testing adaptive learning models. *Knowledge and Information Systems*, In Press, 2014.

[48] I. Zliobaite, A. Bifet, M. Gaber, B. Gabrys, J. Gama, L. Minku, and K. Musial. Next challenges for adaptive learning systems. *SIGKDD Explorations*, 14(1):48–55, 2012.

[49] I. Zliobaite and B. Gabrys. Adaptive preprocessing for streaming data. *IEEE Trans. on Knowledge and Data Engineering*, 26(2):309–321, 2014.

Twitter Analytics: A Big Data Management Perspective

Oshini Goonetilleke[†], Timos Sellis[†], Xiuzhen Zhang[†], Saket Sathe[§]
[†]School of Computer Science and IT, RMIT University, Melbourne, Australia
[§]IBM Melbourne Research Laboratory, Australia
{oshini.goonetilleke, timos.sellis, xiuzhen.zhang}@rmit.edu.au, ssathe@au.ibm.com

ABSTRACT

With the inception of the Twitter microblogging platform in 2006, a myriad of research efforts have emerged studying different aspects of the Twittersphere. Each study exploits its own tools and mechanisms to capture, store, query and analyze Twitter data. Inevitably, platforms have been developed to replace this ad-hoc exploration with a more structured and methodological form of analysis. Another body of literature focuses on developing languages for querying Tweets. This paper addresses issues around the big data nature of Twitter and emphasizes the need for new data management and query language frameworks that address limitations of existing systems. We review existing approaches that were developed to facilitate twitter analytics followed by a discussion on research issues and technical challenges in developing integrated solutions.

1. INTRODUCTION

The massive growth of data generated from social media sources have resulted in a growing interest on efficient and effective means of collecting, analyzing and querying large volumes of social data. The existing platforms exploit several characteristics of big data, including large volumes of data, velocity due to the streaming nature of data, and variety due to the integration of data from the web and other sources. Hence, social network data presents an excellent testbed for research on big data.

In particular, online social networking and microblogging platform Twitter has seen exponential growth in its user base since its inception in 2006 with now over 200 million monthly active users producing 500 million tweets (*Twittersphere*, the postings made to Twitter) daily[1]. A wide research community has been established since then with the hope of understanding interactions on Twitter. For example, studies have been conducted in many domains exploring different perspectives of understanding human behavior.

Prior research focuses on a variety of topics including opinion mining [12, 14, 38], event detection [46, 65, 76], spread of pandemics [26,58,68], celebrity engagement [74] and analysis of political discourse [28, 40, 70]. These types of efforts have enabled researchers to understand interactions on Twitter related to the fields of journalism, education, marketing, disaster relief etc.

[1]http://tnw.to/s0n9u

The systems that perform analysis in the context of these interactions typically involve the following major components: focused crawling, data management and data analytics. Here, data management comprises of information extraction, pre-processing, data modeling and query processing components. Figure 1 shows a block diagram of such a system and depicts interactions between various components. Until now, there has been significant amount of prior research around improving each of the components shown in Figure 1, but to the best of our knowledge, there have been no frameworks that propose a unified approach to Twitter data management that seamlessly integrate all these components. Following these observations, in this paper we extensively survey the techniques that have been proposed for realising each of the components shown in Figure 1, and then motivate the need and challenges of a unified framework for managing Twitter data.

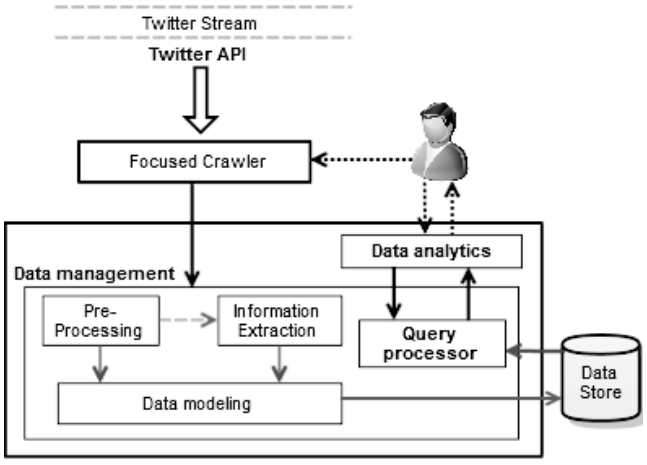


Figure 1: An abstraction of a Twitter data management platform

In our survey of existing literature we observed ways in which researchers have tried to develop general platforms to provide a repeatable foundation for Twitter data analytics. We review the tweet analytics space primarily focusing on the following key elements:

- **Data Collection.** Researchers have several options for collecting a suitable data set from Twitter. In Section 2 we briefly describe mechanisms and tools that focus primarily on facilitating the initial data acquisition phase. These tools systematically capture the data using any of

the Twitter's publicly accessible APIs.

- **Data management frameworks.** In addition to providing a module for crawling tweets, these frameworks provide support for pre-processing, information extraction and/or visualization capabilities. Prepossessing deals with preparing tweets for data analysis. Information extraction aims to derive more insight from the tweets, which is not directly reported by the Twitter API, e.g. a sentiment for a given tweet. Several frameworks provide functionality to present the results in many output forms of visualizations while others are built exclusively to search over a large tweet collection. In Section 3 we review existing data management frameworks.
- **Languages for querying tweets.** A growing body of literature proposes declarative query languages as a mechanism of extracting structured information from tweets. Languages present end users with a set of primitives beneficial in exploring the Twittersphere along different dimensions. In Section 4 we investigate declarative languages and similar systems developed for querying a variety of tweet properties.

We have identified the essential ingredients for a unified Twitter data management solution, with the intention that an analyst will easily be able to extend its capabilities for specific types of research. Such a solution will allow the data analyst to focus on the use cases of the analytics task by conveniently using the functionality provided by an integrated framework. In Section 5 we present our position and emphasize the need for integrated solutions that address limitations of existing systems. Section 6 outlines research issues and challenges associated with the development of integrated platforms. Finally we conclude in Section 7.

2. OPTIONS FOR DATA COLLECTION

Researchers have several options when choosing an API for data collection, i.e. the Search, Streaming and the REST API. Each API has varying capabilities with respect to the type and the amount of information that can be retrieved. The Search API is dedicated for running searches against an index of recent tweets. It takes keywords as queries with the possibility of multiple queries combined as a comma separated list. A request to the search API returns a collection of relevant tweets matching a user query.

The Streaming API provides a stream to continuously capture the public tweets where parameters are provided to filter the results of the stream by hashtags, keywords, twitter user ids, usernames or geographic regions. The REST API can be used to retrieve a fraction of the most recent tweets published by a Twitter user. All three APIs limit the number of requests within a time window and rate-limits are posed at the user and the application level. Response obtained from Twitter API is generally in the JSON format. Third party libraries[2] are available in many programming languages for accessing the Twitter API. These libraries provide wrappers and provide methods for authentication and other functions to conveniently access the API.

Publicly available APIs do not guarantee complete coverage of the data for a given query as the feeds are not designed for enterprise access. For example, the streaming API only provides a random sample of 1% (known as the *Spritzer* stream) of the public Twitter stream in real-time. Applications where this rate limitation is too restrictive rely on third party resellers like GNIP, DataSift or Topsy [3], who provides access to the entire collection of the tweets known as the *Twitter FireHose*. At a cost, resellers can provide unlimited access to archives of historical data, real-time streaming data or both. It is mostly the corporate businesses who opt for such alternatives to gain insights into their consumer and competitor patterns.

In order to obtain a dataset sufficient for an analysis task, it is necessary to efficiently query the respective API methods, within the bounds of imposed rate limits. The requests to the API may have to run continuously spanning across several days or weeks. Creating the users social graph for a community of interest requires additional modules that crawl user accounts iteratively. Large crawls with more complete coverage was made possible with the use of whitelisted accounts [21, 45] and using the computation power of cloud computing [55]. Due to Twitters current policy, whitelisted accounts are discontinued and are no longer an option as means of large data collection. Distributed systems have been developed [16, 45] to make continuously running, large scale crawls feasible. There are other solutions that provide extended features to process the incoming Twitter data as discussed next.

3. DATA MANAGEMENT FRAMEWORKS

3.1 Focused Crawlers

The focus in studies like TwitterEcho [16] and Byun *et al.* [19] is data collection, where the primary contributions are driven by crawling strategies for effective retrieval and better coverage. TwitterEcho describes an open source distributed crawler for Twitter. Data can be collected from a focused community of interest and it adapts a centralized distributed architecture in which multiple thin clients are deployed to create a scalable system. TwitterEcho devises a user expansion strategy by which the user's follower lists are crawled iteratively using the REST API. The system also includes modules responsible for controlling the expansion strategy. The user selection feature identifies user accounts to be monitored by the system with modules for user profile analysis and language identification. The user selection feature is customized to crawl the focused community of Portuguese tweets but can be adapted to target other communities. Byun *et al.* [19] in their work propose a rule-based data collection tool for Twitter with the focus of analysing sentiment of Twitter messages. It is a java-based open source tool developed using the Drools[4] rule engine. They stress the importance of an automated data collector that also filters out unnecessary data such as spam messages.

3.2 Pre-processing and Information Extraction

Apart from data collection, several frameworks implement methods to perform extensive pre-processing and information extraction of the tweets. Pre-processing tasks of TrendMiner [61] take into account the challenges posed by the noisy genre of tweets. Tokenization, stemming and POS

[2] `https://dev.twitter.com/docs/twitter-libraries`

[3] `http://gnip.com/`, `http://datasift.com/`, `http://about.topsy.com/`

[4] `http://drools.jboss.org/`

tagging are some of the text processing tasks that better prepare tweets for the analysis task. The platform provides separate built-in modules to extract information such as location, language, sentiment and named entities that are deemed very useful in data analytics. The creation of a pipeline of these tools allows the data analyst to extend and reuse each component with relative ease.

TwitIE [17] is another open-source information extraction NLP pipeline customized for microblog text. For the purpose of information extraction (IE), the general purpose IE pipeline ANNIE is used and it consists of components such as sentence splitter, POS tagger and gazetteer lists (for location prediction). Each step of the pipeline addresses drawbacks in traditional NLP systems by addressing the inherent challenges in microblog text. As a result, individual components of ANNIE are customized. Language identification, tokenisation, normalization, POS tagging and named entity recognition is performed with each module reporting accuracy on tweets.

Baldwin [11] presents a system designed for event detection on Twitter with functionality for pre-processing. JSON results returned by the Streaming API are parsed and piped through language filtering and lexical normalisation components. Messages that do not have location information are geo-located, using probabilistic models since it's a critical issue in identifying where an event occurs. Information extraction modules require knowledge from external sources and are generally more expensive tasks than language processing. Platforms that support real-time analysis [11, 76] require processing tasks to be conducted on-the-fly where the speed of the underlying algorithms is a crucial consideration.

3.3 Generic Platforms

There are several proposals in which researchers have tried to develop generic platforms to provide a repeatable foundation for Twitter data analytics. Twitter Zombie [15] is a platform to unify the data gathering and analysis methods by presenting a candidate architecture and methodological approach for examining specific parts of the Twittersphere. It outlines architecture for standard capture, transformation and analysis of Twitter interactions using the Twitter's Search API. This tool is designed to gather data from Twitter by executing a series of independent search jobs on a continual basis and the collected tweets and their metadata is kept in a relational DBMS. One of the interesting features of TwitterZombie is its ability to capture hierarchical relationships in the data returned by Twitter. A network translator module performs post-processing on the tweets and stores hashtags, mentions and retweets, separately from the tweet text. Raw tweets are transformed into a representation of interactions to create networks of retweets, mentions and users mentioning hashtags. This feature captured by TwitterZombie, which other studies pay little attention to, is helpful in answering different types of research questions with relative ease. Social graphs are created in the form of a retweet or mention network and they do not crawl for the user graph with traditional following relationships. It also draws discussion on how multi-byte tweets in languages like Arabic or Chinese can be stored by performing transliteration.

More recently, TwitHoard [69] suggests a framework of supporting processors for data analytics on Twitter with emphasis on selection of a proper data set for the definition of a campaign. The platform consists of three layers; campaign crawling layer, integrated modeling layer and the data analysis layer. In the campaign crawling layer, a configuration module follows an iterative approach to ensure the campaign converges to a proper set of filters (keywords). Collected tweets, metadata and the community data (relationships among Twitter users) are stored in a graph database. This study should be highlighted for its distinction to allow for a flexible querying mechanism on top of a data model built on raw data. The model is generated in the integrated modeling layer and comprises a representation of associations between terms (e.g. hashtags) used in tweets and their evolution in time. Their approach is interesting as it captures the often overlooked temporal dimension. In the third data analysis layer, a query language is used to design a target view of the campaign data that corresponds to a set of tweets that contain for example, the answer to an opinion mining question.

While including components for capture and storage of tweets, additional tools have been developed to search through the collected tweets. The architecture of CoalMine [72] presents a social network data mining system demonstrated on Twitter, designed to process large amounts of streaming social data. The ad-hoc query tool provides an end user with the ability to access one or more data files through a Google-like search interface. Appropriate support is provided for a set of boolean and logical operators for ease of querying on top of a standard Apache Lucene index. The data collection and storage component is responsible for establishing connections to the REST API and to store the JSON objects returned in compressed formats.

In building support platforms it is necessary to make provisions for practical considerations such as processing big data. TrendMiner [61] facilitates real-time analysis of tweets and takes into consideration scalability and efficiency of processing large volumes of data. TrendMiner makes an effort to unify some of the existing text processing tools for Online Social Networking (OSN) data, with emphasis on adapting to real-life scenarios that include processing batches of millions of data. They envision the system to be developed for both batch-mode and online processing. TwitIE [17] as discussed in the previous section is another open-source information extraction NLP pipeline customized for microblog text.

3.4 Application-specific Platforms

Apart from the above mentioned general purpose platforms, there are many frameworks targeted at conducting specific types of analysis with Twitter data. Emergency Situation Awareness (ESA) [76] is a platform developed to detect, assess, summarise and report messages of interest published on Twitter for crisis coordination tasks. The objective of their work is to convert large streams of social media data into useful situation awareness information in real-time. The ESA platform consists of modules to detect incidents, condense and summarise messages, classify messages of high value, identify and track issues and finally to conduct forensic analysis of historical events. The modules are enriched by a suite of visualisation interfaces. Baldwin *et al.* [11] propose another support platform focused on detecting events on Twitter. The Twitter stream is queried with a set of keywords specified by the user with the objective of filtering the

stream on a topic of interest. The results are piped through text processing components and the geo-located tweets are visualised on a map for better interaction. Clearly, platforms of this nature that deal with incident exploration need to make provisions for real-time analysis of the incoming Twitter stream and produce suitable visualizations of detected incidents.

3.5 Data Model and Storage Mechanisms

Data models are not discussed in detail in most studies, as a simple data model is sufficient to conduct basic form of analysis. When standard tweets are collected, flat files [11, 72] is the preferred choice. Several studies that capture the social relationships [15, 19] of the Twittersphere, employs the relational data model but do not necessarily store the relationships in a graph database. As a consequence, many analyses that can be performed conveniently on a graph are not captured by these platforms. Only TwitHoard [69] in their paper models co-occurrence of terms as a graph with temporally evolving properties. Twitter Zombie [15] and TwitHoard [69] should be highlighted for capturing interactions including the retweets and term associations apart from the traditional follower/friend social relationships. TrendMiner [61] draws explicit discussion on making provisions for processing millions of data and takes advantage of Apache Hadoop MapReduce framework to perform distributed processing of the tweets stored as key-value pairs. CoalMine [72] also has Apache Hadoop at the core of their batch processing component responsible for efficient processing of large amount of data.

3.6 Support for Visualization Interfaces

There are many platforms designed with integrated tools predominantly for visualization, to analyse data in spatial, temporal and topical perspectives. One tool is tweetTracker [44], which is designed to aid monitoring of tweets for humanitarian and disaster relief. TweetXplorer [54] also provides useful visualization tools to explore Twitter data. For a particular campaign, visualizations in tweetXplorer help analysts to view the data along different dimensions; most interesting days in the campaign (when), important users and their tweets (who/what) and important locations in the dataset (where). Systems like TwitInfo [48], Twitcident [8] and Torettor [65] also provide a suite of visualisation capabilities to explore tweets in different dimensions relating to specific applications like fighting fire and detecting earthquakes. Web-mashups like Trendsmap [5] and Twitalyzer [6] provide a web interface and enterprise business solutions to gain real-time trend and insights of user groups.

Table 1 illustrates an overview of related approaches and features of different platforms. *Pre-processing* in Table 1 indicates if any form of language processing tasks such as POS tagging or normalization are conducted. *Information extraction* refers to the types of post processing performed to infer additional information, such as sentiment or named entities (NEs). Multiple ticks (✓) correspond to a task that is carried out extensively. In addition to collecting tweets, some studies also capture the user's social graph while others propose the need to regard interactions of hashtags, retweets, mentions as separate properties. Backend data models supported by the platform shape the types of analysis that can be conveniently done on each framework. From the summary in Table 1, we can observe that each study on data management frameworks concentrate on a set of challenges more than others.

We aim to recognize the key ingredients of an integrated framework that takes into account shortcomings of existing systems.

4. LANGUAGES FOR QUERYING TWEETS

The goal of proposing declarative languages and systems for querying tweets is to put forward a set of primitives or an interface for analysts to conveniently query specific interactions on Twitter exploring the user, time, space and topical dimensions. High level languages for querying tweets extend capabilities of existing languages such as SQL and SPARQL. Queries are either executed on the Twitter stream in real-time or on a stored collection of tweets.

4.1 Generic Languages

TweeQL [49] provides a streaming SQL-like interface to the Twitter API and provides a set of user defined functions (UDFs) to manipulate data. The objective is to introduce a query language to extract structure and useful information that is embedded in unstructured Twitter data. The language exploits both relational and streaming semantics. UDFs allow for operations such as location identification, string processing, sentiment prediction, named entity extraction and event detection. In the spirit of streaming semantics, it provides SQL constructs to perform aggregations over the incoming stream on a user-specified time window. The result of a given query can be stored in a relational fashion for subsequent querying.

Models for representing any social network in RDF have been proposed by Martin and Gutierrez [50] allowing queries in SPARQL. The work explores the feasibility of adoption of this model by demonstrating their idea with an illustrative prototype but does not focus on a single social network like Twitter in particular. TwarQL [53] extracts content from tweets and encodes it in RDF format using shared and well known vocabularies (FOAF, MOAT, SIOC) enabling querying in SPARQL. The extraction facility processes plain tweets and expands its description by adding sentiment annotations, DBPedia entities, hashtag definitions and URLs. The annotation of tweets using different vocabularies enables querying and analysis in different dimensions such as location, users, sentiment and related named entities. The infrastructure of TwarQL enables subscription to a stream that matches a given query and returns streaming annotated data in real-time.

Temporal and topical features are of paramount importance in an evolving microblogging stream like Twitter. In the languages above, time and topic of a tweet (topic can be represented simply by a hashtag) are considered meta-data of the tweet and is not treated any different from other metadata reported. Topics are regarded as part of the tweet content or what drives the data filtering task from the Twitter API. There have been efforts to exploit features that go well beyond a simple filter based on time and topic. Plachouras and Stavrakas [59] stress the need for temporal modelling of terms in Twitter to effectively capture changing trends. A term refers to any word or short phrase of interest in a tweet, including hashtags or output of an entity recognition process. Their proposed query operators can express complex queries for associations between terms over vary-

Table 1: Overview of related approaches in data management frameworks.

	Prepossessing	Examples of extracted information	Social and/or other interactions captured?	Data Store
TwitterEcho [16]	✓	Language	Yes	Not given
Byun *et al.* [19]		Location	Yes	Relational
Twitter Zombie [15]	✓		Yes	Relational
TwitHoard [69]	✓		Yes	Graph DB
CoalMine [72]			No	Files
TrendMiner [61]	✓✓	Location, Sentiment, NEs	No	Key-value pairs
TwitIE [17]	✓✓	Language, Location, NEs	No	Not given
ESA [76]	✓	Location, NEs	No	Not given
Baldwin *et al.* [11]	✓✓	Language, Location	No	Flat files

ing time granularities, to discover context of collected data. Operators also allow retrieving a subset of tweets satisfying these complex conditions on term associations. This enables the end user to select a good set of terms (hashtags) that drive the data collection which has a direct impact on the quality of the results generated from the analysis.

Spatial features are another property of tweets often overlooked in complex analysis. Previously discussed work uses the location attribute as a mechanism to filter tweets in space. To complete our discussion we briefly outline two studies that use geo-spatial properties to perform complex analysis using the location attribute. Doytsher *et al.* [31] introduced a model and query language suited for integrated data connecting a social network of users with a spatial network to identify places visited frequently. Edges named *life-patterns* are used to associate the social and spatial networks. Different time granularities can be expressed for each visited location represented by the life-pattern edge. Even though the implementation employs a partially synthetic dataset, it will be interesting to investigate how the socio-spatial networks and the life-pattern edges that are used to associate the spatial and social networks can be represented in a real social network dataset with location information, such as Twitter. GeoScope [18] finds information trends by detecting significant correlations among trending location-topic pairs in a sliding window. This gives rise to the importance of capturing the notion of spatial information trends in social networks in analysis tasks. Real-time detection of crisis events from a location in space, exhibits the possible value of Geoscope. In one of the experiments Twitter is used as a case study to demonstrate its usefulness, where a hashtag is chosen to represent the topic and city from which the tweet originates chosen to capture the location.

4.2 Data Model for the Languages

Relational, RDF and Graphs are the most common choices of data representation. There is a close affiliation in these data models observing that, for instance, a graph can easily correspond to a set of RDF triples or vice versa. In fact, some studies like Plachouras and Stavrakas [59] have put forward their data model as a labeled multi digraph and have chosen a relational database for its implementation. None of these query systems models Twitter social network with following or retweet relationships among users. Doytsher *et al.* [31] implement their algebraic query operators with the use of both graph and a relational database as the underlying data storage. They experimentally compare relational and graph database to demonstrate the feasibility of the model. languages that operate on the twitter stream like TweeQL and TwarQL generates the output in real-time; TweeQL [49] allows the resulting tweets to be collected in batches then stores them in a relational database, while TwarQL [53] at the end of the information extraction phase, annotated tweets are encoded in RDF.

4.3 Query Languages for Social Networks

To the best of our knowledge, there is no existing work focusing on high level languages operating on the Twitter's social graph. However it is important to note proposals for declarative query languages tailored for querying social networks in general [10, 30, 32, 50, 51, 64]. One of the queries supported are path queries satisfying a set of conditions on the path, and the languages in general take advantage of inherent properties of social networks. Semantics of the languages are based on Datalog [51], SQL [32, 64] or RDF/SPARQL [50]. Implementations are conducted on bibliographical networks [32], Facebook and evolving social content sites like Yahoo! Travel [10] and are not tested on Twitter networks taking Twitter specific affordances into consideration.

4.4 Information Retrieval - Tweet Search

Another class of systems presents textual queries to efficiently search over a corpus of tweets. The challenges in this area are similar to that of information retrieval in addition with having to deal with peculiarities of tweets. The short length of tweets in particular creates added complexity to text-based search tasks as it is difficult to identify relevant tweets matching a user query [13,35]. Expanding tweet content is suggested as a way to enhance the meaning. The goal of such systems is to express a user's information need in the form of a simple text query, much like in search engines, and return a tweet list in real-time with effective strategies for ranking and relevance measurements [33, 34, 71]. Indexing mechanisms are discussed in [23] as they directly impact efficient retrieval of tweets. The TREC micro-blogging track[5] is dedicated to calling participants to conduct real-time adhoc search tasks over a given tweet collection. Publications of TREC [56], documents the findings of all systems in the task of ranking the most relevant tweets matching a predefined set of user queries.

Table 2 illustrates an overview of related approaches in systems for querying tweets. Data models and dimensions in-

[5] `http://trec.nist.gov/`

Table 2: Overview of approaches in systems for querying tweets.

	Data Model			Explored dimensions				
	Relational	RDF	Graph	Text	Time	Space	Social Network	Real-Time
TweeQL [49]	✓			✓	✓	✓		Yes
TwarQL [53]		✓		✓	✓	✓		Yes
Plachouras *et al.* [60]	✓			✓	✓✓			No
Doytsher *et al.* [31]*	✓		✓		✓	✓✓	✓	No
GeoScope *et al.* [18]*				✓	✓	✓ ✓		Yes
Languages on social networks*	✓	✓	✓			✓	✓✓	No
Tweet search systems	✓			✓✓	✓		✓	Yes

vestigated in each system are depicted in the Table 2. Systems that have made provision for the real-time streaming nature of the tweets are indicated in the *Real-time* column. Multiple ticks (✓) correspond to a dimension explored in detail. Note that the systems marked with an asterisk (*) are not implemented specifically targeting tweets, though their application is meaningful and can be extended to the Twittersphere. We observe potential for developing languages for querying tweets that include querying by dimensions that are not captured by existing systems, especially the social graph.

5. THE NEED FOR INTEGRATED SOLUTIONS

There is a need to assimilate individual efforts with the goal of providing a unified framework that can be used by researchers and practitioners across many disciplines. Integrated solutions should ideally handle the entire workflow of the data analysis life cycle from collecting the tweets to presenting the results to the user. The literature we have reviewed in previous sections outlines efforts that support different parts of the workflow. In this section, we present our position with the aim of outlining significant components of an integrated solution addressing the limitations of existing systems.

According to a review of literature conducted on the microblogging platform [25], the majority of published work on Twitter concentrates on the user domain and the message domain. The user domain explores properties of Twitter users in the microblogging environment while the message domain deals with properties exhibited by the tweets themselves. In comparison to the extent of work done on the microblogging platform, only a few investigates the development of data management frameworks and query languages that describe and facilitate processing of online social networking data. In consequence, there is opportunity for improvement in this area for future research addressing the challenges in data management. We elicit the following high-level components and envisage a platform for Twitter that encompasses such capabilities:

Focused crawler: Responsible for retrieval and collection of Twitter data by crawling the publicly accessible Twitter APIs. A focused crawler should allow the user to define a campaign with suitable filters, monitor output and iteratively crawl Twitter for large volumes of data until its coverage of relevant tweets is satisfactory.

Pre-processor: As highlighted in Section 3.2, this stage usually comprises of modules for pre-processing and information extraction considering the inherent peculiarities of tweets and not all frameworks we discussed provided this functionality. Pre- processing components for example, normalization and tokenization should be implemented, with the option for the end users to customize the modules to suit their requirements. Information extraction modules such as location prediction, sentiment classification and named entity recognition are useful in conducting analysis on tweets and attempt to derive more information from plain tweet text and their metadata. Ideally, a user should be able to integrate any combination of the components into their own applications.

Data model: Much of the literature presented (Section 3.5 and 4.2) does not emphasize or draw explicit discussions on the data model in use. The logical data model greatly influences the types of analysis that can be done with relative ease on collected data. A physical representation of the model involving suitable indexing and storage mechanisms of large volumes of data is an important consideration for efficient retrieval. We notice that current research pays little attention to queries on Twitter interactions, the social graph in particular. A graph view of the Twittersphere is consistently overlooked and we recognize great potential in this area. The graph construction on Twitter is not limited to considering the users as nodes and links as following relationships; embracing useful characteristics such as retweet, mention and hashtag (co-occurrence) networks in the data model will create opportunity to conduct complex analysis on these structural properties of tweets. We envision a new data model that proactively captures structural relationships taking into consideration efficient retrieval of relevant data to perform the queries.

Query language: Languages described in Section 4, defines both simple operators to be applied on tweets and advanced operators that extract complex patterns, which can be manipulated in different types of applications. Some languages provide support for continuous queries on the stream or queries on a stored collection, while others offer flexibility for both. The advent of a graph view makes crucial contributions in analyzing the Twittersphere allowing us to query twitter data in novel and varying forms. It will be interesting to investigate how typical functionality [73] provided by graph query languages can be adapted to Twitter networks. In developing query languages, one could investigate the distinction between real-time and batch mode processing. **Visualizing** the data retrieved as a result of a query in a suitable manner is also an important concern. A set of pre-defined output formats will be useful in order to provide an informative visualization over a map for a query that re-

turns an array of locations. Another interesting avenue to explore is the introduction of a **ranking mechanism** on the query result. Ranking criteria may involve relevance, timeliness or network attributes like the reputation of users in the case of a social graph. Ranking functions are a standard requirement in the field of information retrieval [23,39] and studies like SociQL [30] report the use of visibility and reputations metrics to rank results generated from a social graph. A query language with a graph view of the Twittersphere along with capabilities for visualizations and ranking will certainly benefit the upcoming data analysis efforts of Twitter.

Here, we focused on the key ingredients required for a fully developed solution and discussed improvements we can make on existing literature. In the next section, we identify challenges and research issues involved.

6. CHALLENGES AND RESEARCH ISSUES

To complete our discussion, in this section we summarize key research issues in data management and present technical challenges that need to be addressed in the context of building a data analytics platform for Twitter.

6.1 Data Collection Challenges

Once a suitable Twitter API has been identified, we can define a *campaign* with a set of parameters. The focused crawler can be programmed to retrieve all tweets matching the query of the campaign. If a social graph is necessary, separate modules would be responsible to create this network iteratively. Exhaustively crawling all the relationships between Twitter users is prohibitive given the restrictions set by the Twitter API. Hence it is required for the focused crawler to prioritize the relationships to crawl based on the impact and importance of specific Twitter accounts. In the case that the platform handles multiple campaigns in parallel, there is a need to optimize the access to the API. Typically, the implementation of a crawler should aim to minimize the number of API requests, considering the restrictions, while fetching data for many campaigns in parallel. Hence building an effective crawling strategy is a challenging task, in order to optimize the use of API requests available.

Appropriate coverage of the campaign is another significant concern and denotes whether all the relevant information has been collected. When specifying the parameters to define the campaign, a user needs a very good knowledge on the relevant keywords. Depending on the specified keywords, a collection may miss relevant tweets in addition to the tweets removed due to restrictions by APIs. Plachouras and Stavrakas work [60] is an initial step in this direction as it investigates this notion of coverage and proposes mechanisms to automatically adapt the campaign to evolving hashtags.

6.2 Pre-processing Challenges

Many problems associated with summarization, topic detection and part-of-speech (POS) tagging, in the case of well-formed documents, e.g. news articles, have been extensively studied in the literature. Traditional named entity recognizers (NERs) heavily depend on local linguistic features [62] of well-formed documents like capitalization and POS tagging of previous words. None of the characteristics hold for tweets with short utterances of tweets limited to 140 characters, which make use of informal language, undoubtedly making a simple task of POS tagging more challenging. Besides the length limit, heavy and inconsistent usage of abbreviations, capitalizations and uncommon grammar constructions pose additional challenges to text processing. With the volume of tweets being orders of magnitude more than news articles, most of the conventional methods cannot be directly applied to the noisy genre of Twitter data. Any effort that uses Twitter data needs to make use of appropriate twitter-specific strategies to pre-process text addressing the challenges associated with intrinsic properties of tweets. Similarly, information extraction from tweets is not straightforward as it is difficult to derive context and topics from a tweet that is a scattered part of a conversation. There is separate literature on identifying entities (references to organizations, places, products, persons) [47,63], languages [20,36], sentiment [57] present in the tweet text for a richer source of information. Location is another vital property representing spatial features either of the tweet or of the user. The location of each tweet may be optionally recorded if using a GPS-enabled device. A user can also specify his or her location as a part of the user profile and is often reported in varying granularities. The drawback is that only a small portion of about 1% of the tweets are geo-located [24]. Since analysis almost always requires the location property, when absent, studies conduct their own mechanisms to infer location of the user, a tweet, or both. There are two major approaches for location prediction: content analysis with probabilistic language models [24, 27, 37] or inference from social and other relations [22, 29, 67].

6.3 Data Management Challenges

In Section 3.5 and Section 4.2 we outlined several alternative approaches in literature for a data model to characterise the Twittersphere. The relational and RDF models are frequently chosen while graph-based models are acknowledged, however not realized concretely at the implementation phase. Ways in which we can apply graph data management in Twitter are extremely diverse and interesting; different types of networks can be constructed apart from the traditional social graph as outlined in Section 5. With the advent of a graph view to model the Twittersphere, gives rise to a range of queries that can be performed on the structure of the tweets essentially capturing a wider range of use case scenarios used in typical data analytics tasks.

As discussed in Section 4.3, there are already languages similar to SQL adapted to social networks. Many of the techniques in literature are for the case of generic social networks under a number of specific assumptions. For example the social networks satisfy properties such as the power law distribution, sparsity and small diameters [52]. We envision queries that take another step further and executes on Twitter graphs. Simple query languages FQL [1] and YQL [7] provide features to explore properties of Facebook and Yahoo APIs but are limited to querying only part(usually a single user's connections) of the large social graph. As pointed out in the report on the Databases and Web 2.0 Panel at VLDB 2007 [9], understanding and analyzing trust, authority, authenticity, and other quality measures in social networks pose major research challenges. While there are investigations on these quality measures in Twitter, it's about time that we enable declarative querying of networks using such measurements.

One of the predominant challenges is the management of large graphs that inevitably results from modeling users, tweets and their properties as graphs. With the large volume of data involved in any practical task, a data model should be information rich, yet a concise representation that enables expression of useful queries. Queries on graphs should be optimized for large networks and should ideally run independent of the size of the graph. There arealready approaches that investigate efficient algorithms on very large graphs [41–43, 66]. Efficient encoding and indexing mechanisms should be in place taking into account variations of indexing systems already proposed for tweets [23] and indexing of graphs [75] in general. We need to consider maintaining indexes for tweets, keywords, users, hashtags for efficient access of data in advance queries. In certain situations it may be impractical to store the entire raw tweet and the complete user graph and it may be desirable to either compress or drop portions of the data. It is important to investigate which properties of tweets and the graph should be compressed.

Besides the above challenges, tweets impose general research issues related to big data. Challenges should be addressed in the same spirit as any other big data analytics task. In the face of challenges posed by large volumes of data being collected, the NoSQL paradigm should be considered as an obvious choice of dealing with them. Developed solutions should be extensible for upcoming requirements and should indeed scale well. When collecting data, a user need to consider scalable crawling as there are large volumes of tweets received, processed and indexed per second. With respect to implementation, it is necessary to investigate paradigms that scale well, like MapReduce which is optimized for offline analytics on large data partitioned on hundreds of machines. Depending on the complexity of the queries supported, it might be difficult to express graph algorithms intuitively in MapReduce graph models, consequently databases such as Titan [4], DEX [3], and Neo4j [2] should be compared for graph implementations.

7. CONCLUSION

In this paper we addressed issues around the big data nature of Twitter analytics and the need for new data management and query language frameworks. By conducting a careful and extensive review of the existing literature we observed ways in which researchers have tried to develop general platforms to provide a repeatable foundation for Twitter data analytics. We reviewed the tweet analytics space by exploring mechanisms primarily for data collection, data management and languages for querying and analyzing tweets. We have identified the essential ingredients required for a unified framework that address the limitations of existing systems. The paper outlines research issues and identifies some of the challenges associated with the development of such integrated platforms.

8. REFERENCES

[1] FaceBook Query Language(FQL) overview. `https://developers.facebook.com/docs/technical-guides/fql`.

[2] Neo4j: The world's leading graph database. `http://www.neo4j.org/`.

[3] Sparksee: Scalable high-performance graph database. `http://www.sparsity-technologies.com/`.

[4] Titan: distributed graph database. `http://thinkaurelius.github.io/titan`.

[5] TrendsMap, Realtime local twitter trends. `http://trendsmap.com/`.

[6] Twitalyzer: Serious analytics for social business. `http://twitalyzer.com`.

[7] Yahoo! Query Language guide on YDN. `https://developer.yahoo.com/yql/`.

[8] F. Abel, C. Hauff, and G. Houben. Twitcident: fighting fire with information from social web streams. In *WWW*, pages 305–308, 2012.

[9] S. Amer-Yahia, V. Markl, A. Halevy, A. Doan, G. Alonso, D. Kossmann, and G. Weikum. Databases and Web 2.0 panel at VLDB 2007. In *SIGMOD Record*, volume 37, pages 49–52, Mar. 2008.

[10] S. AmerYahia;, L. V. Lakshmanan;, and Cong Yu. SocialScope : Enabling information discovery on social content sites. In *CIDR*, 2009.

[11] T. Baldwin, P. Cook, and B. Han. A support platform for event detection using social intelligence. In *Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 69–72, 2012.

[12] L. Barbosa and J. Feng. Robust sentiment detection on Twitter from biased and noisy data. pages 36–44, Aug. 2010.

[13] M. S. Bernstein, B. Suh, L. Hong, J. Chen, S. Kairam, and E. H. Chi. Eddi: interactive topic-based browsing of social status streams. In *23nd annual ACM symposium on User interface software and technology - UIST*, pages 303–312, Oct. 2010.

[14] A. Bifet and E. Frank. Sentiment knowledge discovery in twitter streaming data. *Discovery Science. Springer Berlin Heidelberg*, pages 1–15, Oct. 2010.

[15] A. Black, C. Mascaro, M. Gallagher, and S. P. Goggins. Twitter Zombie: Architecture for capturing, socially transforming and analyzing the Twittersphere. In *International conference on Supporting group work*, pages 229–238, 2012.

[16] M. Boanjak and E. Oliveira. TwitterEcho - A distributed focused crawler to support open research with twitter data. In *International conference companion on World Wide Web*, pages 1233–1239, 2012.

[17] K. Bontcheva and L. Derczynski. TwitIE: an open-source information extraction pipeline for microblog text. In *International Conference on Recent Advances in Natural Language Processing*, 2013.

[18] C. Budak, T. Georgiou, and D. E. Abbadi. GeoScope: Online detection of geo-correlated information trends in social networks. *PVLDB*, 7(4):229–240, 2013.

[19] C. Byun, H. Lee, Y. Kim, and K. K. Kim. Twitter data collecting tool with rule-based filtering and analysis module. *International Journal of Web Information Systems*, 9(3):184–203, 2013.

[20] S. Carter, W. Weerkamp, and M. Tsagkias. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, 47(1):195–215, June 2012.

[21] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *ICWSM*, pages 10–17, 2010.

[22] S. Chandra, L. Khan, and F. B. Muhaya. Estimating twitter user location using social interactions–a content based approach. In *IEEE Conference on Privacy, Security, Risk and Trust*, pages 838–843, Oct. 2011.

[23] C. Chen, F. Li, C. Ooi, and S. Wu. TI : An efficient indexing mechanism for real-time search. In *SIGMOD*, pages 649–660, 2011.

[24] Z. Cheng, J. Caverlee, K. Lee, and C. Science. A content-driven framework for geo-locating microblog users. *ACM Transactions on Intelligent Systems and Technology*, 2012.

[25] M. Cheong and S. Ray. A literature review of recent microblogging developments. Technical report, Clayton School of Information Technology, Monash University, 2011.

[26] Chew, Cynthia, and G. Eysenbach. Pandemics in the age of twitter: content analysis of tweets during the 2009 H1N1 outbreak. *PloS one*, 5(11), 2010.

[27] B. O. Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287, 2010.

[28] Conover, Michael, J. Ratkiewicz, M. Francisco, B. Gonçalves, F. Menczer, and A. Flammini. Political polarization on Twitter. In *ICWSM*, 2011.

[29] J. David. Thats what friends are for inferring location in online social media platforms based on social relationships. In *ICWSM*, 2013.

[30] Diego Serrano, Eleni Stroulia, Denilson Barbosa and V. Guana. SociQL: A query language for the social Web. In E. Kranakis, editor, *Advances in Network Analysis and its Applications*, chapter 17, pages 381–406. 2013.

[31] Y. Doytsher and B. Galon. Querying geo-social data by bridging spatial networks and social networks. In *2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, pages 39–46, 2010.

[32] A. Dries, S. Nijssen, and L. De Raedt. A query language for analyzing networks. In *CIKM*, pages 485–494, 2009.

[33] M. Efron. Hashtag retrieval in a microblogging environment. pages 787–788, 2010.

[34] S. Frénot and S. Grumbach. An in-browser microblog ranking engine. In *International conference on Advances in Conceptual Modeling*, volume 7518, pages 78–88, 2012.

[35] G. Golovchinsky and M. Efron. Making sense of Twitter search. In *CHI*, 2010.

[36] M. Graham, S. A. . Hale, and D. Gaffney. Where in the world are you ? Geolocation and language identification in Twitter. In *ICWSM*, pages 518–521, 2012.

[37] B. Hecht, L. Hong, B. Suh, and E. Chi. Tweets from Justin Bieber's heart: the dynamics of the location field in user profiles. In *Conference on Human Factors in Computing Systems*, pages 237–246, 2011.

[38] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Twitter power: Tweets as electronic word of mouth. *Journal of the American Society for Information Science and Technology*, 60(11):2169–2188, Nov. 2009.

[39] J. Jiang, L. Hidayah, T. Elsayed, and H. Ramadan. BEST of KAUST at TREC-2011 : Building effective search in Twitter. *TREC*, 2011.

[40] P. Jürgens, A. Jungherr, and H. Schoen. Small worlds with a difference: new gatekeepers and the filtering of political information on Twitter. In *International Web Science Conference-WebSci*, pages 1–5, June 2011.

[41] U. Kang, D. H. Chau, and C. Faloutsos. Managing and mining large graphs : Systems and implementations. In *SIGMOD*, volume 1, pages 589–592, 2012.

[42] U. Kang and C. Faloutsos. Big graph mining : Algorithms and discoveries. *SIGKDD Explorations*, 14(2):29–36, 2013.

[43] U. Kang, H. Tong, J. Sun, C.-Y. Lin, and C. Faloutsos. Gbase: An efficient analysis platform for large graphs. *VLDB Journal*, 21(5):637–650, June 2012.

[44] S. Kumar, G. Barbier, M. Abbasi, and H. Liu. TweetTracker: An analysis tool for humanitarian and disaster relief. In *ICWSM*, pages 661–662, 2011.

[45] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.

[46] C.-H. Lee, H.-C. Yang, T.-F. Chien, and W.-S. Wen. A novel approach for event detection by mining spatio-temporal information on microblogs. In *International Conference on Advances in Social Networks Analysis and Mining*, pages 254–259, July 2011.

[47] C. Li, J. Weng, Q. He, Y. Yao, and A. Datta. TwiNER: named entity recognition in targeted twitter stream. In *SIGIR*, pages 721–730, 2012.

[48] A. Marcus, M. Bernstein, and O. Badar. Tweets as data: demonstration of TweeQL and Twitinfo. In *SIGMOD*, pages 1259–1261, 2011.

[49] A. Marcus, M. Bernstein, and O. Badar. Processing and visualizing the data in tweets. *SIGMOD Record*, 40(4), 2012.

[50] M. S. Martín and C. Gutierrez. Representing, querying and transforming social networks with RDF/SPARQL. In *European Semantic Web Conference*, pages 293–307, 2009.

[51] P. T. W. Mauro San Martín, Claudio Gutierrez. SNQL : A social network query and transformation language. In *5th Alberto Mendelzon International Workshop on Foundations of Data Management*, 2011.

[52] M. Mcglohon and C. Faloutsos. Statistical properties of social networks. In C. C. Aggarwal, editor, *Social Network Data Analytics*, chapter 2, pages 17–42. 2011.

[53] P. Mendes, A. Passant, and P. Kapanipathi. Twarql: tapping into the wisdom of the crowd. In *Proceedings of the 6th International Conference on Semantic Systems*, pages 3–5, 2010.

[54] F. Morstatter, S. Kumar, H. Liu, and R. Maciejewski. Understanding Twitter data with TweetXplorer. In *SIGKDD*, pages 1482–1485, 2013.

[55] P. Noordhuis, M. Heijkoop, and A. Lazovik. Mining Twitter in the cloud: A case study. In *IEEE 3rd International Conference on Cloud Computing*, pages 107–114, July 2010.

[56] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the TREC-2011 Microblog Track. In *20th Text REtrieval Conference (TREC)*, 2011.

[57] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *International Conference on Language Resources and Evaluation*, pages 1320–1326, 2010.

[58] Paul, M. J, and M. Dredze. In *ICWSM*, pages 265–272.

[59] V. Plachouras and Y. Stavrakas. Querying term associations and their temporal evolution in social data. In *International VLDB Workshop on Online Social Systems*, 2012.

[60] V. Plachouras, Y. Stavrakas, and A. Andreou. Assessing the coverage of data collection campaigns on Twitter: A case study. In *On the Move to Meaningful Internet Systems: OTM 2013 Workshops*, pages 598–607. 2013.

[61] D. Preotiuc-Pietro, S. Samangooei, and T. Cohn. Trendminer : An architecture for real time analysis of social media text. In *Workshop on Real-Time Analysis and Mining of Social Streams*, pages 4–7, 2012.

[62] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Conference on Computational Natural Language Learning (CoNLL)*, number June, pages 147–155, 2009.

[63] A. Ritter, S. Clark, and O. Etzioni. Named entity recognition in tweets : an experimental study. In *Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, 2011.

[64] R. Ronen and O. Shmueli. SoQL: A language for querying and creating data in social networks. In *ICDE*, pages 1595–1602, Mar. 2009.

[65] T. Sakaki. Earthquake shakes twitter users : Real-time event detection by social sensors. In *WWW*, pages 851–860, 2010.

[66] S. Salihoglu and J. Widom. GPS : A graph processing system. In *International Conference on Scientific and Statistical Database Management*, pages 1–31, 2013.

[67] A. Schulz, A. Hadjakos, and H. Paulheim. A multi-indicator approach for geolocalization of tweets. In *ICWSM*, pages 573–582, 2013.

[68] A. Signorini, A. M. Segre, and P. M. Polgreen. The use of Twitter to track levels of disease activity and public concern in the U.S. during the influenza A H1N1 pandemic. *PloS one*, 6(5), Jan. 2011.

[69] Y. Stavrakas and V. Plachouras. A platform for supporting data analytics on twitter challenges and objectives. *Intl. Workshop on Knowledge Extraction & Consolidation from Social Media*, (Ict 270239), 2013.

[70] Tumasjan, Andranik, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In *ICWSM*, pages 178–185, 2010.

[71] J. Weng, E.-p. Lim, and J. Jiang. TwitterRank : Finding topic-sensitive influential twitterers. In *WSDM*, pages 261–270, 2010.

[72] J. S. White, J. N. Matthews, and J. L. Stacy. Coalmine: an experience in building a system for social media analytics. In I. V. Ternovskiy and P. Chin, editors, *Proceedings of SPIE*, volume 8408, 2012.

[73] P. T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, Apr. 2012.

[74] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts. Who says what to whom on twitter. In *WWW*, pages 705–714, Mar. 2011.

[75] X. Yan, P. S. Yu, and J. Han. Graph indexing : A frequent structure-based approach. In *SIGMOD*, pages 335–346, 2004.

[76] J. Yin, S. Karimi, B. Robinson, and M. Cameron. ESA: emergency situation awareness via microbloggers. In *CIKM*, pages 2701–2703, 2012.

What is Tumblr: A Statistical Overview and Comparison

Yi Chang[†], Lei Tang[§], Yoshiyuki Inagaki[†], Yan Liu[‡]
[†]Yahoo Labs, Sunnyvale, CA 94089
[§]@WalmartLabs, San Bruno, CA 94066
[‡]University of Southern California, Los Angeles, CA 90089
yichang@yahoo-inc.com, leitang@acm.org, inagakiy@yahoo-inc.com, yanliu.cs@usc.edu

ABSTRACT

Tumblr, as one of the most popular microblogging platforms, has gained momentum recently. It is reported to have 166.4 millions of users and 73.4 billions of posts by January 2014. While many articles about Tumblr have been published in major press, there is not much scholar work so far. In this paper, we provide some pioneer analysis on Tumblr from a variety of aspects. We study the social network structure among Tumblr users, analyze its user generated content, and describe reblogging patterns to analyze its user behavior. We aim to provide a comprehensive statistical overview of Tumblr and compare it with other popular social services, including blogosphere, Twitter and Facebook, in answering a couple of key questions: *What is Tumblr? How is Tumblr different from other social media networks?* In short, we find Tumblr has more rich content than other microblogging platforms, and it contains hybrid characteristics of social networking, traditional blogosphere, and social media. This work serves as an early snapshot of Tumblr that later work can leverage.

1. INTRODUCTION

Tumblr, as one of the most prevalent microblogging sites, has become phenomenal in recent years, and it is acquired by Yahoo! in 2013. By mid-January 2014, Tumblr has 166.4 millions of users and 73.4 billions of posts[1]. It is reported to be the most popular social site among young generation, as half of Tumblr's visitor are under 25 years old[2]. Tumblr is ranked as the 16th most popular sites in United States, which is the 2nd most dominant blogging site, the 2nd largest microblogging service, and the 5th most prevalent social site[3]. In contrast to the momentum Tumblr gained in recent press, little academic research has been conducted over this burgeoning social service. Naturally questions arise: *What is Tumblr? What is the difference between Tumblr and other blogging or social media sites?*

Traditional blogging sites, such as Blogspot[4] and Live Journal[5], have high quality content but little social interactions. Nardi *et al.* [17] investigated blogging as a form of personal communication and expression, and showed that the vast majority of blog posts are written by ordinary people with a small audience. On the contrary, popular social networking sites like Facebook[6], have richer social interactions, but lower quality content comparing with blogosphere. Since most social interactions are either unpublished or less meaningful for the majority of public audience, it is natural for Facebook users to form different communities or social circles. Microblogging services, in between of traditional blogging and online social networking services, have intermediate quality content and intermediate social interactions. Twitter[7], which is the largest microblogging site, has the limitation of 140 characters in each post, and the Twitter following relationship is not reciprocal: a Twitter user does not need to follow back if the user is followed by another. As a result, Twitter is considered as a new social media [11], and short messages can be broadcasted to a Twitter user's followers in real time.

Tumblr is also posed as a microblogging platform. Tumblr users can follow another user without following back, which forms a non-reciprocal social network; a Tumblr post can be re-broadcasted by a user to its own followers via *reblogging*. But unlike Twitter, Tumblr has no length limitation for each post, and Tumblr also supports multimedia post, such as images, audios or videos. With these differences in mind, are the social network, user generated content, or user behavior on Tumblr dramatically different from other social media sites?

In this paper, we provide a statistical overview over Tumblr from assorted aspects. We study the social network structure among Tumblr users and compare its network properties with other commonly used ones. Meanwhile, we study content generated in Tumblr and examine the content generation patterns. One step further, we also analyze how a blog post is being reblogged and propagated through a network, both topologically and temporally. Our study shows that Tumblr provides hybrid microblogging services: it contains dual characteristics of both social media and traditional blogging. Meanwhile, surprising patterns surface. We describe these intriguing findings and provide insights, which hopefully can be leveraged by other researchers to understand more about this new form of social media.

2. TUMBLR AT FIRST SIGHT

Tumblr is ranked the second largest microblogging service, right after Twitter, with over 166.4 million users and 73.4 billion posts by January 2014. Tumblr is easy to register, and one can sign up for Tumblr service with a valid email address within 30 seconds. Once sign in Tumblr, a user can follow other users. Different from Facebook, the connections in Tumblr do not require mutual confirmation. Hence the social network in Tumblr is unidirectional.

[1]http://www.tumblr.com/about
[2]http://www.webcitation.org/64UXrbl8H
[3]http://www.alexa.com/topsites/countries/US
[4]http://blogspot.com
[5]http://livejournal.com
[6]http://facebook.com
[7]http://twitter.com

Both Twitter and Tumblr are considered as microblogging platforms. Comparing with Twitter, Tumblr exposes several differences:

- There is no length limitation for each post;
- Tumblr supports multimedia posts, such as images, audios and videos;
- Similar to hashtags in Twitter, bloggers can also tag their blog post, which is commonplace in traditional blogging. But tags in Tumblr are seperate from blog content, while in Twitter the hashtag can appear anywhere within a tweet.
- Tumblr recently (Jan. 2014) allowed users to mention and link to specific users inside posts. This *@user* mechanism needs more time to be adopted by the community;
- Tumblr does not differentiate verified account.

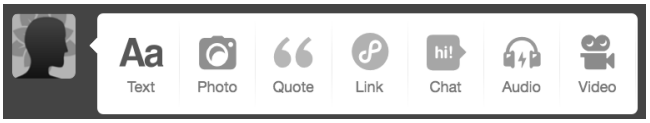


Figure 1: Post Types in Tumblr

Specifically, Tumblr defines 8 types of posts: *photo*, *text*, *quote*, *audio*, *video*, *chat*, *link* and *answer*. As shown in Figure 1, one has the flexibility to start a post in any type except *answer*. *Text*, *photo*, *audio*, *video* and *link* allow one to post, share and comment any multimedia content. *Quote* and *chat*, which are not available in most other social networking platforms, let Tumblr users share quote or chat history from ichat or msn. *Answer* occurs only when one tries to interact with other users: when one user posts a question, in particular, writes a post with text box ending with a question mark, the user can enable the option for others to answer the question, which will be disabled automatically after 7 days. A post can also be reblogged by another user to broadcast to his own followers. The reblogged post will quote the original post by default and allow the reblogger to add additional comments.

Figure 2 demonstrates the distribution of Tumblr post types, based on 586.4 million posts we collected. As seen in the figure, even though all kinds of content are supported, *photo* and *text* dominate the distribution, accounting for more than 92% of the posts. Therefore, we will concentrate on these two types of posts for our content analysis later.

Since Tumblr has a strong presence of photos, it is natural to compare it to other photo or image based social networks like Flickr[8] and Pinterest[9]. Flickr is mainly an image hosting website, and Flicker users can add contact, comment or like others' photos. Yet, different from Tumblr, one cannot reblog another's photo in Flickr. Pinterest is designed for curators, allowing one to share photos or videos of her taste with the public. Pinterest links a pin to the commercial website where the product presented in the pin can be purchased, which accounts for a stronger e-commerce behavior. Therefore, the target audience of Tumblr and Pinterest are quite different: the majority of users in Tumblr are under age 25, while Pinterest is heavily used by women within age from 25 to 44 [16].

We directly sample a sub-graph snapshot of social network from Tumblr on August 2013, which contains 62.8 million nodes and 3.1 billion edges. Though this graph is not yet up-to-date, we believe that many network properties should be well preserved given the scale of this graph. Meanwhile, we sample about 586.4 million of Tumblr posts from August 10 to September 6, 2013. Unfortunately, Tumblr does not require users to fill in basic profile information, such as gender or location. Therefore, it is impossible for us to conduct user profile analysis as done in other works. In order to handle such large volume of data, most statistical patterns are computed through a MapReduce cluster, with some algorithms being tricky. We will skip the involved implementation details but concentrate solely on the derived patterns.

[8]http://flickr.com

[9]http://pinterest.com

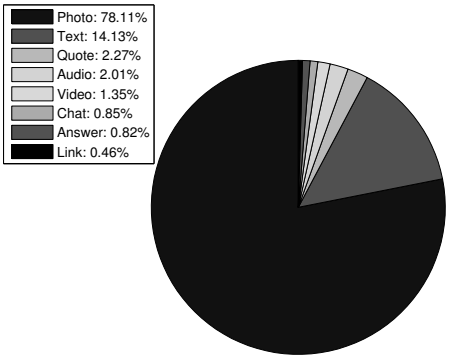


Figure 2: Distribution of Posts (Better viewed in color)

Most statistical patterns can be presented in three different forms: *probability density function* (PDF), *cumulative distribution function* (CDF) or *complementary cumulative distribution function* (CCDF), describing $Pr(X = x)$, $Pr(X \leq x)$ and $Pr(X \geq x)$ respectively, where X is a random variable and x is certain value. Due to the space limit, it is impossible to include all of them. Hence, we decide which form(s) to include depending on presentation and comparison convenience with other relevant papers. That is, if CCDF is reported in a relevant paper, we try to also report CCDF here so that rigorous comparison is possible.

Next, we study properties of Tumblr through different lenses, in particular, as a social network, a content generation website, and an information propagation platform, respectively.

3. TUMBLR AS SOCIAL NETWORK

We begin our analysis of Tumblr by examining its social network topology structure. Numerous social networks have been analyzed in the past, such as traditional blogosphere [21], Twitter [10; 11], Facebook [22], and instant messenger communication network [13]. Here we run an array of standard network analysis to compare with other networks, with results summarized in Table 1[10].

Degree Distribution. Since Tumblr does not require mutual confirmation when one follows another user, we represent the follower-followee network in Tumblr as a directed graph: in-degree of a user represents how many followers the user has attracted, while out-degree indicates how many other users one user has been following. Our sampled sub-graph contains 62.8 million nodes and 3.1 billion

[10]Even though we wish to include results over other popular social media networks like Pinterest, Sina Weibo and Instagram, analysis over those websites not available or just small-scale case studies that are difficult to generalize to a comprehensive scale for a fair comparison. Actually in the Table, we observe quite a discrepancy between numbers reported over a small twitter data set and another comprehensive snapshot.

Table 1: Comparison of Tumblr with other popular social networks. The numbers of Blogosphere, Twitter-small, Twitter-huge, Facebook, and MSN are obtained from [21; 10; 11; 22; 13], respectively. In the table, – implies the corresponding statistic is not available or not applicable; GCC denotes the giant connected component; the symbols in parenthesis *m, d, e, r* respectively represent *mean*, *median*, *the* 90% *effective diameter*, and *diameter* (the maximum shortest path in the network).

Metric	Tumblr	Blogosphere	Twitter-small	Twitter-huge	Facebook	MSN
#nodes	62.8M	143,736	87,897	41.7M	721M	180M
#links	3.1B	707,761	829,467	1.47B	68.7B	1.3B
in-degree distr	$\propto k^{-2.19}$	$\propto k^{-2.38}$	$\propto k^{-2.4}$	$\propto k^{-2.276}$	–	–
degree distr in r-graph	$\neq$ power-law	–	–	–	$\neq$ power-law	$\propto k^{0.8}e^{-0.03k}$
direction	directed	directed	directed	directed	undirected	undirected
reciprocity	29.03%	3%	58%	22.1%	–	–
degree correlation	0.106	–	–	> 0	0.226	–
avg distance	4.7(m), 5(d)	9.3(m)	–	4.1(m), 4(d)	4.7(m), 5(d)	6.6(m), 6(d)
diameter	5.4(e), $\geq$ 29(r)	12(r)	6(r)	4.8(e), $\geq$ 18(r)	$<$ 5(e)	7.8(e), $\geq$ 29(r)
GCC coverage	99.61%	75.08%	93.03%	–	99.91%	99.90%

edges. Within this social graph, 41.40% of nodes have 0 in-degree, and the maximum in-degree of a node is 4.06 million. By contrast, 12.74% of nodes have 0 out-degree, the maximum out-degree of a node is 155.5k. Top popular Tumblr users include *equipo*[11], *instagram*[12], and *woodendreams*[13]. This indicates the media characteristic of Tumblr: the most popular user has more than 4 million audience, while more than 40% of users are purely audience since they don't have any followers.

Figure 3(a) demonstrates the distribution of in-degrees in the blue curve and that of out-degrees in the red curve, where y-axis refers to the cumulated density distribution function (CCDF): the probability that accounts have at least k in-degrees or out-degrees, i.e., $P(K >= k)$. It is observed that Tumblr users' in-degree follows a power-law distribution with exponent -2.19, which is quite similar from the power law exponent of Twitter at -2.28 [11] or that of traditional blogs at -2.38 [21]. This also confirms with earlier empirical observation that most social network have a power-law exponent between -2 and -3 [6].

In regard to out-degree distribution, we notice the red curve has a big drop when out-degree is around 5000, since there was a limit that ordinary Tumblr users can follow at most 5000 other users. Tumblr users' out-degree does not follow a power-law distribution, which is similar to blogosphere of traditional blogging [21].

If we explore user's in-degree and out-degree together, we could generate normalized 3-D histogram in Figure 3(b). As both in-degree and out-degree follow the heavy-tail distribution, we only zoom in those user who have less than 2^{10} in-degrees and out-degrees. Apparently, there is a positive correlation between in-degree and out-degree because of the dominance of diagonal bars. In aggregation, a user with low in-degree tends to have low out-degree as well, even though some nodes, especially those top popular ones, have very imbalanced in-degree and out-degree.

Reciprocity. Since Tumblr is a directed network, we would like to examine the reciprocity of the graph. We derive the backbone of the Tumblr network by keeping those reciprocal connections only, i.e., user a follows b and vice versa. Let *r-graph* denote the corresponding reciprocal graph. We found 29.03% of Tumblr user pairs have reciprocity relationship, which is higher than 22.1% of reciprocity on Twitter [11] and 3% of reciprocity on Blogosphere [21], indicating a stronger interaction between users in the network. Figure 3(c) shows the distribution of degrees in the r-graph. There is a turning point due to the Tumblr limit of 5000 followees for ordinary users. The reciprocity relationship on Tumblr does not follow the power law distribution, since the curve mostly is convex, similar to the pattern reported over Facebook[22].

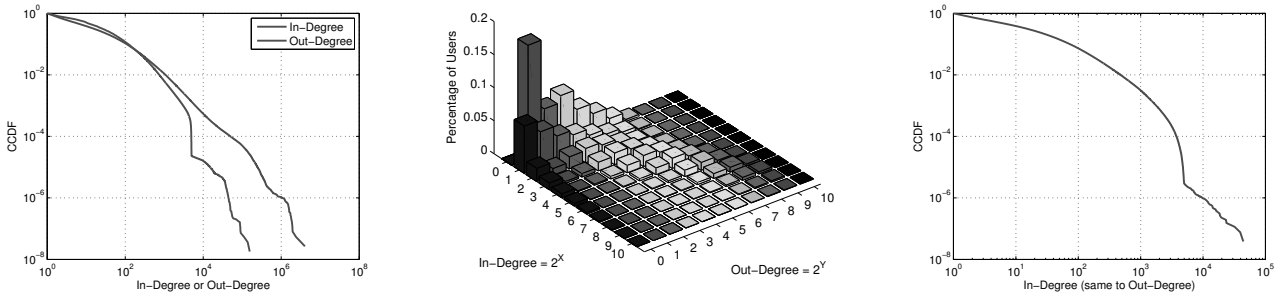
Meanwhile, it has been observed that one's degree is correlated with the degree of his friends. This is also called *degree correlation* or *degree assortativity* [18; 19]. Over the derived r-graph, we obtain a correlation of 0.106 between terminal nodes of reciprocate connections, reconfirming the positive degree assortativity as reported in Twitter [11]. Nevertheless, compared with the strong social network Facebook, Tumblr's degree assortativity is weaker (0.106 vs. 0.226).

Degree of Separation. Small world phenomenon is almost universal among social networks. With this huge Tumblr network, we are able to validate the well-known "six degrees of separation" as well. Figure 4 displays the distribution of the shortest paths in the network. To approximate the distribution, we randomly sample 60,000 nodes as seed and calculate for each node the shortest paths to other nodes. It is observed that the distribution of paths length reaches its mode with the highest probability at 4 hops, and has a median of 5 hops. On average, the distance between two connected nodes is 4.7. Even though the longest shortest path in the approximation has 29 hops, 90% of shortest paths are within 5.4 hops. All these numbers are close to those reported on Facebook and Twitter, yet significantly smaller than that obtained over blogosphere and instant messenger network [13].

Component Size. The previous result shows that those users who are connected have a small average distance. It relies on the assumption that most users are connected to each other, which we shall confirm immediately. Because the Tumblr graph is directed, we compute out all weakly-connected components by ignoring the direction of edges. It turns out the giant connected component (GCC) encompasses 99.61% of nodes in the graph. Over the derived r-graph, 97.55% are residing in the corresponding GCC. This finding suggests the whole graph is almost just one connected component, and almost all users can reach others through just few hops.

To give a palpable understanding, we summarize commonly used network statistics in Table 1. Those numbers from other popular social networks (blogosphere, Twitter, Facebook, and MSN) are also included for comparison. From this compact view, it is obvious traditional blogs yield a significantly different network structure. Tumblr, even though originally proposed for blogging, yields a network structure that is more similar to Twitter and Facebook.

[11] http://equipo.tumblr.com
[12] http://instagram.tumblr.com
[13] http://woodendreams.tumblr.com



(a) in/out degree distribution (b) in/out degree correlation (c) degree distribution in r-graph

Figure 3: Degree Distribution of Tumblr Network

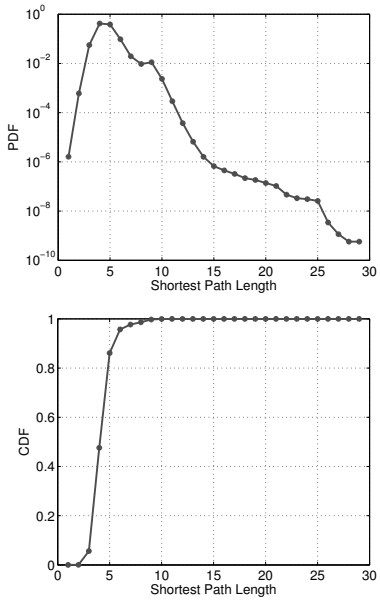


Figure 4: Shortest Path Distribution

	Text Post Dataset	Photo Caption Dataset
# Posts	21.5 M	26.3 M
Mean Post Length	426.7 Bytes	64.3 Bytes
Median Post Length	87 Bytes	29 Bytes
Max Post Length	446.0 K Bytes	485.5 K Bytes

Table 2: Statistics of User Generated Contents

4. TUMBLR AS BLOGOSPHERE FOR CONTENT GENERATION

As Tumblr is initially proposed for the purpose of blogging, here we analyze its user generated contents. As described earlier, *photo* and *text* posts account for more than 92% of total posts. Hence, we concentrate only on these two types of posts. One text post may contain URL, quote or raw message. In this study, we are mainly interested in the authentic contents generated by users. Hence, we extract raw messages as the content information of each text post, by removing quotes and URLs. Similarly, photo posts contains 3 categories of information: photo URL, quote photo caption, raw photo caption. While the photo URL might contain lots of additional meta information, it would require tremendous effort to analyze all images in Tumblr. Hence, we focus on raw photo captions as the content of each photo post. We end up with two datasets of content: one is *text post*, and the other is *photo caption*.

What's the effect of no length limit for post? Both Tumblr and Twitter are considered microblogging platforms, yet there is one key difference: Tumblr has no length limit while Twitter enforces the strict limitation of 140 bytes for each tweet. How does this key difference affect user post behavior?

It has been reported that the average length of posts on Twitter is 67.9 bytes and the median is 60 bytes[14]. Corresponding statistics of Tumblr are shown in Table 2. For the text post dataset, the average length is 426.7 bytes and the median is 87 bytes, which both, as expected, are longer than that of Twitter. Keep in mind Tumblr's numbers are obtained after removing all quotes, photos and URLs, which further discounts the discrepancy between Tumblr and Twitter. The big gap between mean and median is due to a small percentage of extremely long posts. For instance, the longest text post is 446K bytes in our sampled dataset. As for photo captions, naturally we expect it to be much shorter than text posts. The average length is around 64.3 bytes, but the median is only 29 bytes. Although photo posts are dominant in Tumblr, the number of text posts and photo captions in Table 2 are comparable, because majority of photo posts don't contain any raw photo captions.

A further related question: *is the 140-byte limit sensible?* We plot post length distribution of the text post dataset, and zoom into less than 280 bytes in Figure 5. About 24.48% of posts are beyond 140 bytes, which indicates that at least around one quarter of posts will have to be rewritten in a more compact version if the limit was enforced in Tumblr.

Blending all numbers above together, we can see at least two types of posts: one is more like posting a reference (URL or photo) with added information or short comments, the other is authentic user generated content like in traditional blogging. In other words, Tumblr is a mix of both types of posts, and its no-length-limit policy encourages its users to post longer high-quality content directly.

What are people talking about? Because there is no length limit on Tumblr, the blog post tends to be more meaningful, which al-

[14]http://www.quora.com/Twitter-1/What-is-the-average-length-of-a-tweet

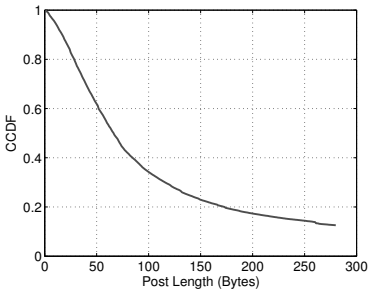


Figure 5: Post Length Distribution

Topic	Topical Keywords
Pop Music	music song listen iframe band album lyrics video guitar
Sports	game play team win video cookie ball football top sims fun beat league
Internet	internet computer laptop google search online site facebook drop website app mobile iphone
Pets	big dog cat animal pet animals bear tiny small deal puppy
Medical	anxiety pain hospital mental panic cancer depression brain stress medical
Finance	money pay store loan online interest buying bank apply card credit

Table 3: Topical Keywords from Text Post Dataset

lows us to run topic analysis over the two datasets to have an overview of the content. We run LDA [4] with 100 topics on both datasets, and showcase several topics and their corresponding keywords on Tables 3 and 4, which also show the high quality of textual content on Tumblr clearly. *Medical, Pets, Pop Music, Sports* are shared interests across 2 different datasets, although representative topical keywords might be different even for the same topic. *Finance, Internet* only attracts enough attentions from text posts, while only significant amount of photo posts show interest to *Photography, Scenery* topics. We want to emphasize that most of these keywords are semantically meaningful and representative of the topics.

Who are the major contributors of contents? There are two potential hypotheses. 1) One supposes those *socially popular users* post more. This is derived from the result that those popular users are followed by many users, therefore blogging is one way to attract more audience as followers. Meanwhile, it might be true that blogging is an incentive for celebrities to interact or reward their followers. 2) The other assumes that *long-term users* (in terms of registration time) post more, since they are accustomed to this service, and they are more likely to have their own focused communities or social circles. These peer interactions encourage them to generate more authentic content to share with others.

Do socially popular users or long-term users generate more contents? In order to answer this question, we choose a fixed time window of two weeks in August 2013 and examine how frequent each user blogs on Tumblr. We sort all users based on their in-degree (or duration time since registration) and then partition them into 10 equi-width bins. For each bin, we calculate the average blogging frequency. For easy comparison, we consider the maximal value of all bins as 1, and normalize the relative ratio for other bins. The results are displayed in Figure 6, where x-axis from left to right indicates increasing in-degree (or decreasing duration time).

Topic	Topical Keywords
Pets	cat dog cute upload kitty batch puppy pet animal kitten adorable
Scenery	summer beach sun sky sunset sea nature ocean island clouds lake pool beautiful
Pop Music	music song rock band album listen lyrics punk guitar dj pop sound hip
Photography	photo instagram pic picture check daily shoot tbt photography
Sports	team world ball win football club round false soccer league baseball
Medical	body pain skin brain depression hospital teeth drugs problems sick cancer blood

Table 4: Topical Keywords from Photo Caption Dataset

For brevity, we just show the result for text post dataset as similar patterns were observed over photo captions.

The patterns are strong in both figures. Those users who have higher in-degree tend to post more, in terms of both mean and median. One caveat is that what we observe and report here is merely correlation, and it does not derive causality. Here we draw a conservative conclusion that the social popularity is highly positively correlated with user blog frequency. A similar positive correlation is also observed in Twitter[11].

In contrast, the pattern in terms of user registration time is beyond our imagination until we draw the figure. Surprisingly, those users who either register earliest or register latest tend to post less frequently. Those who are in between are inclined to post more frequently. Obviously, our initial hypothesis about the incentive for new users to blog more is invalid. There could be different explanations in hindsight. Rather than guessing the underlying explanation, we decide to leave this phenomenon as an open question to future researchers.

As for reference, we also look at average post-length of users, because it has been adopted as a simple metric to approximate quality of blog posts [1]. The corresponding correlations are plot in Figure 7. In terms of post length, the tail users in social networks are the winner. Meanwhile, long-term or recently-joined users tend to post longer blogs. Apparently, this pattern is exactly opposite to post frequency. That is, the more frequent one blogs, the shorter the blog post is. And less frequent bloggers tend to have longer posts. That is totally valid considering each individual has limited time and resources. We even changed the post length to the maximum for each individual user rather than average, but the pattern remains still.

In summary, without the post length limitation, Tumblr users are inclined to write longer blogs, and thus leading to higher-quality user generated content, which can be leveraged for topic analysis. The social celebrities (those with large number of followers) are the main contributors of contents, which is similar to Twitter [24]. Surprisingly, long-term users and recently-registered users tend to blog less frequently. The post-length in general has a negative correlation with post frequency. The more frequently one posts, the shorter those posts tend to be.

5. TUMBLR FOR INFORMATION PROPAGATION

Tumblr offers one feature which is missing in traditional blog services: *reblog*. Once a user posts a blog, other users in Tumblr can reblog to comment or broadcast to their own followers. This en-

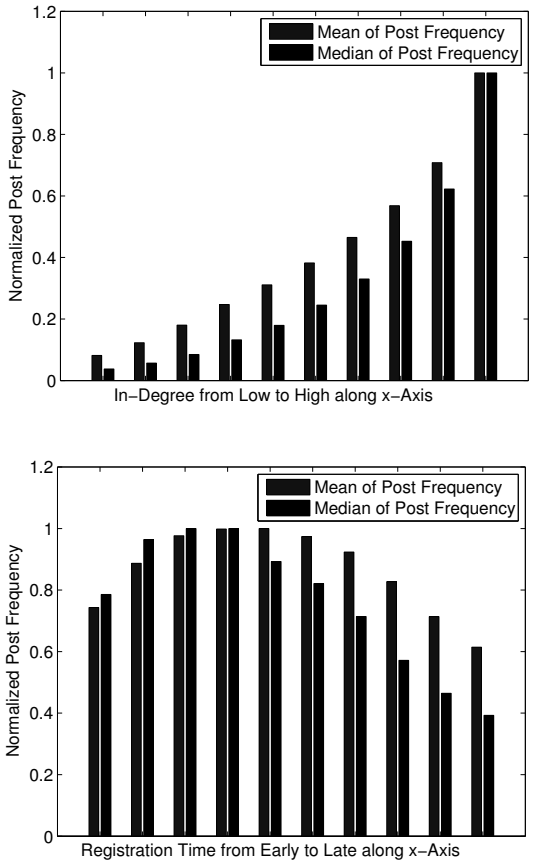


Figure 6: Correlation of Post Frequency with User In-degree or Duration Time since Registration

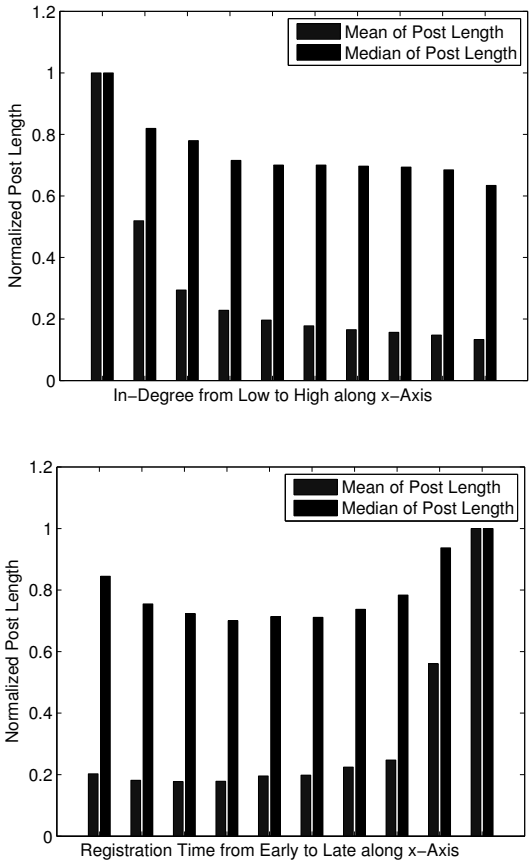


Figure 7: Correlation of Post Length with User In-degree or Duration Time since Registration

ables information to be propagated through the network. In this section, we examine the reblogging patterns in Tumblr. We examine all blog posts uploaded within the first 2 weeks, and count reblog events in the subsequent 2 weeks right after the blog is posted, so that there would be no bias because of the time window selection in our blog data.

Who are reblogging? Firstly, we would like to understand which users tend to reblog more? Those people who reblog frequently serves as the information transmitter. Similar to the previous section, we examine the correlation of reblogging behavior with users' in-degree. As shown in the Figure 8, social celebrities, who are the major source of contents, reblog a lot more compared with other users. This reblogging is propagated further through their huge number of followers. Hence, they serve as both content contributor and information transmitter. On the other hand, users who registered earlier reblog more as well. The socially popular and long-term users are the backbone of Tumblr network to make it a vibrant community for information propagation and sharing.

Reblog size distribution. Once a blog is posted, it can be reblogged by others. Those reblogs can be reblogged even further, which leads to a tree structure, which is called reblog cascade, with the first author being the root node. The reblog cascade size indicates the number of reblog actions that have been involved in the cascade. Figure 9 plots the distribution of reblog cascade sizes. Not surprisingly, it follows a power-law distribution, with majority of reblog cascade involving few reblog events. Yet, within a time window of two weeks, the maximum cascade could reach $116.6K$. In order to have a detailed understanding of reblog cascades, we zoom into the short head and plot the CCDF up to reblog cascade size equivalent to 20 in Figure 9. It is observed that only about 19.32% of reblog cascades have size greater than 10. By contrast, only 1% of retweet cascades have size larger than 10 [11]. The reblog cascades in Tumblr tend to be larger than retweet cascades in Twitter.

Reblog depth distribution. As shown in previous sections, almost any pair of users are connected through few hops. How many hops does one blog to propagate to another user in reality? Hence, we look at the reblog cascade depth, the maximum number of nodes to pass in order to reach one leaf node from the root node in the reblog cascade structure. Note that reblog depth and size are different. A cascade of depth 2 can involve hundreds of nodes if every other node in the cascade reblogs the same root node.

Figure 10 plots the distribution of number of hops: again, the reblog cascade depth distribution follows a power law as well according to the PDF; when zooming into the CCDF, we observe that only 9.21% of reblog cascades have depth larger than 6. That is, majority of cascades can reach just few hops, which is consistent with the findings reported over Twitter [3]. Actually, 53.31% of cascades in Tumblr have depth 2. Nevertheless, the maximum depth among all cascades can reach 241 based on two week data. This looks un-

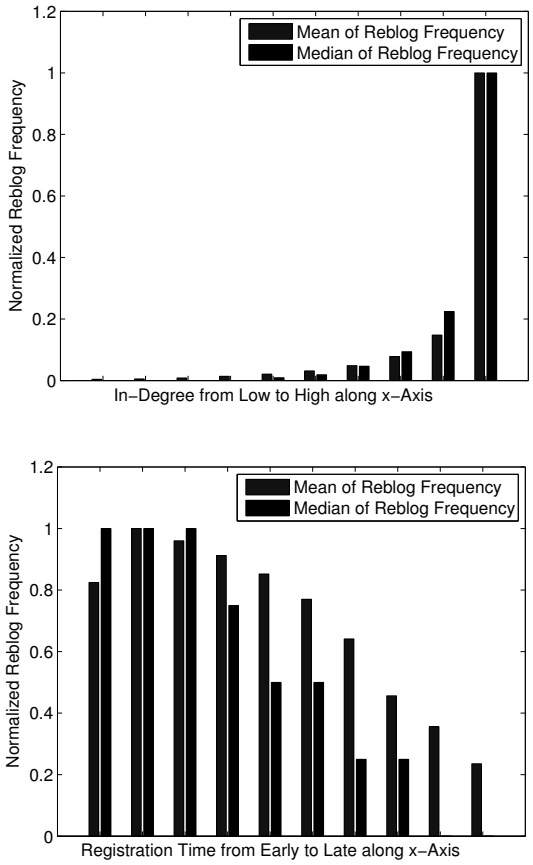


Figure 8: Correlation of Reblog Frequency with User In-degree or Duration Time since Registration

likely at first glimpse, considering any two users are just few hops away. Indeed, this is because users can add comment while reblogging, and thus one user is likely to involve in one reblog cascade multiple times. We notice that some Tumblr users adopt reblog as one way for conversation or chat.

Reblog Structure Distribution. Since most reblog cascades are few hops, here we show the cascade tree structure distribution up to size 5 in Figure 11. The structures are sorted based on their coverage. Apparently, a substantial percentage of cascades (36.05%) are of size 2, i.e., a post being reblogged merely once. Generally speaking, a reblog cascade of a flat structure tends to have a higher probability than a reblog cascade of the same size but with a deep structure. For instance, a reblog cascade of size 3 have two variants, of which the flat one covers 9.42% cascade while the deep one drops to 5.85%. The same patten applies to reblog cascades of size 4 and 5. In other words, it is easier to spread a message widely rather than deeply in general. This implies that it might be acceptable to consider only the cascade effect under few hops and focus those nodes with larger audience when one tries to maximize influence or information propagation.

Temporal patten of reblog. We have investigated the information propagation spatially in terms of network topology, now we study how fast for one blog to be reblogged? Figure 12 displays the distribution of time gap between a post and its first reblog. There is a strong bias toward recency. The larger the time gap since a blog

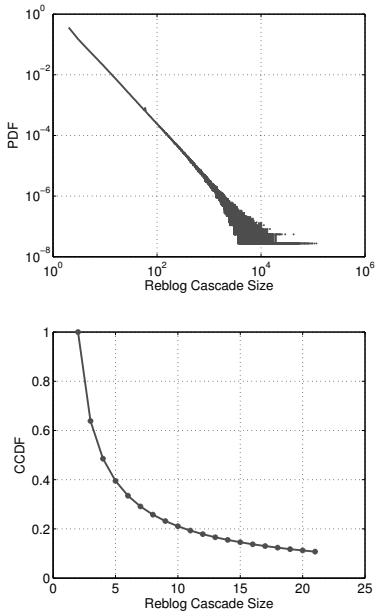


Figure 9: Distribution of Reblog Cascade Size

is posted, the less likely it would be reblogged. 75.03% of first reblog arrive within the first hour since a blog is posted, and 95.84% of first reblog appears within one day. Comparatively, It has been reported that "half of retweeting occurs within an hour and 75% under a day" [11] on Twitter. In short, Tumblr reblog has a strong bias toward recency, and information propagation on Tumblr is fast.

6. RELATED WORK

There are rich literatures on both existing and emerging online social network services. Statistical patterns across different types of social networks are reported, including traditional blogosphere [21], user-generated content platforms like Flickr, Youtube and LiveJournal [15], Twitter [10; 11], instant messenger network [13], Facebook [22], and Pinterest [7; 20]. Majority of them observe shared patterns such as long tail distribution for user degrees (power law or power law with exponential cut-off), small (90% quantile effective) diameter, positive degree association, homophily effect in terms of user profiles (age or location), but not with respect to gender. Indeed, people are more likely to talk to the opposite sex [13]. The recent study of Pinterest observed that ladies tend to be more active and engaged than men [20], and women and men have different interests [5]. We have compared Tumblr's patterns with other social networks in Table 1 and observed that most of those trend hold in Tumblr except for some number difference.

Lampe *et al.* [12] did a set of survey studies on Facebook users, and shown that people use Facebook to maintain existing offline connections. Java *et al.* [10] presented one of the earliest research paper for Twitter, and found that users leverage Twitter to talk their daily activities and to seek or share information. In addition, Schwartz [7] is one of the early studies on Pinterest, and from a statistical point of view that female users repin more but with fewer followers than male users. While Hochman and Raz [8] published an early paper using Instagram data, and indicated differences in local color usage, cultural production rate, for the analysis of location-based visual information flows.

Existing studies on user influence are based on social networks or

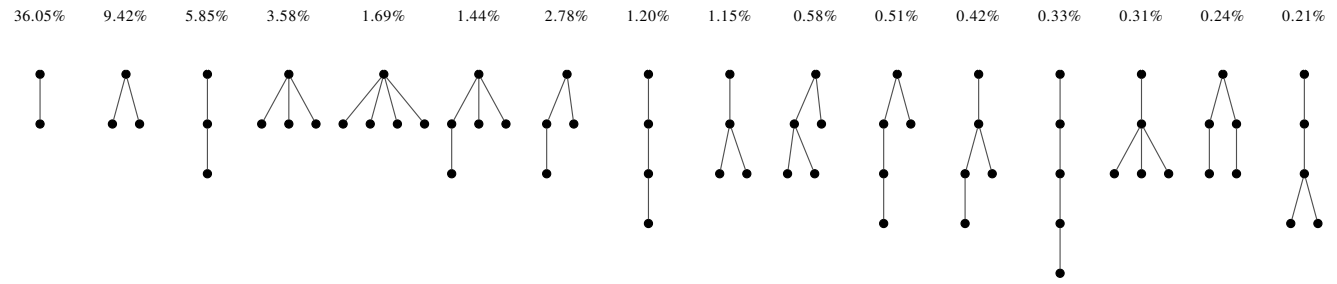


Figure 11: Cascade Structure Distribution up to Size 5. The percentage at the top is the coverage of cascade structure.

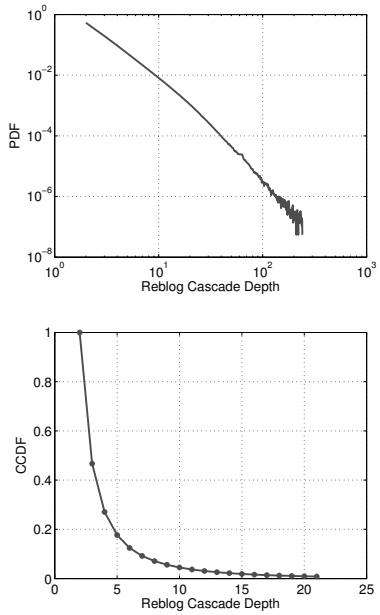


Figure 10: Distribution of Reblog Cascade Depth

content analysis. McGlohon *et al.* [14] found topology features can help us distinguish blogs, the temporal activity of blogs is very non-uniform and bursty, but it is self-similar. Bakshy *et al.* [3] investigated the attributes and relative influence based on Twitter follower graph, and concluded that word-of-mouth diffusion can only be harnessed reliably by targeting large numbers of potential influencers, thereby capturing average effects. Hopcroft *et al.* [9] studied the Twitter user influence based on two-way reciprocal relationship prediction. Weng *et al.* [23] extended PageRank algorithm to measure the influence of Twitter users, and took both the topical similarity between users and link structure into account. Kwak *et al.* [11] study the topological and geographical properties on the entire Twittersphere and they observe some notable properties of Twitter, such as a non-power-law follower distribution, a short effective diameter, and low reciprocity, marking a deviation from known characteristics of human social networks.

However, due to data access limitation, majority of the existing scholar papers are based on either Twitter data or traditional blogging data. This work closes the gap by providing the first overview of Tumblr so that others can leverage as a stepstone to investigate more over this evolving social service or compare with other related services.

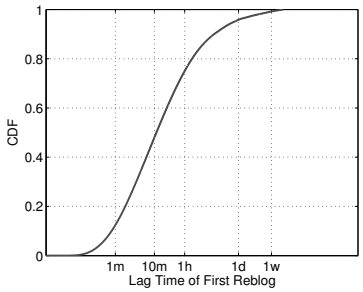


Figure 12: Distribution of Time Lag between a Blog and its first Reblog

7. CONCLUSIONS AND FUTURE WORK

In this paper, we provide a statistical overview of Tumblr in terms of social network structure, content generation and information propagation. We show that Tumblr serves as a social network, a blogosphere and social media simultaneously. It provides high quality content with rich multimedia information, which offers unique characteristics to attract youngsters. Meanwhile, we also summarize and offer as rigorous comparison as possible with other social services based on numbers reported in other papers. Below we highlight some key findings:

- With multimedia support in Tumblr, photos and text account for majority of blog posts, while audios and videos are still rare.
- Tumblr, though initially proposed for blogging, yields a significantly different network structure from traditional blogosphere. Tumblr's network is much denser and better connected. Close to 29.03% of connections on Tumblr are reciprocate, while blogosphere has only 3%. The average distance between two users in Tumblr is 4.7, which is roughly half of that in blogosphere. The giant connected component covers 99.61% of nodes as compared to 75% in blogosphere.
- Tumblr network is highly similar to Twitter and Facebook, with power-law distribution for in-degree distribution, non-power law out-degree distribution, positive degree associativity for reciprocate connections, small distance between connected nodes, and a dominant giant connected component.
- Without post length limitation, Tumblr users tend to post longer. Approximately 1/4 of text posts have authentic contents beyond 140 bytes, implying a substantial portion of high quality blog posts for other tasks like topic

- Those social celebrities tend to be more active. They post analysis and text mining. and reblog more frequently, serving as both content generators and information transmitters. Moreover, frequent bloggers like to write short, while infrequent bloggers spend more effort in writing longer posts.
- In terms of duration since registration, those long-term users and recently registered users post less frequently. Yet, long-term users reblog more.
- Majority of reblog cascades are tiny in terms of both size and depth, though extreme ones are not uncommon. It is relatively easier to propagate a message wide but shallow rather than deep, suggesting the priority for influence maximization or information propagation.
- Compared with Twitter, Tumblr is more vibrant and faster in terms of reblog and interactions. Tumblr reblog has a strong bias toward recency. Approximately $3/4$ of the first reblogs occur within the first hour and 95.84% appear within one day.

This snapshot research is by no means to be complete. There are several directions to extend this work. First, some patterns described here are correlations. They do not illustrate the underlying mechanism. It is imperative to differentiate correlation and causality [2] so that we can better understand the user behavior. Secondly, it is observed that Tumblr is very popular among young users, as half of Tumblr's visitor base being under 25 years old. Why is it so? We need to combine content analysis, social network analysis, together with user profiles to figure out. In addition, since more than 70% of Tumblr posts are images, it is necessary to go beyond photo captions, and analyze image content together with other meta information.

8. REFERENCES

[1] N. Agarwal, H. Liu, L. Tang, and P. S. Yu. Identifying the influential bloggers in a community. In *Proceedings of WSDM*, 2008.

[2] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *Proceedings of KDD*, 2008.

[3] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of WSDM*, 2011.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan:. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] S. Chang, V. Kumar, E. Gilbert, and L. Terveen. Specialization, homophily, and gender in a social curation site: Findings from pinterest. In *Proceedings of The 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW'14, 2014.

[6] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *arXiv*, 706, 2007.

[7] E. Gilbert, S. Bakhshi, S. Chang, and L. Terveen. 'i need to try this!': A statistical overview of pinterest. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2013.

[8] N. Hochman and R. Schwartz. Visualizing instagram: Tracing cultural visual rhythms. In *Proceedings of the Workshop on Social Media Visualization (SocMedVis) in conjunction with ICWSM*, 2012.

[9] J. E. Hopcroft, T. Lou, and J. Tang. Who will follow you back?: reciprocal relationship prediction. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1137–1146, 2011.

[10] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *WebKDD/SNA-KDD '07*, pages 56–65, New York, NY, USA, 2007. ACM.

[11] H. Kwak, C. Lee, H. Park, and S. B. Moon. What is twitter, a social network or a news media. In *Proceedings of 19th International World Wide Web Conference (WWW)*, 2010.

[12] C. Lampe, N. Ellison, and C. Steinfield. A familiar face(book): Profile elements as signals in an online social network. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2007.

[13] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 915–924, New York, NY, USA, 2008. ACM.

[14] M. McGlohon, J. Leskovec, C. Faloutsos, M. Hurst, and N. S. Glance. Finding patterns in blog shapes and blog evolution. In *Proceedings of ICWSM*, 2007.

[15] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, New York, NY, USA, 2007. ACM.

[16] S. Mittal, N. Gupta, P. Dewan, and P. Kumaraguru. The pinbang theory: Discovering the pinterest world. *arXiv preprint arXiv:1307.4952*, 2013.

[17] B. Nardi, D. J. Schiano, S. Gumbrecht, and L. Swartz. Why we blog. *Commun. ACM*, 47(12):41–46, 2004.

[18] M. E. J. Newman. Assortative mixing in networks. *Physical review letters, 89(20): 208701*, 2002.

[19] M. E. J. Newman. Mixing patterns in networks. *Physical Review E, 67(2): 026126*, 2003.

[20] R. Ottoni, J. P. Pesce, D. Las Casas, G. Franciscani, P. Kumaruguru, and V. Almeida. Ladies first: Analyzing gender roles and behaviors in pinterest. *Proceedings of ICWSM*, 2013.

[21] X. Shi, B. Tseng, , and L. A. Adamic. Looking at the blogosphere topology through different lenses. In *Proceedings of ICWSM*, 2007.

[22] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[23] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of WSDM*, pages 1137–1146, 2010.

[24] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts. Who says what to whom on twitter. In *Proceedings of WWW 2011*.

Change Detection in Streaming Data in the Era of Big Data: Models and Issues

Dang-Hoan Tran
Vietnam Maritime University
484 Lach Tray, Ngo Quyen
Haiphong, Vietnam
dang-hoan.tran@vimaru.edu.vn

Mohamed Medhat Gaber
School of Computing Science and Digital Media, Robert Gordon University
Riverside East Garthdee Road
Aberdeen
AB10 7GJ, UK
m.gaber1@rgu.ac.uk

Kai-Uwe Sattler
Ilmenau University of Technology
PO Box 100565
Ilmenau, Germany
kus@tu-ilmenau.de

ABSTRACT

Big Data is identified by its three *Vs*, namely velocity, volume, and variety. The area of data stream processing has long dealt with the former two *Vs* velocity and volume. Over a decade of intensive research, the community has provided many important research discoveries in the area. The third *V* of Big Data has been the result of social media and the large unstructured data it generates. Streaming techniques have also been proposed recently addressing this emerging need. However, a hidden factor can represent an important fourth *V*, that is variability or change. Our world is changing rapidly, and accounting to variability is a crucial success factor. This paper provides a survey of change detection techniques as applied to streaming data. The review is timely with the rise of Big Data technologies, and the need to have this important aspect highlighted and its techniques categorized and detailed.

1. INTRODUCTION

Today's world is changing very fast. The changes occur in every aspects of life. Therefore, the ability to detect, adapt, and react to the change play an important role in all aspects of life. The physical world is often represented in some model or some information system. The changes in the physical world are reflected in terms of the changes in data or model built from data. Therefore, the nature of data is changing.

The advance of technology results in the data deluge. The data volume is increasing with an estimated rate of 50% per year [39]. Data flood makes traditional methods including traditional distributed framework and parallel models inappropriate for processing, analyzing, storing, and understanding these massive data sets. Data deluge needs a new generation of computing tools that Jim Gray calls the 4^{th} paradigm in scientific computing [25]. Recently, there have been some emerging computing paradigms that meet the requirements of Big Data as follows. Parallel batch processing model only deals with the stationary massive data [17]. However, evolving data continuously arrives with high speed. In fact, online data stream processing is the main approach to dealing with the problem of three characteristics of Big Data including big volume, big velocity, and big variety. Streaming data processing is a model of Big Data processing. Streaming data is temporal data in nature. In addition to the temporal nature, streaming data may include spatial characteristics. For example, geographic information systems can produce spatial-temporal data stream. Streaming data processing and mining have been deploying in real-world systems such as InforSphere Streams (IBM)[1], Rapidminer Streams Plugin [2], StreamBase [3], MOA [4], AnduIN [5]. In order to deal with the high-speed data streams, a hybrid model that combines the advantages of both parallel batch processing model and streaming data processing model is proposed. Some projects for such hybrid model include S4 [6], Storm [7], and Grok [8] .

One of these challenges facing data stream processing and mining is the changing nature of streaming data. Therefore, the ability to identify trends, patterns, and changes in the underlying processes generating data contributes to the success of processing and mining massive high-speed data streams.

A model of continuous distributed monitoring has been recently proposed to deal with streaming data coming from multiple sources. This model has many observers where each observer monitors a single data stream. The goal of continuous distributed monitoring is to perform some tasks that need to aggregate the incoming data from the observers. The continuous distributed monitoring is applied to monitor networks such as sensor networks, social networks, networks of ISP [11].

Change detection is the process of identifying differences in the state of an object or phenomenon by observing it at different times or different locations in space. In the streaming context, change detection is the process of segmenting a data stream into different segments by identifying the points where the stream dynamics change [53]. A change detection method consists of the following tasks: change detection and localization of change. Change detection identifies whether a change occurs, and responds to the presence of such change. Besides change detection, localization of changes determines the location of change. The problem of locating the change has been studied in statistics in the problems of change point detection.

This paper presents the background issues and notation relevant to the problem of change detection in data streams.

2. CHANGE DETECTION IN STREAMING DATA

[1] http://www-01.ibm.com/software/data/infosphere/streams/
[2] http://www-ai.cs.uni-dortmund.de/auto?self=$eit184kc
[3] http://www.streambase.com/
[4] http://moa.cs.waikato.ac.nz/
[5] http://www.tu-ilmenau.de/dbis/research/anduin/
[6] http://incubator.apache.org/s4/
[7] https://github.com/nathanmarz/storm/wiki/Tutorial
[8] https://www.numenta.com/grok_info.html

Streaming computational model is considered one of the widely-used models for processing and analyzing massive data. Streaming data processing helps the decision-making process in real-time. A data stream is defined as follows.

DEFINITION 1. *A data stream is an infinite sequence of elements*

$$S = \{(X_1, T_1), .., (X_j, T_j), ...\} \quad (1)$$

Each element is a pair (X_j, T_j) *where* X_j *is a d-dimensional vector* $X_j = (x_1, x_2, ..., x_d)$ *arriving at the time stamp* T_j*. Time-stamp is defined over discrete domain with a total order. There are two types of time-stamps: explicit time-stamp is generated when data arrive; implicit time-stamp is assigned by some data stream processing system.*

Streaming data includes the fundamental characteristics as follows. First, data arrives continuously. Second, streaming data evolves overtime. Third, streaming data is noisy, corrupted. Forth, timely interfering is important. From the characteristics of streaming data and data stream model, data stream processing and mining pose the following challenges. First, as streaming data arrives rapidly, the techniques of streaming data process and analysis must keep up with the data rate to prevent from the loss of important information as well as avoid data redundancy. Second, as the speed of streaming data is very high, the data volume overcomes the processing capacity of the existing systems. Third, the value of data decreases over time, the recent streaming data is sufficient for many applications. Therefore, one can only capture and process the data as soon as it is generated.

2.1 Change Detection: Definitions and Notation

This section presents concepts and classification of changes and change detection methods. To develop a change detection method, we should understand what a change is.

DEFINITION 2. *Change is defined as the difference in the state of an object or phenomenon over time and/or space [52; 1].*

In the view of system, change is the process of transition from a state of a system to another. In other words, a change can be defined as the difference between an earlier state and a later state. An important distinction between change and difference is that a change refers to a transition in the state of an object or a phenomenon overtime while the difference means the dissimilarity in the characteristics of two objects. A change can reflect the short-term trend or long-term trend. For example, a stock analyst may be interested in the short-term change of the stock price.

Change detection is defined as the process of identifying differences in the state of an object or phenomenon by observing it at different times [54]. In the above definition, a change is detected on the basis of differences of an object at different times without considering the differences of an object in locations in space. In many real world applications, changes can occur both in terms of both time and space. For example, multiple spatial-temporal data streams representing triple (latitude, longitude, time) are created in traffic information systems using GPS [23]. Hence, change detection can be defined as follows.

DEFINITION 3. *Change detection is the process of identifying differences in the state of an object or phenomenon by observing it at different times and/or different locations in space.*

A distinction between concept drift detection and change detection is that concept drift detection focuses on the labeled data while change detection can deal with both labeled and unlabeled data. Change analysis both detects and explains the change. Hido et al. [26] proposed a method for change analysis by using supervised learning.

DEFINITION 4. *Change point detection is identifying time points at which properties of time series data change[32]*

Depending on specific application, change detection can be called in different terms such as burst detection, outlier detection, or anomaly detection. Burst detection a special kind of change detection. Burst is a period on stream with aggregated sum exceeding a threshold [31]. Outlier detection is a special kind of change detection. Anomaly detection can be seen as a special type of change detection in streaming data.

To find a solution to the problem of change detection, we should consider the aspects of change of the system in which we want to detect. As shown in [52], the following aspects of change, which must be considered, include subject of change, type of change, cause of change, effect of change, response of change, temporal issues, and spatial issues. In particular, to design an algorithm for detecting changes in sensor streaming data, the major questions we need to answer include: What is the system in which the changes need to be detected? What are the principles used to model the problem? What is data type? What are the constraints of the problem? What is the physical subject of change? What is the meaning of change to the user? How to respond and react to this change? How to visualize this change?

A change detection method can fall into one of two types: batch change detection and sequential change detection. Given a sequence of N observations $x_1, .., x_N$, where N is invariant, the task of a batch change detection method is deciding whether a change occurs at some point in the sequence by using all N available observations. When the arriving speed of data is too high, batch change detection is suitable. In other words, change detection method using two adjacent windows model will be used. However, the drawback of batch change detection method is that its running time is very large when detecting changes in a large amount of data. In contrast, the sequential change detection problem is based on the observations so far. If no change is detected, the next observation is processed. Whenever a change is detected, the change detector is reset.

Change detection methods can be classified into the following approaches: threshold-based change detection method; state-based change detection method; trend-based change detection method.

A change detection algorithm should meet three main requirements [37]: accuracy, promptness, and online. The algorithm should detect as many as possible actual change points and generate as few as possible false alarms. The algorithm should detect change point as early as possible. The algorithm should be efficient sufficient for a real time environment.

Change detection in data stream allows us to identify the time-evolving trends, and time-evolving patterns. Research issues on mining changes in data streams include modeling and representation of changes, change-adaptive mining method, and interactive exploration of changes [19]. Change detection plays an important role in the field of data stream analysis. Since change in model may convey interesting time-dependent information and knowledge, the change of the data stream can be used for understanding the nature of several applications. Basically, interesting research problems on mining changes in data streams can be classified into three categories: modeling and representation of changes, mining methods, and interactive exploration of changes. Change detection algorithm can be used as a sub-procedure in many other data stream mining algorithms in order to deal with the changing data in data streams [28; 4]. A definition of change detection for streaming data is given as follows

DEFINITION 5. *Change detection is the process of segment-*

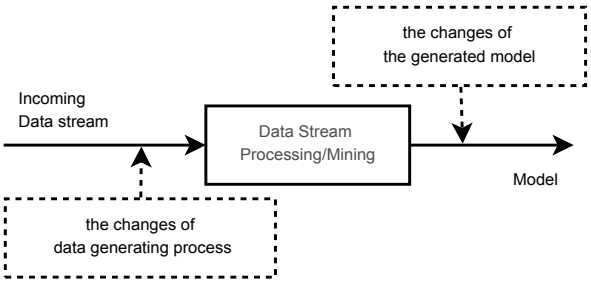


Figure 1: A general diagram for detecting changes in data stream

ing a data stream into different segments by identifying the points where the stream dynamics changes [53].

As data streams evolve overtime in nature, there is growing emphasis on detecting changes not only in the underlying data distribution, but also in the models generated by data stream process and data stream mining. As can be seen in Figure 1, a change can occur in the data stream, or the streaming model. Therefore, there are two types of the problems of change detection: change detection in the data generating process and change detection in the model generated by a data stream processing, or mining. The fundamental issues of detecting changes in data streams includes characterizing and quantifying of changes and detecting changes. A change detection method in streaming data needs a trade-off among space-efficiency, detection performance, and time-efficiency.

2.2 Change Detection Methods in Streaming Data

Over the last 50 years, change detection has been widely studied and applied in both academic research and industry. For example, it has been studied for a long time in the following fields: statistics, signal processing, and control theory. In recent years many change detection methods have been proposed for streaming data. The approaches to detecting changes in data stream can be classified as follows.

- ***Data stream model:*** A data stream can fall into one of the following models: time series model, cash register model, and turnstile model [41]. On the basis of the data stream model, there are change detection algorithms developed for the corresponding data stream model. Krishnamurthy et al presented a sketch-based change detection method for the most general streaming model Turnstile model [35].

- ***Data characteristics:*** Change detection methods can be classified on the basis of the data characteristics of streaming data such as data dimensionality, data label, and data type. A data item coming from the data stream can be univariate or multi-dimensional. It would be great if we could develop a general algorithm able to detect changes in both univariate and multidimensional data streams. Change detection algorithms in streaming multivariate data have been presented [14; 34; 36]. Data streams can be classified into categorial data stream and numerical data stream. We can develop the change detection algorithm for categorial data stream or numerical data stream. In real world applications, each data item in data stream may include multiple attributes of both numerical and categorial data. In such situations, these data streams can be projected by each attribute or group of attributes. Change detection methods can be applied to the corresponding projected data streams afterwards. Data streams are classified into labeled data stream and unlabeled data streams. A labeled data stream is one whose individual example is associated with a given class label, otherwise, it is unlabeled data stream. A change detection algorithm that identifies changes in the labeled data stream is supervised change detection [34; 5], while one detecting changes in the unlabeled data stream is called unsupervised change detection algorithm [7]. The advantage of the supervised approach is that the detection accuracy is high. However, the ground truth data must be generated. Thus a unsupervised change detection approach is preferred to the supervised one in case the ground truth data is unavailable.

- ***Completeness of statistical information:*** On the basis of the completeness of statistical information, a change detection algorithm can fall into one of three following categories. Parametric change detection schemes are based on knowing the full prior information before and after change. For example, in the distributional change detection methods, the data distributions before and after change are known [41; 42]. A recently introduced method to detecting changes in order stock streams is a parametric method in which the distribution of stream of stock orders confide to the Poisson distribution [37]. The advantage of parametric change detection approaches is that they can produce a higher accurate result than semi-parametric and nonparametric methods. However, in many real-time applications, data may not confine to any standard distribution, thus parametric approaches are inapplicable. Semi-parametric methods are based on the assumption that the distribution of observations belongs to some class of distribution function, and parameters of the distribution function change in disorder moments. Recently, Kuncheva [36] has proposed a semi-parametric method using a semi-parametric log-likelihood for testing a change. Nonparametric methods make no distribution assumptions on the data. Nonparametric methods for detecting changes in the underlying data distribution includes Wilcoxon, kernel method, Kullback-Leiber distance, and Kolmogorov-Smirnov test. Nonparametric methods can be classified into two categories: nonparametric methods using window [33]; nonparametric methods without using window [27]. We have paid particular attention to the nonparametric change detection methods using window because in many real-world applications, the distributions of both null hypothesis and alternative hypothesis are unknown in advance. Furthermore, we are only interested in recent data. A common approach to identifying the change is to comparing two samples in order to find out the difference between them, which is called two-sample change detection, or window-based change detection. As data stream is infinite, a sliding window is often used to detect changes. Window based change detection incurs the high delay [37]. Window-based change detection scheme is based on the dissimilarity measure between two distributions or synopses extracted from the reference window and the current window.

- ***Velocity of data change:*** Aggarwal proposes a framework that can deal with the changes in both spatial velocity profile and temporal velocity profile [1; 2]. In this approach, the changes in data density occurring at each location are estimated by estimating velocity density in some user-defined temporal window. An important advantage of this approach is that it visualizes the changes. This visualization of changes helps user understand the changes intuitively.

- ***Speed of response:*** If a change detection method needs to react to the detected changes as fast as possible, the

quickest detection of change should be proposed. Quickest change detection can help a system make a timely alarm. Timely alarm warning is benefit for economical. In some cases, it may save the human life such as in fire-fighting system. Change detection methods using two overlapping windows can quickly react to the changes in streaming data while methods using adjacent windows model may incur the high delay. As change can be abrupt change or gradual change, there exists the abrupt change detection algorithm and gradual change detection algorithm [46; 40].

- ***Decision making methodology:*** Based on the decision making methodology, a change detection method can fall into one of the following categories: rank-based method [33], density-based method [55], information-theoretic method [15]. A change detection problem can be also classified into batch change detection and sequential change detection. Based on detection delay that a change detector suffers from, a change detection methods can fall into one of two following types: real-time change detection, and retrospective change detection. Based on the spatial or temporal characteristics of data, change detection algorithm can fall into one of three kinds: spatial change detection; temporal change detection; or spatio-temporal change detection [6].

- ***Application:*** On the basis of applications that generate data streams, data streams can be classified as into transactional data stream, sensor data stream, network data stream, stock order data stream, astronomy data stream, video data stream, etc. Based on the specific applications, there are the change detection methods for the corresponding applications such as change detection methods for sensor streaming data [56], change detection methods for transactional streaming data [45; 57; 8]. For example, van Leeuwen and Siebes [57] have presented a change detection method for transactional streaming data based on the principle of Minimum Description Length.

- ***Stream processing methodology:*** Based on methodology for processing data stream, a data stream can be classified into online data stream and off-line data stream [38]. In some work, an online data stream is called a live stream while an off-line data stream is called archived data stream [18]. Online data stream needs to be processed online because of its high speed. Such online data streams include streams of stock ticker, streams of network measurements, and streams of sensor data,etc. Off-line stream is a sequence of updates to warehouses or backup devices. The queries over the off-line streams can be processed off-line. However, as it is insufficient time to process off-line streams, techniques for summarizing data are necessary. In off-line change detection method, the entire data set is available for the analysis process to detect the change. The online method detects the change incrementally based on the recently incoming data item. An important distinction between off-line method and online one is that the online method is constrained by the detection and reaction time due to the requirement of real-time applications while the off-line is free from the detection time, and reaction time. Methods for detecting changes can be useful for streaming data warehouses where both live streams of data and archived data streams are available [24; 29]. In this work, we focus on developing the methods for detecting changes in online data streams, in particular, sensor data streams.

The first work on model-based change detection proposed by [21; 22] is FOCUS. The central idea behind FOCUS is that the models can be divided into structural and measurement components. To detect deviation between two models, they compare specific parts of these corresponding models. The models obtained by data mining algorithms includes frequent item sets, decision trees, and clusters. The change in model may convey interesting information or knowledge of an event or phenomenon. Model change is defined in terms of the difference between two set of parameters of two models and the quantitative characteristics of two models. As such, model change detection is finding the difference between two set of parameters of two models and the quantitative characteristics of these two models. We should distinguish between detection of changes in data distribution by using models and detection of changes in model built from streaming data. While model change detection aims to identify the difference between two models, change detection in the underlying data distribution by using models is inferring the changes in two data sets from the difference between two models constructed from two data sets. The changes in the underlying data distribution can induce the corresponding changes in the model produced from the data generating process.

As models can be generated by statistics method or data mining methods, change detection in models can be classified into data mining model and statistical model. Two kinds of models we are interested in detecting changes are predictive model and explanatory model. Predictive model is used to predict the changes in the future. Detecting changes in the pattern can be beneficial for many applications. In explanatory model, a change that occurred is both detected and explained. There are some approaches to change detection: one-model approach, two-model approach, or multiple-model approach.

A model-based change detection algorithm consists of two phases as follows: model construction and change detection. First, a model is built by using some stream mining method such as decision tree, clustering, frequent pattern. Second, a difference measure between two models is computed based the characteristics of the model, this step is also called the quantification of model difference. Therefore, one fundamental issue here is to quantify the changes between two models and to determine criteria for making decision whether and when a change in the model occurs. Recently, some change detection methods in streaming data by clustering have been proposed [10; 3]. Based on the data stream mining model, we may have the corresponding problems of detecting changes in model as follows. Ikonomovska et al. [30] have presented an algorithm for learning regression trees from streaming data in the presence of concept drifts. Their change detection method is based on sequential statistical tests that monitoring the changes of the local error, at each node of tree, and inform the learning process of the local changes.

Detecting changes of stream cluster model has been received increasing attention. Zhou et al. [59] have presented a method for tracking the evolution of clusters over sliding windows by using temporal cluster features and the exponential histogram, which called exponential histogram of cluster features. Chen and Liu [9] have presented a framework for detecting the changes in clustering structures constructed from categorial data streams by using hierarchial entropy trees to capture the entropy characteristics of clusters, and then detecting changes in clustering structures based on these entropy characteristics.

Based on the data stream mining model, we may have the corresponding problems of detecting changes in model as follows [14]. Recently Ng and Dash [44] have introduced an algorithm for mining frequent patterns from evolving data streams. Their algorithm is capable of updating the frequent patterns based on the algorithms for detecting changes in the underlying data distributions. Two windows are used for change detection: the reference window and the current window. At the initial stage, the

reference is initialized with the first batch of transactions from data stream. The current window moves on the data stream and captures the next batch of transactions. Two frequent item sets are constructed from two corresponding windows by using the Apriori algorithm. A statistical test is performed on two absolute support values that are computed by the Apriori from the reference window and current window. Based on the statistical test, the deviation can be significant or insignificant. If the deviation is significant then a change in the data stream is reported. Chang and Lee [8] have presented a method for monitoring the recent change of frequent item sets from data stream by using sliding window.

2.3 Design Methodology

There are two design methodologies for developing the change detection algorithms in streaming data. The first methodology is to adapt the existing change detection methods for streaming data. However, many traditional change detection methods cannot be extended for streaming data because of the high computational complexity such as some kernel-based change detection methods, and density-based change detection methods. The second methodology is to develop new change detection methods for streaming data.

There are two common approaches to the problem of change detection in streaming data distributions: distance-based change detectors and predictive model-based change detectors. In the former, two windows are used to extract two data segments from the data stream. The change is quantified by using some dissimilarity measure. If the dissimilarity measure is greater than a given threshold then a change is detected. Similar to distance-based change detectors, two windows are used for detecting changes. Instead of comparing the dissimilarity measure between two windows with a given threshold, a change is detected by using the prediction error of the model built from the current window and the predictive model constructed from the reference window.

3. DISTRIBUTED CHANGE DETECTION IN STREAMING DATA

Knowledge discovery from massive amount of streaming data can be achieved only when we could develop the change detection frameworks that monitor streaming data created by multiple sources such as sensor networks, WWW [13]. The objectives of designing a distributed change detection scheme are maximizing the lifetime of the network, maximizing the detection capability, and minimizing the communication cost [58].

There are two approaches to the problem of change detection in streaming data that is created from multiple sources. In the centralized approach: all remote sites send raw data to the coordinator. The coordinator aggregates all the raw streaming data that is received from the remote sites. Detection of changes is performed on the aggregated streaming data. In most cases, communication consumes the largest amount of energy. The lifetime of sensors therefore drastically reduces when they communicate raw measurements to a centralized server for analysis. Centralized approaches suffer from the following problems: communication constraint, power consumption, robustness, and privacy. Distributed detection of changes in streaming data addresses the challenges that come from the problem of change detection, data stream processing, and the problem of distributed computing. The challenges coming from the distributed computing environment are as follows

- Distributed change detection in streaming data is a problem of distributed computing in nature. Therefore, a distributed framework for detecting changes should meet the properties of distributed computing such scalability, and fault tolerance. The scalability refers to the ability to extend the size of the network without significantly reducing the performance of the framework. As faults may occur due to the transmission error and the effects of noisy channels between local sensors and fusion center, a distributed change detection method should be able to tolerate these faults in order to assure the function of the system.

- Distributed change detection using the local approach is directly relevant to the problem of multiple hypotheses testing and data fusion because each local change detector needs to perform a hypothesis test to determine whether a change occurs. Therefore, besides considering the detection performance of local change detection algorithms including probability of detection and probability of false alarm at the node level, the detection performance of a distributed change detection method at the fusion center must be taken into account.

Distributed detection and data fusion have been widely studied for many decades. However, only recently, distributed detection in streaming data has received attention.

3.1 Distributed Detection: One-shot versus Continuous

Distributed detection of changes can be classified into two types of models as follows.

- One-shot distributed detection of changes: Figure 2 shows two models of one-shot distributed change detection. One-shot change detection method means a change detector detects and reacts to the detected change once a change is detected. One-shot distributed change detection have received great deal of attention for a long time. One-shot distributed change detection include two models: distributed detection with decision with decision fusion as shown in Figure 2(a); distributed detection without decision fusion as illustrated in Figure 2(b). What are the differences between one-shot detection and continuous detection.

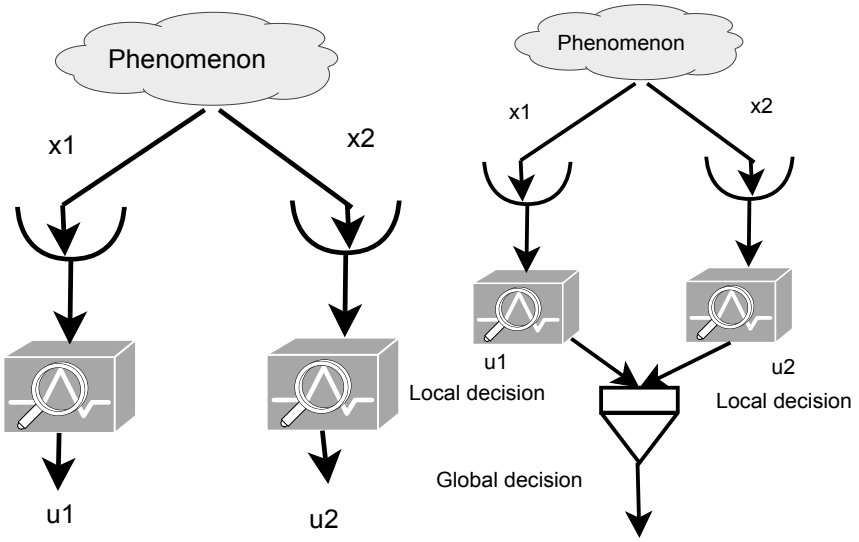
- Continuous distributed detection of changes: In this chapter, we propose two continuous distributed detection models as shown in Figure 3. An important distinction between continuous distributed detection of changes and one-shot distributed detection of changes is that the inputs to the one-shot distributed change detection are batches of data while the inputs to the continuous distributed detection of changes are the data streams in which data items continuously arrive.

Distributed detection model without fusion is a truly distributed detection model in which The decision-making process occurs at each sensor.

3.2 Locality in Distributed Computing

As one of the properties of distributed computational systems is locality [43], a distributed algorithm for detecting changes in streaming data should meet the locality. A local algorithm is defined as one whose resource consumption is independent of the system size. The scalability of distributed stream mining algorithms can be achieved by using the local change detection algorithms

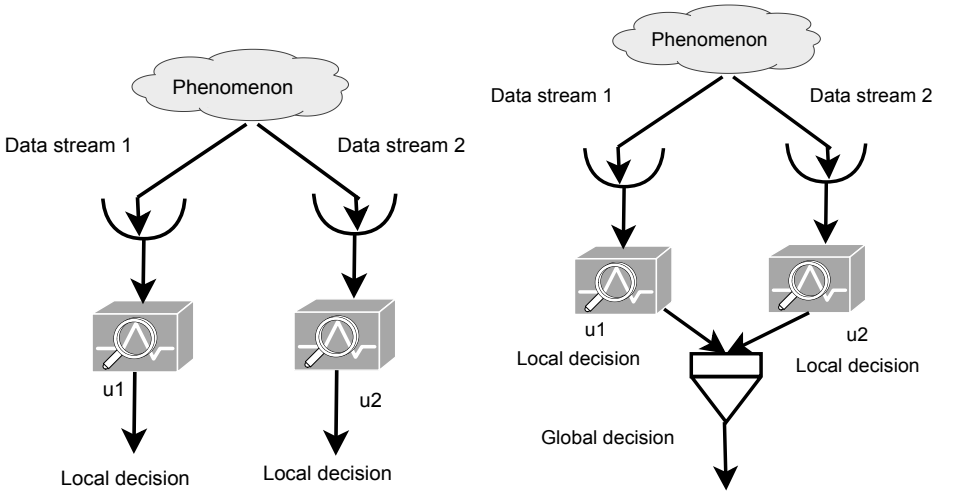
Local algorithms can fall into one of two categories [16]:Exact local algorithms are defined as ones that produce the same results as a centralized algorithm; Approximate local algorithms are algorithms that produce approximations of the results that centralized algorithms would produce. Two attractive properties of local algorithms are scalability and fault tolerance. A distributed



(a) Distributed detection without decision fusion

(b) Distributed detection with decision fusion

Figure 2: One-shot distributed change detection models



(a) Distributed continuous detection without decision fusion

(b) Distributed continuous detection with decision fusion

Figure 3: Continuous distributed change detection models

framework for mining streaming data should be robust to network partitions, and node failures.

The advantage of local approaches is the ability to preserve privacy [20]. A drawback of the local approach to the problem of distributed change detection is the synchronization problem. For example, the local change approach can meet the principle of localized algorithms in wireless sensor networks in which data processing is performed at node-level as much as possible in order to reduce the amount of information to be sent in the network.

3.3 Distributed Detection of Changes in Streaming Data

Over the last decades, the problem of decentralized detection has received much attention. There are two directions of research on decentralized detection. The first approach focuses on aggregating measurements from multiple sensors to test a single hypothesis. The second focuses on dealing with multiple dependent testing/estimation tasks from multiple sensors [51]. Distributed change detection usually involves a set of sensors that receive observations from the environment and then transmit those observations back to fusion center in order to reach the final consensus of detection. Decentralized detection and data fusion are therefore two closely related tasks that arise in the context of sensor networks [48; 47]. Two traditional approaches to the decentralized change detection are data fusion, and decision fusion. In data fusion, each node detects change and sends quantized version of its observation to a fusion center responsible for making decision on the detected changes, and further relaying information. In contrast, in decision fusion, each node performs local change detection by using some local change algorithm and updates its decision based on the received information and broadcasts again its new decision. This process repeats until consensus among the nodes are reached. Compared to data fusion, decision fusion can reduce the communication cost because sensors need only to transmit the local decisions represented by small data structures. Although there is great deal of work on distributed detection and data fusion, most of work focuses on the one-time change detection solutions. One-time query is defined as a query that needs to proceed data once in order to provide the answer [12]. Likewise, one-time change detection method is a change detection that requires to proceed data once in response to the change occurred. In real-world applications, we need the approaches capable of continuously monitoring the changes of the events occurring in the environment. Recently, work on continuous detection and monitoring of changes has been started receiving attention such as [49; 13; 50]. Das et al. [13] have presented a scalable distributed framework for detecting changes in astronomy data streams using local, asynchronous eigen monitoring algorithms. Palpanas et al. [49] proposed a distributed framework for outlier detection in real-time data streams. In their framework, each sensor estimates and maintains a model for its underlying distribution by using kernel density estimators. However, they did not show how to reach the global detection decision.

4. CONCLUDING REMARKS

We argued in this paper that variability, or simply change, is crucial in a world full of affecting factors that alter the behavior of the data, and consequently the underlying model. The ability to detect such changes in centralized as well as distributed system plays an important role in identifying validity of data models.

The paper presented the state-of-the-art in this area of paramount importance. Techniques, in some cases, are tightly coupled with application domains. However, most of the techniques reviewed in this paper are generic and could be adapted to different domains of applications.

With Big Data technologies reaching a mature stage, the future work in change detection is expected to exploit such scalable data processing tools in efficiently detect, localize and classify occurring changes. For example, distributed change detection models can make use of the *MapReduce* framework to accelerate their respective processes.

5. REFERENCES

[1] C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 575–586. ACM New York, NY, USA, 2003.

[2] C. Aggarwal. On change diagnosis in evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, pages 587–600, 2005.

[3] C. Aggarwal. A segment-based framework for modeling and mining data streams. *Knowledge and information systems*, 30(1):1–29, 2012.

[4] C. Aggarwal and P. Yu. A survey of synopsis construction in data streams. *Data streams: models and algorithms*, page 169, 2007.

[5] A. Bondu and M. Boullé. A supervised approach for change detection in data streams. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 519–526. IEEE, 2011.

[6] S. Boriah, V. Kumar, M. Steinbach, C. Potter, and S. Klooster. Land cover change detection: a case study. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 857–865. ACM, 2008.

[7] G. Cabanes and Y. Bennani. Change detection in data streams through unsupervised learning. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6. IEEE, 2012.

[8] J. Chang and W. Lee. estwin: adaptively monitoring the recent change of frequent itemsets over online data streams. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 536–539. ACM, 2003.

[9] K. Chen and L. Liu. HE-Tree: a framework for detecting changes in clustering structure for categorical data streams. *The VLDB Journal*, pages 1–20.

[10] T. CHEN, C. YUAN, A. SHEIKH, and C. NEUBAUER. Segment-based change detection method in multivariate data stream, Apr. 9 2009. WO Patent WO/2009/045,312.

[11] G. Cormode. The continuous distributed monitoring model. *SIGMOD Record*, 42(1):5, 2013.

[12] G. Cormode and M. Garofalakis. Efficient strategies for continuous distributed tracking tasks. *IEEE Data Engineering Bulletin*, 28(1):33–39, 2005.

[13] K. Das, K. Bhaduri, S. Arora, W. Griffin, K. Borne, C. Giannella, and H. Kargupta. Scalable Distributed Change Detection from Astronomy Data Streams using Local, Asynchronous Eigen Monitoring Algorithms. In *SIAM International Conference on Data Mining, Nevada*, 2009.

[14] T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, and K. Yi. Change (Detection) You Can Believe in: Finding Distributional Shifts in Data Streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, page 34. Springer, 2009.

[15] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *38th Symposium on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2005.

[16] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, pages 18–26, 2006.

[17] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[18] N. Dindar, P. M. Fischer, M. Soner, and N. Tatbul. Efficiently correlating complex events over live and archived data streams. In *ACM DEBS Conference*, 2011.

[19] G. Dong, J. Han, L. Lakshmanan, J. Pei, H. Wang, and P. Yu. Online mining of changes from data streams: Research problems and preliminary results. Citeseer.

[20] A. R. Ganguly, J. Gama, O. A. Omitaomu, M. M. Gaber, and R. R. Vatsavai. *Knowledge discovery from sensor data*, volume 7. CRC, 2008.

[21] V. Ganti, J. Gehrke, and R. Ramakrishnan. A framework for measuring changes in data characteristics. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 126–137. ACM, 1999.

[22] V. Ganti, J. Gehrke, R. Ramakrishnan, and W. Loh. A framework for measuring differences in data characteristics. *Journal of Computer and System Sciences*, 64(3):542–578, 2002.

[23] S. Geisler, C. Quix, and S. Schiffer. A data stream-based evaluation framework for traffic information systems. In *Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming*, pages 11–18. ACM, 2010.

[24] L. Golab, T. Johnson, J. S. Seidel, and V. Shkapenyuk. Stream warehousing with datadepot. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 847–854. ACM, 2009.

[25] A. J. Hey, S. Tansley, and K. M. Tolle. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research Redmond, WA, 2009.

[26] S. Hido, T. Idé, H. Kashima, H. Kubo, and H. Matsuzawa. Unsupervised change analysis using supervised learning. In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 148–159. Springer-Verlag, 2008.

[27] S. Ho and H. Wechsler. Detecting changes in unlabeled data streams using martingale. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1912–1917. Morgan Kaufmann Publishers Inc., 2007.

[28] W. Huang, E. Omiecinski, and L. Mark. Evolution in Data Streams. 2003.

[29] W. Huang, E. Omiecinski, L. Mark, and M. Nguyen. History guided low-cost change detection in streams. *Data Warehousing and Knowledge Discovery*, pages 75–86, 2009.

[30] E. Ikonomovska, J. Gama, R. Sebastião, and D. Gjorgjevik. Regression trees from data streams with drift detection. In *Discovery Science*, pages 121–135. Springer, 2009.

[31] M. Karnstedt, D. Klan, C. Pölitz, K.-U. Sattler, and C. Franke. Adaptive burst detection in a stream engine. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1511–1515. ACM, 2009.

[32] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of 2009 SIAM International Conference on Data Mining (SDM2009)*, pages 389–400, 2009.

[33] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, page 191. VLDB Endowment, 2004.

[34] A. Kim, C. Marzban, D. Percival, and W. Stuetzle. Using labeled data to evaluate change detectors in a multivariate streaming environment. *Signal Processing*, 89(12):2529–2536, 2009.

[35] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 234–247. ACM New York, NY, USA, 2003.

[36] L. Kuncheva. Change detection in streaming multivariate data using likelihood detectors. *Knowledge and Data Engineering, IEEE Transactions on*, (99):1–1, 2011.

[37] X. Liu, X. Wu, H. Wang, R. Zhang, J. Bailey, and K. Ramamohanarao. Mining distribution change in stock order streams. *Prof. of ICDE*, pages 105–108, 2010.

[38] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

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

[40] A. Maslov, M. Pechenizkiy, T. Kärkkäinen, and M. Tähtinen. Quantile index for gradual and abrupt change detection from cfb boiler sensor data in online settings. In *Proceedings of the Sixth International Workshop on Knowledge Discovery from Sensor Data*, pages 25–33. ACM, 2012.

[41] S. Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[42] S. Muthukrishnan, E. van den Berg, and Y. Wu. Sequential change detection on data streams. *ICDM Workshops*, 2007.

[43] M. Naor and L. Stockmeyer. What can be computed locally? pages 184–193, 1993.

[44] W. Ng and M. Dash. A change detector for mining frequent patterns over evolving data streams. In *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, pages 2407–2412. IEEE, 2008.

[45] W. Ng and M. Dash. A test paradigm for detecting changes in transactional data streams. In *Database Systems for Advanced Applications*, pages 204–219. Springer, 2008.

[46] D. Nikovski and A. Jain. Fast adaptive algorithms for abrupt change detection. *Machine learning*, 79(3):283–306, 2010.

[47] R. Niu and P. K. Varshney. Performance analysis of distributed detection in a random sensor field. *Signal Processing, IEEE Transactions on*, 56(1):339–349, 2008.

[48] R. Niu, P. K. Varshney, and Q. Cheng. Distributed detection in a large wireless sensor network. *Information Fusion*, 7(4):380–394, 2006.

[49] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *ACM SIGMOD Record*, 32(4):77–82, 2003.

[50] D.-S. Pham, S. Venkatesh, M. Lazarescu, and S. Budhaditya. Anomaly detection in large-scale data stream networks. *Data Mining and Knowledge Discovery*, 28(1):145–189, 2014.

[51] R. Rajagopal, X. Nguyen, S. C. Ergen, and P. Varaiya. Distributed online simultaneous fault detection for multiple sensors. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 133–144. IEEE, 2008.

[52] J. Roddick, L. Al-Jadir, L. Bertossi, M. Dumas, H. Gregersen, K. Hornsby, J. Lufter, F. Mandreoli, T. Mannisto, E. Mayol, et al. Evolution and change in data management:issues and directions. *ACM Sigmod Record*, 29(1):21–25, 2000.

[53] G. Ross, D. Tasoulis, and N. Adams. Online annotation and prediction for regime switching data streams. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1501–1505. ACM, 2009.

[54] A. Singh. Review Article Digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*, 10(6):989–1003, 1989.

[55] X. Song, M. Wu, C. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 667–676. ACM, 2007.

[56] D.-H. Tran and K.-U. Sattler. On detection of changes in sensor data streams. In *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*, pages 50–57. ACM, 2011.

[57] M. van Leeuwen and A. Siebes. Streamkrimp: Detecting change in data streams. *Machine Learning and Knowledge Discovery in Databases*, pages 672–687, 2008.

[58] V. Veeravalli and P. Varshney. Distributed inference in wireless sensor networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958):100–117, 2012.

[59] A. Zhou, F. Cao, W. Qian, and C. Jin. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, 15(2):181–214, 2008.

Contextual Crowd Intelligence

Beng Chin Ooi[†], Kian-Lee Tan[†], Quoc Trung Tran[†], James W. L. Yip[§], Gang Chen[#], Zheng Jye Ling[§], Thi Nguyen[†], Anthony K. H. Tung[†], Meihui Zhang[†]

[†]National University of Singapore [§]National University Health System [#]Zhejiang University
[†] {ooibc, tankl, tqtrung, thi, atung, zmeihui}@comp.nus.edu.sg
[§] {james_yip, zheng_jye_ling}@nuhs.edu.sg [#]cg@zju.edu.cn

ABSTRACT

Most data analytics applications are industry/domain specific, e.g., predicting patients at high risk of being admitted to intensive care unit in the healthcare sector or predicting malicious SMSs in the telecommunication sector. Existing solutions are based on "best practices", i.e., the systems' decisions are *knowledge-driven* and/or *data-driven*. However, there are rules and exceptional cases that can only be precisely formulated and identified by subject-matter experts (SMEs) who have accumulated many years of experience. This paper envisions a more intelligent database management system (DBMS) that captures such knowledge to effectively address the industry/domain specific applications. At the core, the system is a hybrid human-machine database engine where the machine interacts with the SMEs as part of a feedback loop to gather, infer, ascertain and enhance the database knowledge and processing. We discuss the challenges towards building such a system through examples in healthcare predictive analysis – a popular area for big data analytics.

1. INTRODUCTION

Most data analytics applications are industry or domain specific. For example, many prediction tasks in healthcare require prior medical knowledge, such as, identifying patients at high risk of being admitted to the intensive care unit, or predicting the probability of the patients being readmitted into the hospital within 30 days after discharge. Another example from the telecommunication sector is the identification of malicious SMSs requiring inputs from security experts. Building competent tools to effectively address these problems are important, as industrial organizations face increasing pressures to improve outcomes while reducing costs [3].

Existing solutions to industry or domain specific tasks are based on "best practices". These solutions are *knowledge-driven* (i.e., utilizing general guidelines such existing clinical guidelines or literature from medical journals) and/or *data-driven* (i.e., deriving rules from observational data) [31]. Let us consider the task of identifying the risk factors related to heart failure. The knowledge-driven solution uses risk factors identified from existing clinical knowledge or literature, such as, age, hypertension and diabetes status. However, it may miss out other unknown risk factors specific to the population of interest. The reason is that the guidelines are generic and based on existing knowledge, which results in models that may not adequately represent the underlying complex disease processes in the population with a comprehensive list of risk factors [31]. The data-driven solution employs machine learning algorithms to derive risk factors solely from observational data. An alternative approach combines the knowledge-driven and data-driven approaches in the data analytics applications [31]. However, there are exceptional situations where it is not easy to capture or formalize, and where neither general guidelines are available nor rules can be derived from data (e.g., in rare conditions). Instead, it is only through many years of experience can subject-matter experts (SMEs) formulate and identify these situations. The challenge then is to be able to capture and utilize such knowledge to effectively support industry/domain specific applications, e.g., improving the accuracy of the prediction tasks.

This paper proposes building the next generation of *intelligent database management systems (DBMSs) that exploit contextual crowd intelligence*. The crowd intelligence here refers to the knowledge and experience of subject-matter experts (SMEs). Although such knowledge is an important component in transforming data into information, it is currently not captured by a structured system. The participants in an intelligent crowd are domain experts rather than "unknown" lay-persons in existing systems that use crowdsourcing as part of database query processing (e.g., CrowdDB [13], Deco [24], Qurk [23], CDAS [12; 22]) and information extraction or knowledge acquisition (e.g., HIGGINS [21] and CASTLE [28]). For applications where data confidentiality and privacy are important (e.g., healthcare analytics), the intelligent crowd may consist of only experts from within the organization, since the tasks cannot be outsourced to external parties. Given that the crowd is known apriori, there is an assurance of user accountability, which translates to an assurance in the quality of the answers. A recent system, called Data Tamer [30], also proposed to leverage on expert crowdsourcing system to enhance machine computation but in the context of data curation. Our proposition differs from Data Tamer in several aspects. First, the target applications of our work (i.e., data analytics) are different from those in Data Tamer (i.e., data curation). Thus, each system needs to address a unique, different set of challenges. Second, the domain experts in our context are also *users/reviewers* of the system. Thus, the experts are likely to take ownership and hence are motivated to improve the accuracy of the analytics and the usability of the applications. This would reduce the need to *localize/customize* the system since the experts/users are continuously interacting with the system; these experts define the "best practices" for the system. For example, doctors in a particular department may use a different convention or notation from another department, e.g., when doctors write "PID" in the orthopedic department, the acronym refers to the "Prolapsed Intervertebral Disc" only and not the "Pelvic Inflammatory Disease". Clearly, such knowledge can only

be provided by internal domain experts. In contrast, experts in Data Tamer are not the users of the system and hence there is a need to customize/localize the system for different use-cases.

In order to entrench the crowd intelligence into the DBMS, the system needs to keep SMEs as part of the feedback loop. The system can then further utilize feedback provided from the SMEs to infer, ascertain and enhance its processing, thus continuously improving the effectiveness of the system. For example, when predicting the risk of unplanned patient readmissions, the system asks the doctors to label patients who the system has low confidence in predicting their readmissions, and the rules/hypotheses that the doctors used to do the labeling. One example of such an expert rule is that an elderly patient who lives alone and have had several severe diseases is likely to be readmitted into the hospital frequently. The system would then verify or adjust these rules/hypotheses and revert back to the doctors with evidence to support or reject their rules/hypotheses. Such interactions are beneficial to both the system and the doctors. Eventually, the application system evolves over time. SMEs become part of this evolving process by sharing their domain knowledge and rich experience, thereby contributing to the improvement and development of the system. Hence, the experts are more willing and comfortable to use the system to alleviate the burden of their duties.

This work is part of our CIIDAA project on building large scale, Comprehensive IT Infrastructure for Data-intensive Applications and Analysis [2]. Our collaborators are clinicians in the National University Health System (NUHS) [5]. The project aims to harness the power of cloud computing to solve big data problems in the real world, with healthcare predictive analytics being a popular area for big data analytics [26].

Organization. The remainder of this paper is organized as follows. Section 2 presents motivating examples in healthcare predictive analytics. Section 3 discusses the architecture of an intelligent DBMS that aims to embed contextual crowd intelligence. Section 4 elaborates on research problems that we need to address in order to build an intelligent DBMS. Section 5 presents our preliminary results on the problem of predicting the risk of unplanned patient readmissions. Section 6 presents the related work. Finally, Section 7 concludes our work.

2. MOTIVATING EXAMPLES

Let us consider a hospital that has an integrated view of the medical care records of patients as shown in Table 1. The table contains two types of information:

- Structured information, including the case identifier, patient's name, age, gender, race, the number of days that the patient stayed at the hospital during a particular visit (*LengthOfStay*), and the number of days before the patient was readmitted into the hospital after discharge (*Readmission*) ; and
- Unstructured information, i.e., free-text from a doctor's note that contains additional and useful information of a patient healthcare profile such as his past medical history, social factors, previous medications, complaints of patients based on a doctor's investigations, major lab results, issues and progress, etc.

The tuples in this table are extracted from real cases of patients admitted to the National University Hospital (NUH) in Singapore.

Healthcare professionals often have queries relating to predicting the severity of patients' condition, such as, identifying patients at high risk of being admitted to intensive care unit, or predicting the probability of the patients being readmitted into the hospital soon after discharge. There are also queries that monitor real-time data of patients in critical conditions for unusual conditions, such as, whether patients are at high risk of collapsing. With correct predictions, doctors can intervene early to alleviate the deterioration of patient's health outcome. This can potentially reduce the burden of limited healthcare resources in the primary and acute care facilities. For instance, if a patient is at high-risk for unplanned post discharge readmission, he can potentially benefit from close followed-up after discharge, e.g., the hospital sends a case manager or nurse to examine him once every three days. In addition, important queries related to public health surveillance can be answered in a timely fashion. For example, it is critical to provide real-time, early information to alert decision-makers of emerging threats that need to be addressed in a particular population. The ultimate goal of these predictive queries is to predict, pre-empt and prevent for better healthcare outcome.

3. AN INTELLIGENT DBMS FOR BIG DATA ANALYTICS

In this section, we discuss the challenges of addressing big data analytics and present an overview of a hybrid human-machine system for these tasks.

3.1 Challenges of Big Data Analytics

Essentially, many tasks of big data analytics can be viewed as conventional data mining problems, such as, classifying patients into different class labels (high or low risk of being admitted to intensive care units). There are, however, three important aspects that differentiate big data analytics from traditional machine learning problems.

- First, many valuable features for the analytics tasks are stored in unstructured data, for example, doctor's notes [25]. We cannot simply treat these notes as traditional "bag-of-words" documents. Instead, we need powerful tools to extract from these documents the right entities (such as, diseases, medications, laboratory tests) and domain-specific relationships (such as, the relationship between a disease and a laboratory test). The text in unstructured data has to be contextualized to each organization's practice, e.g., doctors in a particular department may use a different convention or notation from another department.
- Second, there is usually a lack of training samples with well-defined class labels. For instance, when predicting the risk of committing suicide for each patient, the total number patients known to have committed suicide (i.e., class 1) is very small. However, it does not mean that all the remaining patients did not commit suicide (i.e., class 0). Hence we need to infer the correct class labels for these patients. This problem also occurs in other domains such as home security and banking. For example, one important task that many national security agencies need to perform is identifying persons or groups of people who will likely commit a crime [4]. In this setting, the agency maintains a very small set of people who have committed crime. However, we cannot simply assume that the remaining people are not likely to commit crime. As before, we need to infer the correct class labels for these people. Another example is in telecommunication, where a service provider wants to predict whether an SMS is malicious. In this case, we do not

CaseID	Name	Age	Gender	Race	LengthOfStay	Readmission	Doctor's note
Case 1	Patient 1	71	Female	Chinese	5	20	PMH: 1 IHD - on GTN 0.5mg prn 2 DM - on Metformin 750mg - HbA1c 7.5% 09/12 3 HL Stays with son ⋯
Case 2	Patient 2	60	Male	Malaysian	10	20	Social issues: Single, no child Used to live with friend in a shophouse Now at sheltered home since Sept 2011. No next-of-skin or visitor. ⋯

Table 1: Medical care table

have any predefined class labels and might need to ask security experts to provide the class labels for some sample cases.

- Lastly, data in different domains (e.g., healthcare, telecommunication, home security) is expected to grow dramatically in the years ahead [26]. For instance, patients in intensive care units are constantly being monitored, and their historical records have to be retained. This can easily result in hundreds of millions of (historical) records of patients. As another example, during a mass casualty disaster (e.g., SARS, H5N1), there is an overwhelming number of patients who have to be monitored and tracked, and information about each patient is huge by itself. Furthermore, streaming data arrive continuously, e.g., new data from the real-time data feed are constantly being inserted. Hence, the system in healthcare setting must provide the real-time predictions, e.g., predicting the survival of patients in the next 6 hours.

The three above mentioned aspects call for a new generation of intelligent DBMSs that can provide effective solutions for big data analytics. Our proposition of exploiting contextual crowd intelligence is, we believe, a big step towards this goal.

3.2 Contextual Data Management

The central theme of crowd intelligence is to get domain experts engaged as both the participants to fine tune the system and the end-users of the system. Figure 1 presents an intelligent system that exploits contextual crowd intelligence for big data analytics.

The system first builds a knowledge base that will be subsequently used for the analytics tasks based on historical data, domain knowledge from SMEs (e.g., doctors), and other sources such as general clinical guidelines. Each source contributes to build some "weak classifiers". The system needs to combine these classifiers to derive a final classifier that achieves a high level of accuracy for prediction purposes. The system also needs to go through several iterations of interaction with the experts to refine, for example, the final classifier. As such, the experts participate in the entire process in fine tuning the system and decide on the "best practices". When real-time data or feed arrives, the system performs the prediction on-the-fly and alerts the experts immediately. Hence, the experts become the end-users of the system.

We have developed the epiC system [1; 10; 19] to support large scale data processing, and are extending it to support healthcare analytics. Figure 2 shows the software stack of epiC. At the bottom, the storage layer supports different storage systems (e.g., Hadoop Distributed File System (HDFS) and a key-value storage system, ES^2 [8]) for both unstructured and structured data. The next layer (which is the security layer) enables users to protect data privacy by encryption. The third layer (which is the distributed processing layer) provides a distributed processing infrastructure called E^3 [9] that supports different parallel processing logics such as MapReduce [11], Directed Acyclic Graph (DAG) and SQL. The top layer (which is the analytics layer) exploits the contextual crowd intelligence for big data analytics. The details of this layer are shown in Figure 1. In Figure 2, KB is the knowledge base and iCrowd is the component that interacts with the domain experts. Different components of the analytics layer (e.g., scalable machine learning algorithms) can process their data with the most appropriate data processing model and their computations will be automatically executed in parallel by the lower layers.

In the remaining of this paper, we focus only on the analytics layer. For more details of the other layers of the epiC system, please refer to [1; 10; 19].

4. RESEARCH PROBLEMS

In this section, we elaborate on the research problems that we need to address in order to build an intelligent system for big data analytics.

4.1 Asking Experts The Right Questions

Given a large volume of data and a limited amount of time that domain experts can participate in building the systems, we need to ask the experts the right questions. In the context of healthcare analytics, we plan to ask the following domain knowledge from doctors.

- **Labelings.** The system asks doctors to label tuples that the system has low confidence in performing the prediction task. There are two important issues here. First, doctors have different levels of confidence when answering different questions, i.e., doctors are reluctant to assess patient profiles that they do not have specialties. Second, since there is so much information about patients, selecting the relevant feature of each patient to present to the doctors in order not to overwhelm them is also a major issue.

  In essence, what we need is a diverse set of labeled patients that covers the whole data space as much as possible. One possible solution is to group similar patient profiles together and show these groups to doctors. The purpose is to let the doctors select the groups of patients that they are comfortable in providing the labels. In addition, for each

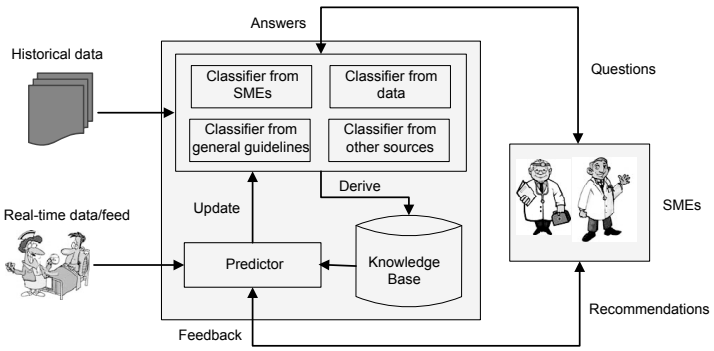


Figure 1: Contextual crowd intelligence for big data analytics.

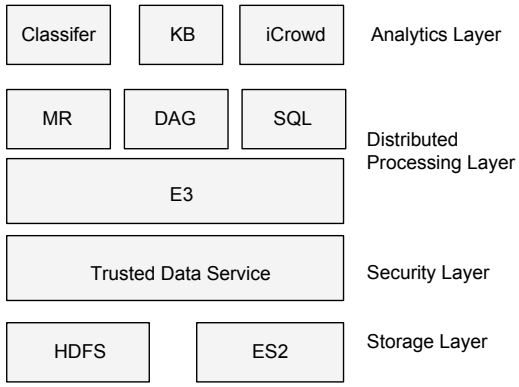


Figure 2: The software stack of epiC for big data analytics.

group, we present only the features which the patients in the group have similar values. In this way, we can avoid overwhelming the doctors with information. Note that, in some cases, we need to perform hierarchical clustering to reduce the number of patients shown to the doctors each time. Selecting the right clustering algorithms and developing effective visualization tools to present patient's profiles are important here.

- **Rules/Hypotheses.** The system collects expert rules/hypotheses that the doctors used to do the labeling. For example, to predict the risk of unplanned patient readmissions, the doctors suggested a hypothesis that social factors and the status of the diseases are important risk indicators for readmission. The system would then verify or adjust these hypotheses and revert back to the doctors with evidence to support or reject their hypotheses. Such interactions are beneficial to both the system and the doctors.

- **Inferred implicit knowledge**. The system can also infer implicit and valuable knowledge based on the answers/reactions of the domain experts. For instance, if the doctors label two patients who belong to a given cluster differently, then the system can adjust the distance function used to compute the similarity between two patients, and thus infer which features are more important. Such knowledge is implicit as the doctors themselves may not be aware of.

We can also ask the same kind of questions for the analytics tasks in other domains. For instance, to predict malicious SMSs, we need to select a small set of messages (by utilizing some clustering algorithms) and ask the experts to provide labels for these samples. We also collect rules and heuristics that the experts utilize to label the SMSs.

4.2 Extracting Domain Entities From Unstructured Data

Feature selection is very important for any machine learning task and can greatly affect the algorithm's quality. Processing doctor's notes for extracting important features is an inevitably important step for healthcare analytics problems. There are several state-of-the-art Natural Language Processing (NLP) engines for processing clinical documents, such as, MedLEE [14] and cTAKES [27]. These engines process clinical notes, identifying types of clinical entities (e.g., medications, diseases, procedures, lab tests) from various medical dictionaries (a.k.a. knowledge base), such as, the Unified Medical Language System (UMLS) [6]. We now discuss several problems raised due to the nature of the unstructured data and the incompleteness of the knowledge base, and subsequently discuss a hybrid human-machine approach to solve these problems. The discussion uses the following running example. We run cTAKES on the doctor's note of patient 1 (in Table 1), and obtain the following clinical entities: (1) *diseases*: IHD (Ischemic Heart Disease) and DM; (2) *medications*: GTN and Metformin; and (3) *laboratory test*: HbA1c.

Ambiguous mentions. In many cases, a mention in the free text may refer to different domain entities. For instance, in the running example, "DM" refers to two different diseases "Dystrophy Myotonic" and "Diabetes Mellitus". We note that this problem is not uncommon as doctors tend to use abbreviations in their notes. For example, "CCF" refers to either "Congestive heart failure" or "Carotid-Cavernous Fistula" diseases; "PID" refers to either "Prolapsed Intervertebral Disc" or "Pelvic Inflammatory Disease". There are also cases where only human but not the machine can understand the meaning of some mentions in the text. For example, assuming that we are extracting the social factor of patients in Table 1. It is rather easy to extract the social factor for patient 1, since the text contains the phraze "stays with son". However, it is challenging, if not possible, for the machine to extract the social factor for patient 2. The reason is that the paragraph contains several different keywords relating to the social factor such as "single", "no child", "live with friend", "sheltered home", "next-of-kin".

Incomplete knowledge base. The knowledge base is incomplete for the following reasons. First, the terms used in the doctor's notes could be specific within a country or a particular hospital, whereas the existing knowledge bases may only cover the universal ones. Thus, these terms do not exist in the dictionary. One example is the term "HL" in our running example, which refers to the "Hyperlipidemia" disease but is not captured in UMLS. Second, the relationships between entities covered in existing medical knowledge bases (like ULMS) are far from complete. In the running example, the fact that the medication Metformin is used to treat Diabetes Mellitus (DM) is also missing in UMLS. The relationships that exist between domain entities can be used to derive implicit and useful information. For instance, from the laboratory result of the lab test HbA1c, we can infer whether the DM condition is well-controlled (i.e., the relationship between a disease and a lab test).

A hybrid human-machine approach. To infer the correct entities from unstructured data, a hybrid human-machine solution should be employed. The system can leverage the information from the knowledge base (e.g., UMLS) together with the implicit information (signals) inherent in the unstructured data (e.g., doctor's notes) to improve the accuracy of its inference process and enhance the knowledge base as well. The system will pose questions to the healthcare professionals for verification. Based on the answers from the experts, the system adjusts its inference results. The inference process gets more accurate and complete as the system runs more iterations. Meanwhile, the knowledge base becomes more comprehensive and customized to each organization's practice. More specifically, in our running example:

- Since "DM" is attached with the laboratory test "HbA1c" in the paragraph, the machine conjectures that "DM" would refer to the "Diabetes Mellitus" disease only. The reason is that HbA1c is a laboratory test that monitors the control of diabetes and HbA1c does not have any relationship with the other disease related to "DM" (i.e., "Dystrophy Myotonic").
- To correctly infer the disease "Hyperlipidemia" for "HL", the machine infers a pattern of "*num d*" where *num* is a fraction annotation and *d* is a disease. ("1 IHD" and "2 DM" are two examples.) The machine then infers that "HL" may refer to a disease since the phrase "3 HL" follows the pattern. The machine then poses a question to a doctor: which disease "HL" represents for? In this case, the doctor confirms that "HL" represents for the "Hyperlipidemia" disease. Based on the answer, the machine adds the mapping between the mention "HL" and the disease "Hyperlipidemia" to the knowledge base. Hence, the knowledge base becomes more comprehensive and customized to NUH's practice.
- To identify the missing relationship between the medication Metformin and the disease DM, the machine infers a pattern of "*d* on *med*", where *d* is a disease, *med* is a medication and *med* is used to treat *d*. ("IHD on GTN" is an example.) The machine conjectures that there should have a relationship between DM and Metformin, since the phraze "DM on Metformin" follows the pattern. The machine then verifies this inference with the doctors. The doctors confirm that they typically write the medications that are used to treat a disease right next to the disease, and connect these relationships by the preposition "on". Clearly, such rule is very useful – the machine will then infer other missing relationships using this expert rule with fewer questions being posed to the doctors.
- To derive the social factor for patient 2, the machine can first attempt to derive the information using a simple strategy such as analyzing the NLP structure of sentences containing patterns like "stay with", "live with". For complicated cases when the machine cannot find out the information, we need to tap on the knowledge of the experts.

4.3 Combining Multiple Weak Classifiers

We can obtain different classifiers from multiple sources such as classifiers built based on the observational data, rules used by the doctors and general clinical guidelines. Each source of knowledge can be considered as a "weak classifier" and the task is to combine these classifiers to derive a final classifier that achieves a very high accuracy in prediction.

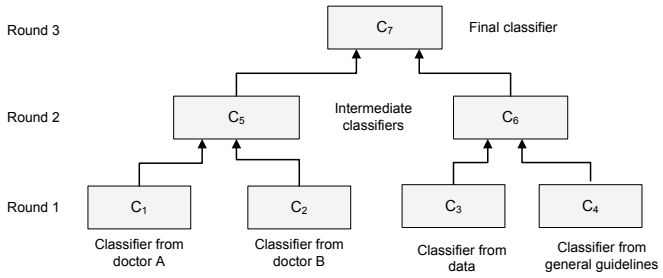


Figure 3: An example of several rounds of learning for healthcare predictive analytics

There are many ways to achieve the goal. Figure 3 shows an example of a process consisting of three rounds of learning for the task of predicting the severity of patients. In the first round, the system computes four classifiers: C_1 and C_2 are the classifiers derived from rules provided by SMEs (i.e., doctors); C_3 is the classifier derived from historical data; and C_4 is the classifier derived from clinical guidelines. It is essential to resolve disagreeing opinions from various sources. There are several ways to combine different classifiers, such as, using majority-voting for the outputs of different rules/classifiers or combining features being used in different input classifiers.

It is likely that all the classifiers built after the first round do not agree with each other for the prediction tasks. Thus, in this example, the system performs two additional rounds of learning to improve the accuracy of the classifier. It is also possible that there is no way to reconcile the classifiers, i.e., there will be multiple different classifiers. In such situations, it may be necessary to "rank" the results of the different classifiers, and pick the answer that is ranked highest. How to do this is an open question.

4.4 Scalable Processing

Big data analytics is characterized by the so-called 3V features: Volume - a huge amount of data, Velocity - a high data ingestion rate, and Variety - a mixed of structured, semi-structured and unstructured data. These requirements force us to rethink the whole software stack to address big data analytics efficiently and effectively, ranging from the storage layer that should manipulate both structured and unstructured data to application layer that should support scalable machine learning algorithms. To illustrate the points, let us reconsider the problem of predicting the malicious SMSs. The collection of SMSs is huge, e.g., in the order of hundreds of tera-bytes. As discussed in Section 4.1, we need to pick a set of SMSs for domain experts to label. Conventional clustering algorithms may not work well here as we need to handle such a large amount of data. The problem is even more challenging in our context, as we need to frequently get the domain experts involved in building the system. The delay from human beings' reaction may be a large factor affecting the low latency of the system.

The scalability of the problems is also in terms of high-dimensional data space. Our data set inherently contains a large number of features. For instance, there are different information about patients such as thousands of different diseases and lab tests. One solution to reduce the dimensions is to group these attributes *semantically*, e.g., grouping together different diseases that share a same "root". For instance, the Hypertension disease, Hypotension disease and Ischaemic Heart disease can be grouped together under the category of Cardiovascular disease.

Clearly, to perform such tasks, we need to consult the domain experts as different hospitals/doctors may have different opinions/reasoning in performing this task. This is, again, an example of getting the domain experts involved in building the systems.

4.5 Engaging Expert Users

As the system needs to interact with SMEs frequently, it is important to engage the experts along the process of building and using the system. The system should provide several functionalities for this purpose:

- A user-friendly interface for the experts to provide their inputs such as rules, hypothesis, labels, etc.
- The system should provide not only the final outcome (e.g., whether the patient is at high/low risk of being sent to ICU) but also the reasons that drive its decision. Therefore, keeping track of the provenance of the knowledge is important. For instance, when the system makes a decision that differs from experts' opinions, the system should be able to trace back whether the mismatch is mainly due to the use of some general guidelines, or due to other experts' opinions.
- Presenting feedback to the experts. For instance, the system can explain how well an expert performs compared to other colleagues. As another example, the system can reveal comments and annotations by other experts to see whether an expert would change her decision. It is also interesting to present new patterns of knowledge that an expert may lack and potentially educate her.

5. PRELIMINARY RESULTS

We are studying the problem of predicting the probability of patients being readmitted into the hospital within 30 days after discharge. We refer to the task as *readmission prediction* for short. We use the clinical data drawn from the National University Hospital's Computerized Clinical Data Repository (CCDR) and focus only on the *elderly patients* (i.e., patients with age older than 60) admitted to the hospital in 2012. The table used for the prediction task is the medical care table[1] that has similar schema as the one presented in Table 1. There are in total 29049 elderly patients admitted to NUH in 2012, where 5658 patients readmitted within 30 days, i.e., the proportion of patients who were readmitted (i.e. class label 1) is 0.188.

5.1 Interacting with Domain Experts

We have been getting the doctors involved in the following tasks.

Hypothesis/Rules. Our clinician collaborators have suggested a hypothesis that the following features (indicators) might be important for the readmission prediction:

- Social-economic factors, e.g., who are the care-givers and the patient's economic status.
- Lab findings. We should extract the lab findings that the doctors mentioned in their notes instead of using the labs recorded in the structured data in CCDR. The reason is that patients typically have hundreds of lab tests but only a small number of them is important and is captured in the doctor's notes. As a result, selecting lab findings mentioned by doctors naturally reduces the dimensions of the data set.
- Comorbidity influence, i.e., we should take into account the past medical history of the patient together with the disease status (whether the disease has been well-controlled).

[1]To derive the medical care table, we joined information from various relations in CCDR, including: Discharge Summary, Patient Demographics, Visit and Encounter, Lab Results and Emergency Department.

	# actual class 1	# actual class 0
#predicted class 1	1071	1321
#predicted class 0	4587	22070

(a) Using only structured features

	# actual class 1	# actual class 0
#predicted class 1	2679	4250
#predicted class 0	2979	19141

(b) Using both structured and derived features

Table 2: The accuracy of our classifier.

Participants in a crowd-sourcing system. We adopted a hybrid human-machine approach to extract the social factors and lab findings from doctor's free-text notes.

To extract the social factors, we use an NLP technique to analyze sentences containing phrases related to the social factor such as "live (with)", "stay (with)", "main care-giver" to pinpoint some keywords such as "daughter", "family", "spouse", etc. The system then asks the doctors to handpick a set of predefined categories of social factors. For instance, living with family and taking care by professional helpers (e.g., maid, domestic helpers) are in a same group. As another example, living alone and living in a community nursing home are in a same group. The system also performs a postprocessing step to pull out cases that can be assigned more than one category of social factors. The system then asks the doctors to label these cases manually. (There are about 200 cases that need to be manually labeled.)

To extract the lab findings, the system first uses a simple pattern matching technique to extract all possible lab tests mentioned in the note. For instance, if the note contains a pattern of the form "*word num*" where *word* is some word and *num* is a number, then *word* is a candidate lab test. A *word* is a correct lab test if it exists in the medical dictionary with the category of lab tests. For the "false" lab tests that are currently not present in the dictionary and appear frequently in the notes, the system asks the doctors to verify them. As a result, there are some actual lab tests that are missing in the dictionary such as "TW", which is a local convention used inside NUH.

Extracting medical concepts. We run the cTAKES NLP engine over the UMLS dictionary to extract the past medical history of a patient. We are in the process of developing algorithms to improve the accuracy of extraction (to resolve problems mentioned in Section 4.2). Thus, we use the number of diseases that the patient has as an indicator instead of the actual diseases.

5.2 Results

After interacting with the doctors to extract relevant features, we obtained two sets of features for the prediction task:

- *Structured features*: patients' demographics (age, gender, race), the number of days that the patient stayed at the hospital, the number of previous hospitalizations, and the

number of prior emergency visits in the last six month before admission.

- *Derived features* from free-texts (We refer to these features as derived features for short): social factors, lab findings, and past medical history (i.e., diseases).

We used WEKA [15] to run a 10-fold cross-validation and the Bayesian Network classifier to construct a readmission classifier[2]. Table 2 reports the accuracy of the prediction across all the 10 validation data. If only structured features are used to build the classifier (Table 2(a)), the resulting classifier can correctly predict 1071 cases that are readmitted (within 30 days). The precision and recall in this case are 0.448 and 0.189, respectively. Meanwhile, if both structured and derived features are used to build the classifier (Table 2(b)), the resulting classifier can correctly predict 2679 cases that are readmitted. The precision and recall are 0.387 and 0.473 respectively. Clearly, the recall has been improved significantly with the usage of the derived features from the free-text doctor's notes. The result is also very promising when we compared it to the result handled manually by domain experts such as physicians, case managers, and nurses [7]. The recall reported in [7] is in the range [0.149, 0.306]. The conclusion in [7] is that care-providers were not able to accurately predict which patients were at highest risk of readmission. However, we believe that a hybrid machine-human solution would greatly alleviate the problem.

We would like to emphasize that there are many rooms to further improve the accuracy of the prediction such as enhancing the feature extraction process, employing additional features, such as, disease status, specific diagnoses, medications, and using special classifiers for highly-imbalanced data set.

6. RELATED WORK

Related works to our proposition can be broadly classified into the following three categories.

Existing solutions for industry/domain specific applications. Existing solutions are currently built based on "best practices". One direction is knowledge-driven approach that is based on general guidelines such as clinical guidelines, e.g., IBM Watson [3]. Another direction is data-driven approach that is based on "rules" extracted from the observational data, e.g., [16; 18; 20]. Recently, IBM proposes to combine the strengths of the two directions [31]. However, these solutions have not explored the exceptionally complicated rules/patterns that can only be provided by *internal domain experts* with years of working experience. Our research aims to fill this gap: we seek to engage the experts as users of the system, and tap on their expertise to enhance the database knowledge and processing. There are several benefits of employing internal domain experts. First, we do not need to customize/localize the system for different use-cases; they themselves define the "best practices" for the system. Second, in terms of the data used to build the knowledge base, our system mainly bases on observational data and knowledge provided by domain experts; whereas others (e.g., IBM Watson) need to process a much larger amount of inputs such as medical journals, white papers, medical policies and practices, information in the web, etc. Third, the system should become more "intelligent" over times when the expert users continuously enhance the system with their expert knowledge.

[2]We also used other classifiers such as decision tree, rule-based classifier, SVM, etc and observe that the Bayesian Network classifier provides the best result.

Crowdsourcing in database. There has been a lot of recent interest in the database community in using crowdsourcing as part of database query processing (e.g., CrowdDB [13], Deco [24], Qurk [23], CDAS [12; 22]). As discussed, the intelligent crowds in our context are domain experts (rather than lay-persons in the existing crowds) who are also users/reviewers of the system. Furthermore, exploiting intelligent crowd can be much more collaborative in nature. In typical crowdsourcing, the crowds are not aware of each other's answers. But in our context, we can actually go through several iterations and see whether the experts will change their decisions when they are provided with comments and annotations by other experts.

A recent system, called Data Tamer [30], also leveraged expert crowdsourcing system to enhance machine computation but in the context of data curation. As discussed in Section 1, the key difference between our proposition and Data Tamer lies in the fact that the domain experts in our context are also *users/reviewers* of the system. Thus, the experts are likely to take ownership and hence are motivated to improve the accuracy of the analytics and the usability of the applications. This would reduce the need to localize/customize the system. Also, each system needs to address a different set of challenges, since the targeted applications are different.

Active learning. In the active learning model, the data come unlabeled but the goal is to ultimately learn a classifier (e.g., [17; 29; 32]). The idea is to query the labels of just a few points that are especially informative in order to obtain an accurate classifier. The labels are obtained from highly-trained experts (e.g., doctors). The scope of our proposition is much more general than active learning in the following points. First, we would like to exploit as much domain knowledge from experts as possible, not restricting to only the class labels as in active learning. For instance, rules and hypotheses provided by experts with many years of experience must be exploited in several cases. Second, active learning focuses on getting a better classifier so the query points presented to the crowd are usually those data points that are at the boundary of the separating plane. However, these are also the data points that the experts are usually not very clear about. As such, we need to be able to identify additional information that should be provided for the experts to be able to make an informed decision. Lastly, we need to handle a large amount of data whereas existing solutions on active learning usually deal with small data set.

7. CONCLUSION

Each of us is a subject-matter expert (SME) of our profession, and we carry with us a vast amount of knowledge and insights not captured by a structured system. This might have explained the emergence of Knowledge Management systems. However, there are many rules and exceptional cases that can only be formulated by experts with many years of experience. Such rules, when properly coded, can help in facilitating contextual decision making. This paper envisions a more intelligent DBMS that captures such information or knowledge. At the core, the system is a hybrid human-machine database processing engine where the machine keeps the SMEs as part of the feedback loop to gather, infer, ascertain and enhance the database knowledge and processing. This paper discussed many open challenges that we need to tackle in order to build such a system.

8. ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation, Prime Minister's Office, Singapore under Grant No.

NRF-CRP8-2011-08. We thank Associate Professor Gerald C.H. Koh and Dr. Chuen Seng Tan (Saw Swee Hock School of Public Health, National University Health System) for sharing with us domain knowledge in healthcare.

9. REFERENCES

[1] http://www.comp.nus.edu.sg/∼epic.

[2] The comprehensive it infrastructure for data-intensive applications and analysis project. http://www.comp.nus.edu.sg/∼ciidaa/.

[3] Ibm big data for healthcare. http://www.ibm.com.

[4] The minority report: Chicago's new police computer predicts crimes, but is it racist? http://www.theverge.com/2014/2/19/5419854/the-minority-report-this-computer-predicts-crime-but-is-it-racist.

[5] National university health system. http://www.nuhs.edu.sg/.

[6] Unified medical language system. http://www.nlm.nih.gov/research/umls/.

[7] N. Allaudeen, J. L. Schnipper, E. J. Orav, R. M. Wachter, and A. R. Vidyarthi. Inability of providers to predict unplanned readmissions. *J Gen Intern Med*, 26(7):771776.

[8] Y. Cao, C. Chen, F. Guo, D. Jiang, Y. Lin, B. C. Ooi, H. T. Vo, S. Wu, and Q. Xu. Es2: A cloud data storage system for supporting both oltp and olap. In *ICDE*, pages 291–302, 2011.

[9] G. Chen, K. Chen, D. Jiang, B. C. Ooi, L. Shi, H. T. Vo, and S. Wu. E3: an elastic execution engine for scalable data processing. *JIP*, 20(1):65–76, 2012.

[10] G. Chen, H. Jagadish, D. Jiang, D. Maier, B. Ooi, K. Tan, and W. Tan. Federation in cloud data management: Challenges and opportunities. *TDKE*, 2014.

[11] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1), Jan. 2008.

[12] J. Fan, M. Lu, B. C. Ooi, W.-C. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *ICDE*, pages 976–987, 2014.

[13] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD Conference*, pages 61–72, 2011.

[14] C. Friedman, P. O. Alderson, J. H. Austin, J. J. Cimino, and S. B. Johnson. A general natural-language text processor for clinical radiology. *JAMIA*, 1(2):161–174, 1994.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[16] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[17] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, pages 417–424, 2006.

[18] A. Hosseinzadeh, M. T. Izadi, A. Verma, D. Precup, and D. L. Buckeridge. Assessing the predictability of hospital readmission using machine learning. In *IAAI*, 2013.

[19] D. Jiang, G. Chen, B. C. Ooi, K.-L. Tan, and S. Wu. epic: an extensible and scalable system for processing big data. In *PVLDB*, 2014.

[20] P. S. Keenan, S.-L. T. Normand, Z. Lin, E. E. Drye, K. R. Bhat, J. S. Ross, J. D. Schuur, B. D. Stauffer, S. M. Bernheim, A. J. Epstein, Y. Wang, J. Herrin, J. Chen, J. J. Federer, J. A. Mattera, Y. Wang, and H. M. Krumholz. An administrative claims measure suitable for profiling hospital performance on the basis of 30-day all-cause readmission rates among patients with heart failure. *Circ Cardiovasc Qual Outcomes*, 1(1):29–37, 2008.

[21] K. S. Kumar, P. Triantafillou, and G. Weikum. Human computing games for knowledge acquisition. In *CIKM*, pages 2513–2516, 2013.

[22] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. Cdas: a crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.

[23] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.

[24] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, pages 1203–1212, 2012.

[25] S. Perera, A. Sheth, K. Thirunarayan, S. Nair, and N. Shah. Challenges in understanding clinical notes: Why nlp engines fall short and where background knowledge can help. In *CIKM Workshop*, 2013.

[26] W. Raghupathi and V. Raghupathi. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2014.

[27] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. K. Schuler, and C. G. Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *JAMIA*, 17(5):507–513, 2010.

[28] T. K. Sean Goldberg, Daisy Zhe Wang. Castle: Crowd-assisted system for textual labeling & extraction. *HCOM*, 2013.

[29] B. Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2010.

[30] M. Stonebraker, D. Bruckner, I. Ilyas, G. Beskales, M. Cherniack, S. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.

[31] J. Sun, J. Hu, D. Luo, M. Markatou, F. Wang, S. Edabollahi, S. E. Steinhubl, Z. Daar, and W. F. Stewart. Combining knowledge and data driven insights for identifying risk factors using electronic health records. In *AMIA*, 2012.

[32] J. Wiens and J. Guttag. Active learning applied to patient-adaptive heartbeat classification. In *NIPS*, pages 2442–2450, 2010.

On Power Law Distributions in Large-scale Taxonomies

Rohit Babbar, Cornelia Metzig, Ioannis Partalas, Eric Gaussier and Massih-Reza Amini
Université Grenoble Alpes, CNRS
F-38000 Grenoble, France
rohit.babbar@imag.fr, cornelia.metzig@imag.fr, ioannis.partalas@imag.fr, eric.gaussier@imag.fr, massih-reza.amini@imag.fr

ABSTRACT

In many of the large-scale physical and social complex systems phenomena fat-tailed distributions occur, for which different generating mechanisms have been proposed. In this paper, we study models of generating power law distributions in the evolution of large-scale taxonomies such as Open Directory Project, which consist of websites assigned to one of tens of thousands of categories. The categories in such taxonomies are arranged in tree or DAG structured configurations having parent-child relations among them. We first quantitatively analyse the formation process of such taxonomies, which leads to power law distribution as the stationary distributions. In the context of designing classifiers for large-scale taxonomies, which automatically assign unseen documents to leaf-level categories, we highlight how the fat-tailed nature of these distributions can be leveraged to analytically study the space complexity of such classifiers. Empirical evaluation of the space complexity on publicly available datasets demonstrates the applicability of our approach.

1. INTRODUCTION

With the tremendous growth of data on the web from various sources such as social networks, online business services and news networks, structuring the data into conceptual taxonomies leads to better scalability, interpretability and visualization. Yahoo! directory, the open directory project (ODP) and Wikipedia are prominent examples of such web-scale taxonomies. The Medical Subject Heading hierarchy of the National Library of Medicine is another instance of a large-scale taxonomy in the domain of life sciences. These taxonomies consist of classes arranged in a hierarchical structure with parent-child relations among them and can be in the form of a rooted tree or a directed acyclic graph. ODP for instance, which is in the form of a rooted tree, lists over 5 million websites distributed among close to 1 million categories and is maintained by close to 100,000 human editors. Wikipedia, on the other hand, represents a more complicated directed graph taxonomy structure consisting of over a million categories. In this context, large-scale hierarchical classification deals with the task of automatically assigning labels to unseen documents from a set of target classes which are represented by the leaf level nodes in the hierarchy.

In this work, we study the distribution of data and the hierarchy tree in large-scale taxonomies with the goal of modelling the process of their evolution. This is undertaken by a quantitative study of the evolution of large-scale taxonomy using models of preferential attachment, based on the famous model proposed by Yule [33] and showing that throughout the growth process, the taxonomy exhibits a fat-tailed distribution. We apply this reasoning to both category sizes and tree connectivity in a simple joint model. Formally, a random variable X is defined to follow a power law distribution if for some positive constant a, the complementary cumulative distribution is given as follows:

$$P(X > x) \propto x^{-a}$$

Power law distributions, or more generally fat-tailed distributions that decay slower than Gaussians, are found in a wide variety of physical and social complex systems, ranging from city population, distribution of wealth to citations of scientific articles [23]. It is also found in network connectivity, where the internet and Wikipedia are prominent examples [27; 7]. Our analysis in the context of large-scale web-taxonomies leads to a better understanding of such large-scale data, and also leveraged in order to present a concrete analysis of space complexity for hierarchical classification schemes. Due to the ever increasing scale of training data size in terms of the number of documents, feature set size and number of target classes, the space complexity of the trained classifiers plays a crucial role in the applicability of classification systems in many applications of practical importance.

The space complexity analysis presented in this paper provides an analytical comparison of the trained model for hierarchical and flat classification, which can be used to select the appropriate model a-priori for the classification problem at hand, without actually having to train any models. Exploiting the power law nature of taxonomies to study the training time complexity for hierarchical Support Vector Machines has been performed in [32; 19]. The authors therein justify the power law assumption only *empirically*, unlike our analysis in Section 3 wherein we describe the generative process of large-scale web taxonomies more concretely, in the context of similar processes studied in other models. Despite the important insights of [32; 19], space complexity has not been treated formally so far.

The remainder of this paper is as follows. Related work on reporting power law distributions and on large scale hierarchical classification is presented in Section 2. In Section 3, we recall important growth models and quantitatively justify the formation of power laws as they are found in hi-

erarchical large-scale web taxonomies by studying the evolution dynamics that generate them. More specifically, we present a process that jointly models the growth in the size of categories, as well as the growth of the hierarchical tree structure. We derive from this growth model why the class size distribution at a given level of the hierarchy also exhibits power law decay. Building on this, we then appeal to Heaps' law in Section 4, to explain the distribution of features among categories which is then exploited in Section 5 for analysing the space complexity for hierarchical classification schemes. The analysis is empirically validated on publicly available DMOZ datasets from the Large Scale Hierarchical Text Classification Challenge (LSHTC)[1] and patent data (IPC) [2] from World Intellectual Property Organization. Finally, Section 6 concludes this work.

2. RELATED WORK

Power law distributions are reported in a wide variety of physical and social complex systems [22], such as in internet topologies. For instance [11; 7] showed that internet topologies exhibit power laws with respect to the in-degree of the nodes. Also the size distribution of website categories, measured in terms of number of websites, exhibits a fat-tailed distribution, as empirically demonstrated in [32; 19] for the Open Directory Project (ODP). Various models have been proposed for the generation power law distributions, a phenomenon that may be seen as fundamental in complex systems as the normal distribution in statistics [25]. However, in contrast to the straight-forward derivation of normal distribution via the central limit theorem, models explaining power law formation all rely on an approximation. Some explanations are based on multiplicative noise or on the renormalization group formalism [28; 30; 16]. For the growth process of large-scale taxonomies, models based on preferential attachment are most appropriate, which are used in this paper. These models are based on the seminal model by Yule [33], originally formulated for the taxonomy of biological species, detailed in section 3. It applies to systems where elements of the system are grouped into classes, and the system grows both in the number of classes, and in the total number of elements (which are here documents or websites). In its original form, Yule's model serves as explanation for power law formation in any taxonomy, irrespective of an eventual hierarchy among categories. Similar dynamics have been applied to explain scaling in the connectivity of a network, which grows in terms of nodes and edges via preferential attachment [2]. Recent further generalizations apply the same growth process to trees [17; 14; 29]. In this paper, describe the approximate power-law in the child-to-parent category relations by the model by Klemm et al. [17]. Furthermore, we combine this formation process in a simple manner with the original Yule model in order to explain also a power law in category sizes, i.e. we provide a comprehensive explanation for the formation process of large-scale web taxonomies such as DMOZ. From the second, we infer a third scaling distribution for the number of features per category. This is done via the empirical Heaps's law [10], which describes the scaling relationship between text length and the size of its vocabulary.

Some of the earlier works on exploiting hierarchy among target classes for the purpose of text classification have been studied in [18; 6] and [8] wherein the number of target classes were limited to a few hundreds. However, the work by [19] is among the pioneering studies in hierarchical classification towards addressing web-scale directories such as Yahoo! directory consisting of over 100,000 target classes. The authors analyse the performance with respect to accuracy and training time complexity for flat and hierarchical classification. More recently, other techniques for large-scale hierarchical text classification have been proposed. Prevention of error propagation by applying *Refined Experts* trained on a validation set was proposed in [4]. In this approach, bottom-up information propagation is performed by utilizing the output of the lower level classifiers in order to improve classification at top level. The deep classification method proposed in [31] first applies hierarchy pruning to identify a much smaller subset of target classes. Prediction of a test instance is then performed by re-training Naive Bayes classifier on the subset of target classes identified from the first step. More recently, Bayesian modelling of large-scale hierarchical classification has been proposed in [15] in which hierarchical dependencies between the parent-child nodes are modelled by centring the prior of the child node at the parameter values of its parent.

In addition to prediction accuracy, other metrics of performance such as prediction and training speed as well as space complexity of the model have become increasingly important. This is especially true in the context of challenges posed by problems in the space of Big Data, wherein an optimal trade-off among such metrics is desired. The significance of prediction speed in such scenarios has been highlighted in recent studies such as [3; 13; 24; 5]. The prediction speed is directly related to space complexity of the trained model, as it may not be possible to load a large trained model in the main memory due to sheer size. Despite its direct impact on prediction speed, no earlier work has focused on space complexity of hierarchical classifiers.

Additionally, while the existence of power law distributions has been used for analysis purposes in [32; 19] no thorough justification is given on the existence of such phenomenon. Our analysis in Section 3, attempts to address this issue in a quantitative manner. Finally, power law semantics have been used for model selection and evaluation of large-scale hierarchical classification systems [1]. Unlike problems studied in classical machine learning sense which deal with a limited number of target classes, this application forms a blue-print on extracting hidden information in big data.

3. POWER LAW IN LARGE-SCALE WEB TAXONOMIES

We begin by introducing the complementary cumulative size distribution for category sizes. Let N_i denote the size of category i (in terms of number of documents), then the probability that $N_i > N$ is given by

$$P(N_i > N) \propto N^{-\beta} \tag{1}$$

where $\beta > 0$ denotes the exponent of the power law distribution.[3] Empirically, it can be assessed by plotting the rank of a category's size against its size (see Figure 1) The derivative of this distribution, the category size probability

[1] http://lshtc.iit.demokritos.gr/

[2] http://web2.wipo.int/ipcpub/

[3] To avoid confusion, we denote the power law exponents for in-degree distribution and feature size distribution γ and δ.

density $p(N_i)$, then also follows a power law with exponent $(\beta + 1)$, i.e. $p(N_i) \propto N_i^{-(\beta+1)}$.
Two of our empirical findings are a power law for both the complementary cumulative category size distribution and the counter-cumulative in-degree distribution, shown in Figures 1 and 2, for LSHTC2-DMOZ dataset which is a subset of ODP. The dataset[4] contains 394,000 websites and 27,785 categories. The number of categories at each level of the hierarchy is shown in Figure 3.

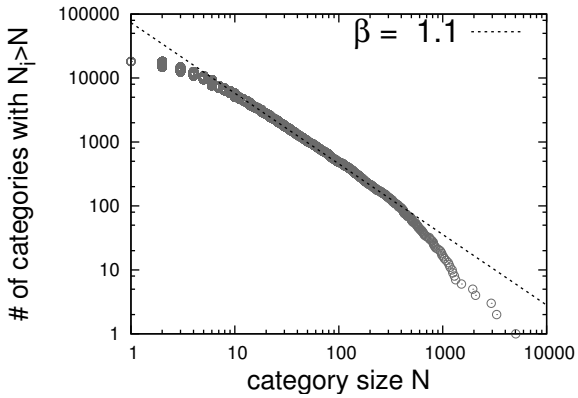


Figure 1: Category size vs rank distribution for the LSHTC2-DMOZ dataset.

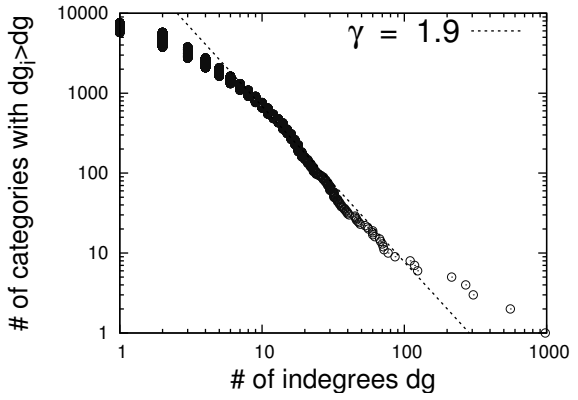


Figure 2: Indegree vs rank distribution for the LSHTC2-DMOZ dataset.

We explain the formation of these two laws via models by Yule [33] and a related model by Klemm [17], detailed in sections 3.1 and 3.2, which are then related in section 3.3.

3.1 Yule's model

Yule's model describes a system that grows in two quantities, in elements and in classes in which the elements are assigned. It assumes that for a system having κ classes, the probability that a new element will be assigned to a certain class is proportional to its current size,

$$p(i) = \frac{N_i}{\sum_{i'=1}^{\kappa} N_{i'}} \tag{2}$$

[4]http://lshtc.iit.demokritos.gr/LSHTC2_datasets

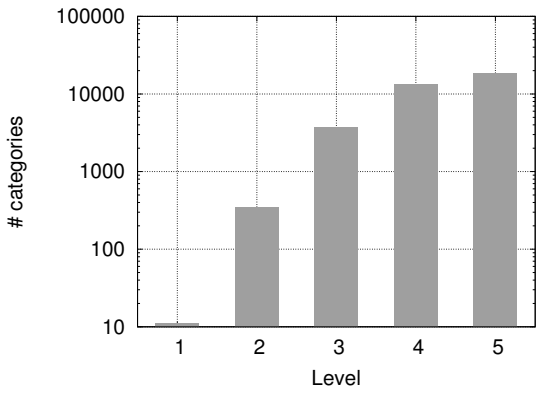


Figure 3: Number of categories at each level in the hierarchy of the LSHTC2-DMOZ database.

It further assumes that for every m elements that are added to the pre-existing classes in the system, a new class of size 1 is created[5].
The described system is constantly growing in terms of elements and classes, so strictly speaking, a stationary state does not exist [20]. However, a stationary distribution, the so-called Yule distribution, has been derived using the approach of the master equation with similar approximations by [26; 23; 17]. Here, we follow Newman [23], who considers as one time-step the duration between creation of two consecutive classes. From this follows that the average number of elements per class is always $m + 1$, and the system contains $\kappa(m + 1)$ elements at a moment where the number of classes is κ. Let $p_{N,\kappa}$ denote the fraction of classes having N elements when the total number of classes is κ. Between two successive time instances, the probability for a given pre-existing class i of size N_i to gain a new element is $mN_i/(\kappa(m + 1))$. Since there are $\kappa\, p_{N,\kappa}$ classes of size N, the expected number such classes which gain a new element (and grow to size $(N + 1)$) is given by :

$$\frac{mN}{\kappa(m+1)}\,\kappa\, p_{N,\kappa} = \frac{m}{(m+1)}\,N\, p_{N,\kappa} \tag{3}$$

The number of classes with N websites are thus fewer by the above quantity, but some which had $(N-1)$ websites prior to the addition of a new class have now one more website. This step depicting the change of the state of the system from κ classes to $(\kappa + 1)$ classes is shown in Figure 4. Therefore, the expected number of classes with N documents when the number of classes is $(\kappa+1)$ is given by the following equation:

$$(\kappa+1)p_{N,(\kappa+1)} = \kappa\, p_{N,\kappa} + \frac{m}{m+1}[(N-1)(p_{(N-1),\kappa}) - Np_{N,\kappa}] \tag{4}$$

The first term in the right hand side of Equation 4 corresponds to classes with N documents when the number of classes is κ. The second term corresponds to the contribution from classes of size $(N-1)$ which have grown to size N, this is shown by the left arrow (pointing rightwards) in Figure 4. The last term corresponds to the decrease resulting

[5]The initial size may be generalized to other small sizes; for instance Tessone et al. consider entrant classes with size drawn from a truncated power law [29] .

Variables	
N_i	Number of elements in class i
dg_i	Number of subclasses of class i
d_i	Number of features of class i
κ	Total number of classes
DG	Total number of in-degrees (=subcategories)
$p_{N,\kappa}$	Fraction of classes having N elements when the total number of classes is κ
Constants	
m	Number of elements added to the system after which a new class is added
w	$\in [0,1]$ Probability that attachment of subcategories is preferential
Indices	
i	Index for the class

Table 1: Summary of notation used in Section 3

from classes which have gained an element and have become of size $(N+1)$, this is shown by the right arrow (pointing rightwards) in Figure 4. The equation for the class of size 1 is given by:

$$(\kappa+1)p_{1,(\kappa+1)} = \kappa\, p_{1,\kappa} + 1 - \frac{m}{m+1}p_{1,\kappa} \qquad (5)$$

As the number κ of classes (and therefore the number of elements $\kappa(m+1)$) in the system increases, the probability that a new element is classified into a class of size N, given by Equation 3, is assumed to remain constant and independent of κ. Under this hypothesis, the stationary distribution for class sizes can be determined by solving Equation 4 and using Equation 5 as the initial condition. This is given by

$$p_N = (1+1/m)B(N, 2+1/m) \qquad (6)$$

where $B(.,.)$ is the beta distribution. Equation 6 has been termed *Yule distribution* [26]. Written for a continuous variable N, it has a power law tail:

$$p(N) \propto N^{-2-\frac{1}{m}}$$

From the above equation the exponent of the density function is between 2 and 3. Its cumulative size distribution $P(N_k > N)$, as given by Equation 1, has an exponent given by

$$\beta = (1+(1/m)) \qquad (7)$$

which is between 1 and 2. The higher the frequency $1/m$ at which new classes are introduced, the bigger β becomes, and the lower the average class size. This exponent is stable over time although the taxonomy is constantly growing.

3.2 Preferential attachment models for networks and trees

A similar model has been formulated for network growth by Barabási and Albert [2], which explains the formation of a power law distribution in connectivity degree of nodes. It assumes that the networks grow in terms of nodes and edges, and that every newly added node to the system connects with a fixed number of edges to existing nodes. Attachment is again preferential, i.e. the probability for a newly added node i to connect to a certain existing node j is proportional to its number of existing edges of node j.

A node in the Barabási-Albert (BA) model corresponds a class in Yule's model, and a new edge to two newly assigned element. Every added edge counts both to the degree of an existing node j, as well as to the newly added node i. For this reason the existing nodes j and the newly added node i grow always by the same number of edges, implying $m = 1$ and consequently $\beta = 2$ in the BA-model, *independently* of the number of edges that each new node creates.

The seminal BA-model has been extended in many ways. For hierarchical taxonomies, we use a preferential attachment model for trees by [17]. The authors considered growth via directed edges, and explain power law formation in the *in-degree*, i.e. the edges directed from children to parent in a tree structure. In contrast to the BA-model, newly added nodes and existing nodes do not increase their in-degree by the same amount, since new nodes start with an in-degree of 0. Leaf nodes thus cannot attract attachment of nodes, and preferential attachment alone cannot lead to a power-law. A small random term ensures that some nodes attach to existing ones independently of their degree, which is the analogous to the start of a new class in the Yule model. The probability v that a new node attaches as a child to the existing node i of with indegree dg_i becomes

$$v(i) = w\frac{d_i - 1}{DG} + (1-w)\frac{1}{DG}\,, \qquad (8)$$

where DG is the size of the system measured in the total number of in-degrees. $w \in [0,1]$ denotes the probability that the attachment is preferential, $(1-w)$ the probability that it is random to any node, independently of their numbers of indegrees. As it has been done for the Yule process [26; 23; 14; 29], the stationary distribution is again derived via the master Equation 4. The exponent of the asymptotic power law in the in-degree distribution is $\beta = 1 + 1/w$.This model is suitable to explain scaling properties of the tree or network structure of large-scale web taxonomies, which have also been analysed empirically, for instance for subcategories of Wikipedia [7]. It has also been applied to directory trees in [14].

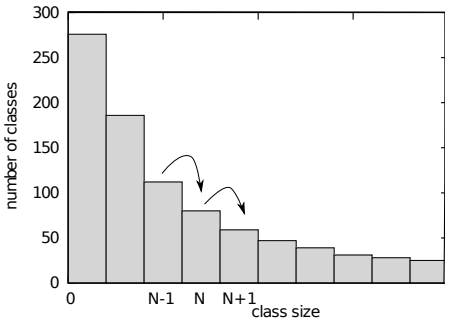


Figure 4: Illustration of Equation 4. Individual classes grow constantly i.e., move to the right over time, as indicated by arrows. A stationary distribution means that the height of each bar remains constant.

3.3 Model for hierarchical web taxonomies

We now apply these models to large-scale web taxonomies like DMOZ. Empirically, we uncovered two scaling laws: (a)

one for the size distribution of leaf categories and (b) one for the indegree (child-to-parent link) distribution of categories (shown in Figure 2). These two scaling laws are linked in a non-trivial manner: a category may be very small or even not contain any websites, but nevertheless be highly connected. Since on the other hand (a) and (b) arise jointly, we propose here a model generating the two scaling laws in a simple generic manner. We suggest a combination of the two processes detailed in subsections 3.1 and 3.2 to describe the growth process: websites are continuously added to the system, and classified into categories by human referees. At the same time, the categories are not a mere set, but form a tree structure, which grows itself in two quantities: in the number nodes (categories) and in the number of in-degrees of nodes (child-to-parent links, i.e. subcategory-to-category links). Based on the rules for voluntary referees of the DMOZ how to classify websites, we propose a simple combined description of the process. Altogether, the database grows in *three* quantities:

(i) *Growth in websites.* New websites are assigned into categories i, with probability $p(i) \propto N_i$ (Figure 5). This assignment happens independently of the hierarchy level of category. However, only leaf categories may receive documents.

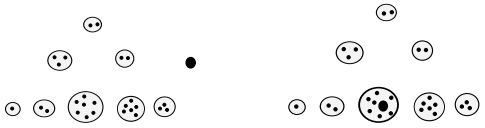


Figure 5: (i): A website is assigned to existing categories with $p(i) \propto N_i$.

(ii) *Growth in categories.* With probability $1/m$, the referees assign a website into a newly created category, at any level of the hierarchy (Figure 6).

This assumption would suffice to create a power law in the category size distribution, but since a tree-structure among categories exists, we also assume that the event of category creation is also attaching at particular places to the tree structure. The probability $v(i)$ that a category is created as the child of a certain parent category i can depend in addition on the *in-degree* d_i of that category (see Equation 9).

Figure 6: (ii): Growth in categories is equivalent to growth of the tree structure in terms of in-degrees.

(iii) *Growth in children categories.* Finally, the hierarchy may also grow in terms of levels, since with a certain probability $(1 - w)$, new children categories are assigned independently of the number of children, i.e. its in-degree d_i of the category i. (Figure 7). Like in [17], the attachment probability to a parent i is

$$v(i) = w\frac{dg_i - 1}{DG} + (1 - w)\frac{\epsilon_i}{DG}\,. \qquad (9)$$

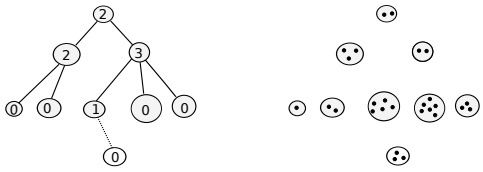


Figure 7: (iii): Growth in children categories.

Equation 8, where $\epsilon_i = 1$, would suffice to explain power law in-degrees dg_i and in category sizes N_i.

To link the two processes more plausibly, it can be assumed that the second term in Equation 9 denoting assignment of new 'first children' depends on the size N_i of parent categories,

$$\epsilon_i = \frac{N_i}{N}\,, \qquad (10)$$

since this is closer to the rules by which the referees create new categories, but is not essential for the explanation of the power laws. It reflects that the bigger a leaf category, the higher the probability that referees create a child category when assigning a new website to it.

To summarize, the central idea of this joint model is to consider two measures for the size of a category: the number of its websites N_i (which governs the preferential attachment of new websites), and its in-degree, i.e. the number of its children dg_i, which governs the preferential attachment of new categories. To explain the power law in the category sizes, assumptions (i) and (ii) are the requirements. For the power law in the number of indegrees, assumptions (ii) and (iii) are the requirements. The empirically found exponents $\beta = 1.1$ and $\gamma = 1.9$ yield a frequency of new categories 1/m=0.1 and a frequency of new indegrees $(1 - w) = 0.9$.

3.4 Other interpretations

Instead of assuming in Equations 9 and 10 that referees decide to open a single child category, it is more realistic to assume that an existing category is *restructured*, i.e. one or several child categories are created, and websites are moved into these new categories such that the parent category contains less websites or even none at all. If one of the new children categories inherits all websites of the parent category (see Figure 8), the Yule model applies directly. If the websites are partitioned differently, the model contains effective shrinking of categories. This is not described by the Yule model, and the master Equation 4 considers only growing categories. However, it has been shown [29; 21] that models including shrinking categories also lead to the formation of power laws. Further generalizations compatible with power law formation are that new categories do not necessarily start with one document, and that the frequency of new categories does not need to be constant.

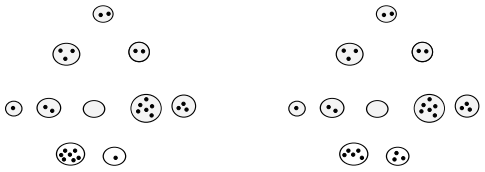


Figure 8: Model without and with shrinking categories. In the left figure, a child category inherits all the elements of its parent and takes its place in the size distribution.

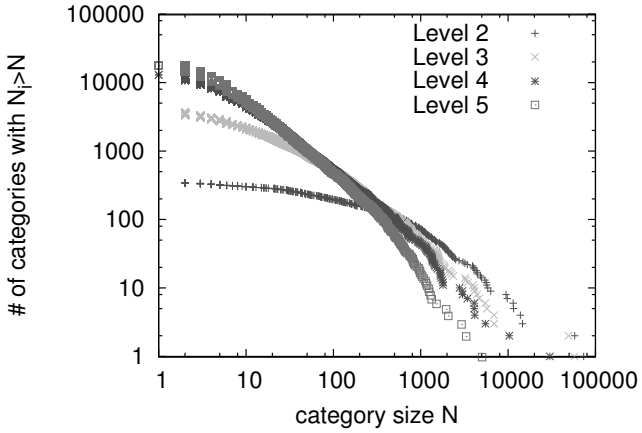


Figure 9: Category size distribution for each level of the LSHTC2-DMOZ dataset.

3.5 Limitations

However, Figures 1 and 2 do not exhibit perfect power law decay for several reasons. Firstly, the dataset is limited. Secondly, the hypothesis that the assignment probability (Equation 2) depends uniquely on the size of a category might be too strong for web directories, neglecting the change in importance of topics. In reality, big categories can exist which receive only few new documents or none at all. Dorogovtsev and Mendes [9] have studied this problem by introducing an assignment probability that decays exponentially with age. For a low decay parameter they show that the stronger this decay, the steeper the power law; for strong decay, no power law forms. A last reason might be that referees re-structure categories in ways strongly deviating from the rules (i) - (iii).

3.6 Statistics per hierarchy level

The tree-structure of a database allows also to study the sizes of class belonging to a given level of the hierarchy. As shown in Figure 3 the DMOZ database contains 5 levels of different size. If only classes on a given level l of the hierarchy are considered, we equally found a power law in category size distribution as shown in Figure 9. Per-level power law decay has also been found for the in-degree distribution. This result may equally be explained by the model introduced above: Equations 2 and 9 respectively, are valid also if instead of $p(k)$ one considers the conditional probability $p(l)p(i|l)$, where $p(l) = \frac{\sum_{i'=1,l}^{\kappa} N_{i',l}}{\sum_{i'=1}^{\kappa} N_{i'}}$ is the probability of assignment to a given level, and $p(i|l) = \frac{N_{i,l}}{\sum_{i'=1,l}^{\kappa} N_{i',l}}$ the probability of being assigned to a given class within that level. The formation process may be seen as a Yule process *within* a level if $\sum_{i'=1,l}^{\kappa} N_{i',l}$ is used for the normalization in Equation 2, and this formation happens with probability $p(l)$ that a website gets assigned into level l. Thereby, the rate at m_l at which new classes are created need not be the same for every level, and therefore the exponent of the power law fit may vary from level to level. Power law decay for the per-level class size distribution is a straightforward corollary of the described formation process, and will be used in Section 5 to analyse the space complexity of hierarchical classifiers.

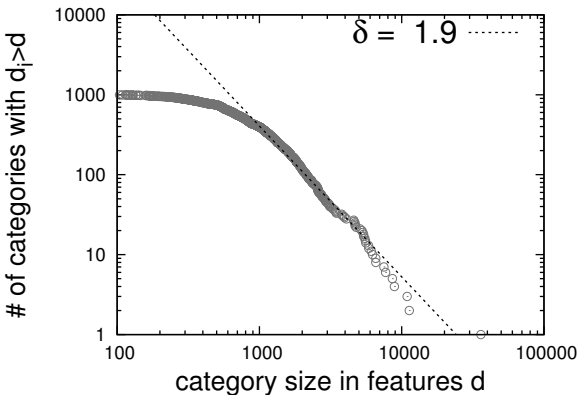


Figure 11: Number of features vs rank distribution.

4. RELATION BETWEEN CATEGORY SIZE AND NUMBER OF FEATURES

Having explained the formation of two scaling laws in the database, a third one has been found for the number of features d_i in each category, $G(d)$ (see Figures 11 and 12). This is a consequence of both the category size distribution, shown (in Figure 1) in combination with another power law, termed Heaps' law [10]. This empirical law states that the number of distinct words R in a document is related to the length n of a document as follows

$$R(n) = Kn^{\alpha}\,, \tag{11}$$

where the empirical α is typically between 0.4 and 0.6. For the LSHTC2-DMOZ dataset, Figure 10 shows that for the collection of words and the collection of websites, similar exponents are found. An interpretation of this result is that the total number words in a category can be measured approximately by the number of websites in a category, although not all websites have the same length.

Figure 10 shows that bigger categories contain also more features, but this increase is weaker than the increase in websites. This implies that less very 'feature-rich' categories exist, which is also reflected in the high decay exponent $\delta = 1.9$ of a power-law fit in Figure 11, (compared to the slower decay of the category size distribution shown in figure 1 where $\beta = 1.1$). Catenation of the size distribution measured in features and Heaps' law yields again size distribution measured in websites: $P(i) = R(G(d_i))$, i.e. multiplication of the exponents yields that $\delta \cdot \alpha = 1.1$ which confirms our empirically found value $\beta = 1.1$.

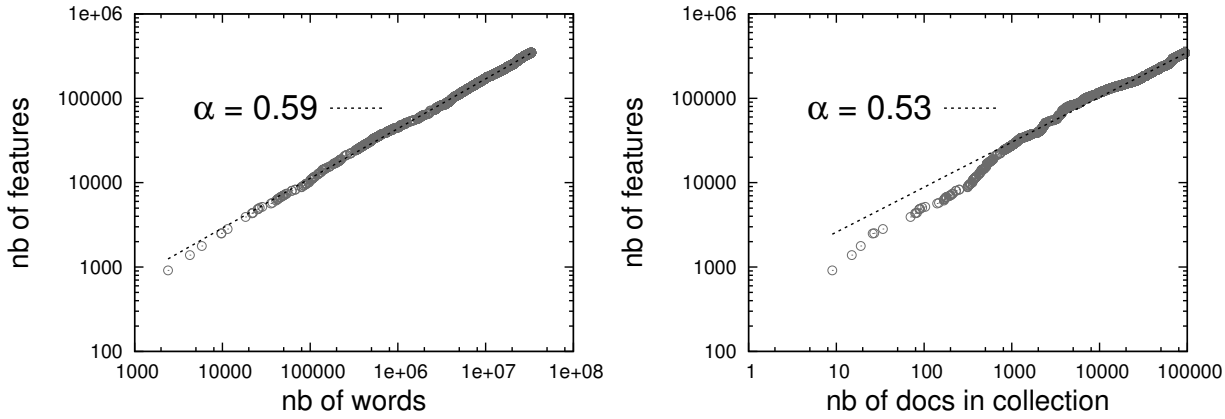


Figure 10: Heaps' law: number of distinct words vs. number of words, and vs number of documents.

5. SPACE COMPLEXITY OF LARGE-SCALE HIERARCHICAL CLASSIFICATION

Fat-tailed distributions in large-scale web taxonomies highlight the underlying structure and semantics which are useful to visualize important properties of the data especially in big data scenarios. In this section we focus on the applications in the context of large-scale hierarchical classification, wherein the fit of power law distribution to such taxonomies can be leveraged to concretely analyse the space complexity of large-scale hierarchical classifiers in the context of a generic linear classifier deployed in top-down hierarchical cascade.

In the following sections we first present formally the task of hierarchical classification and then we proceed to the space complexity analysis for large-scale systems. Finally, we empirically validate the derived bounds.

5.1 Hierarchical Classification

In single-label multi-class hierarchical classification, the training set can be represented by $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$. In the context of text classification, $\mathbf{x}^{(i)} \in \mathcal{X}$ denotes the vector representation of document i in an input space $\mathcal{X} \subseteq \mathbb{R}^d$.

The hierarchy in the form of rooted tree is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} \supseteq \mathcal{Y}$ denotes the set of nodes of $\mathcal{G}$, and $\mathcal{E}$ denotes the set of edges with parent-to-child orientation. The leaves of the tree which usually form the set of target classes is given by $\mathcal{Y} = \{u \in \mathcal{V} : \nexists v \in \mathcal{V}, (u, v) \in \mathcal{E}\}$. Assuming that there are K classes, the label $y^{(i)} \in \mathcal{Y}$ represents the class associated with the instance $\mathbf{x}^{(i)}$. The hierarchical relationship among categories implies a transition from generalization to specialization as one traverses any path from root towards the leaves. This implies that the documents which are assigned to a particular leaf also belong to the inner nodes on the path from the root to that leaf node.

5.2 Space Complexity

The prediction speed for large-scale classification is crucial for its application in many scenarios of practical importance. It has been shown in [32; 3] that hierarchical classifiers are usually faster to train and test time as compared to flat classifiers. However, given the large physical memory of modern systems, what also matters in practice is the size of the trained model with respect to the available physical memory. We, therefore, compare the space complexity of hierarchical and flat methods which governs the size of the trained model in large scale classification. The goal of this analysis is to determine the conditions under which the size of the hierarchically trained linear model is lower than that of flat model.

As a prototypical classifier, we use a linear classifier of the form $\mathbf{w}^T\mathbf{x}$ which can be obtained using standard algorithms such as Support Vector Machine or Logistic Regression. In this work, we apply one-vs-all $L2$-regularized $L2$-loss support vector classification as it has been shown to yield state-of-the-art performance in the context of large scale text classification [12]. For flat classification one stores weight vectors $\mathbf{w}_y, \forall y$ and hence in a K class problem in d dimensional feature space, the space complexity for flat classification is:

$$Size_{Flat} = d \times K \tag{12}$$

which represents the size of the matrix consisting of K weight vectors, one for each class, spanning the entire input space. We need a more sophisticated analysis for computing the space complexity for hierarchical classification. In this case, even though the total number of weight vectors is much more since these are computed for all the nodes in the tree and not only for the leaves as in flat classification. Inspite of this, the size of hierarchical model can be much smaller as compared to flat model in the large scale classification. Intuitively, when the feature set size is high (top levels in the hierarchy), the number of classes is less, and on the contrary, when the number of classes is high (at the bottom), the feature set size is low.

In order to analytically compare the relative sizes of hierarchical and flat models in the context of large scale classification, we assume power law behaviour with respect to the number of features, across levels in the hierarchy. More precisely, if the categories at a level in the hierarchy are ordered with respect to the number of features, we observe a power law behaviour. This has also been verified empirically as illustrated in Figure 12 for various levels in the hierarchy, for one of the datasets used in our experiments. More formally, the feature size $d_{l,r}$ of the r-th ranked category, according to the number of features, for level l, $1 \leq l \leq L-1$, is given by:

$$d_{l,r} \approx d_{l,1} r^{-\beta_l} \tag{13}$$

where $d_{l,1}$ represents the feature size of the category ranked 1 at level l and $\beta > 0$ is the parameter of the power law. Using this ranking as above, let $b_{l,r}$ represent the number of children of the r-th ranked category at level l ($b_{l,r}$ is the branching factor for this category), and let B_l represents the total number of categories at level l. Then the size of the entire hierarchical classification model is given by:

$$Size_{Hier} = \sum_{l=1}^{L-1} \sum_{r=1}^{B_l} b_{l,r} d_{l,r} \approx \sum_{l=1}^{L-1} \sum_{r=1}^{B_l} b_{l,r} d_{l,1} r^{-\beta_l} \quad (14)$$

Here level $l = 1$ corresponds to the root node, with $B_1 = 1$.

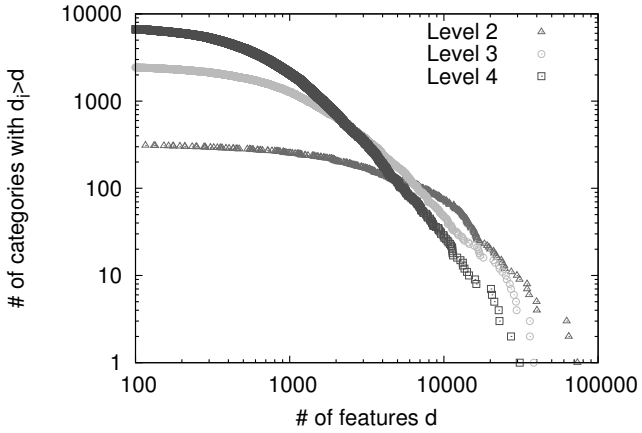


Figure 12: Power-law variation for features in different levels for LSHTC2-a dataset, Y-axis represents the feature set size plotted against rank of the categories on X-axis

We now state a proposition that shows that, under some conditions on the depth of the hierarchy, its number of leaves, its branching factors and power law parameters, the size of a hierarchical classifier is below that of its flat version.

PROPOSITION 1. *For a hierarchy of categories of depth L and K leaves, let $\beta = \min_{1 \leq l \leq L} \beta_l$ and $b = \max_{l,r} b_{l,r}$. Denoting the space complexity of a hierarchical classification model by $Size_{hier}$ and the one of its corresponding flat version by $Size_{flat}$, one has:*

$$For\ \beta > 1, if\ \beta > \frac{K}{K - b(L-1)} (> 1),\ then \quad Size_{hier} < Size_{flat} \quad (15)$$

$$For\ 0 < \beta < 1, if\ \frac{b^{(L-1)(1-\beta)} - 1}{(b^{(1-\beta)} - 1)} < \frac{1-\beta}{b} K, then \quad Size_{hier} < Size_{flat} \quad (16)$$

PROOF. As $d_{l,1} \leq d_1$ and $B_l \leq b^{(l-1)}$ for $1 \leq l \leq L$, one has, from Equation 14 and the definitions of β and b:

$$Size_{hier} \leq b d_1 \sum_{l=1}^{L-1} \sum_{r=1}^{b^{(l-1)}} r^{-\beta}$$

One can then bound $\sum_{r=1}^{b^{(l-1)}} r^{-\beta}$ using ([32]):

$$\sum_{r=1}^{b^{(l-1)}} r^{-\beta} < \left[\frac{b^{(l-1)(1-\beta)} - \beta}{1-\beta} \right] \text{ for } \beta \neq 0, 1 \quad (17)$$

leading to, for $\beta \neq 0, 1$:

$$\begin{aligned} Size_{hier} &< b d_1 \sum_{l=1}^{L-1} \left[\frac{b^{(l-1)(1-\beta)} - \beta}{1-\beta} \right] \\ &= b d_1 \left[\frac{b^{(L-1)(1-\beta)} - 1}{(b^{(1-\beta)} - 1)(1-\beta)} - (L-1) \frac{\beta}{(1-\beta)} \right] \end{aligned} \quad (18)$$

where the last equality is based on the sum of the first terms of the geometric series $(b^{(1-\beta)})^l$.

If $\beta > 1$, since $b > 1$, it implies that $\frac{b^{(L-1)(1-\beta)} - 1}{(b^{(1-\beta)} - 1)(1-\beta)} < 0$. Therefore, Inequality 18 can be re-written as:

$$Size_{hier} < b d_1 (L-1) \frac{\beta}{(\beta - 1)}$$

Using our notation, the size of the corresponding flat classifier is: $Size_{flat} = K d_1$, where K denotes the number of leaves. Thus:

$$\text{If } \beta > \frac{K}{K - b(L-1)} (> 1), \text{then } Size_{hier} < Size_{flat}$$

which proves Condition 15.

The proof for Condition 16 is similar: assuming $0 < \beta < 1$, it is this time the second term in Equation 18 ($-(L-1)\frac{\beta}{(1-\beta)}$) which is negative, so that one obtains:

$$Size_{hier} < b d_1 \left[\frac{b^{(L-1)(1-\beta)} - 1}{(b^{(1-\beta)} - 1)(1-\beta)} \right]$$

and then:

$$\text{If } \frac{b^{(L-1)(1-\beta)} - 1}{(b^{(1-\beta)} - 1)} < \frac{1-\beta}{b} K, \text{then } Size_{hier} < Size_{flat}$$

which concludes the proof of the proposition. □

It can be shown, but this is beyond the scope of this paper, that Condition 16 is satisfied for a range of values of $\beta \in]0, 1[$. However, as is shown in the experimental part, it is Condition 15 of Proposition 1 that holds in practice.

The previous proposition complements the analysis presented in [32] in which it is shown that the training and test time of hierarchical classifiers is importantly decreased with respect to the ones of their flat counterpart. In this work we show that the space complexity of hierarchical classifiers is also better, under a condition that holds in practice, than the one of their flat counterparts. Therefore, for large scale taxonomies whose feature size distribution exhibit power law decay, hierarchical classifiers should be better in terms of speed than flat ones, due to the following reasons:

1. As shown above, the space complexity of hierarchical classifier is lower than flat classifiers.

2. For K classes, only $O(\log K)$ classifiers need to be evaluated per test document as against $O(K)$ classifiers in flat classification.

In order to empirically validate the claim of Proposition 1, we measured the trained model sizes of a standard top-down hierarchical scheme (TD), which uses a linear classifier at each parent of the hierarchy, and the flat one.

We use the publicly available DMOZ data of the LSHTC challenge which is a subset of Directory Mozilla. More specifically, we used the large dataset of the LSHTC-2010

edition and two datasets were extracted from the LSHTC-2011 edition. These are referred to as LSHTC1-large, LSHTC2-a and LSHTC2-b respectively in Table 2. The fourth dataset (IPC) comes from the patent collection released by World Intellectual Property Organization. The datasets are in the LibSVM format, which have been preprocessed by stemming and stopword removal. Various properties of interest for the datasets are shown in Table 2.

Dataset	#Tr./#Test	#Classes	#Feat.
LSHTC1-large	93,805/34,880	12,294	347,255
LSHTC2-a	25,310/6,441	1,789	145,859
LSHTC2-b	36,834/9,605	3,672	145,354
IPC	46,324/28,926	451	1,123,497

Table 2: Datasets for hierarchical classification with the properties: Number of training/test examples, target classes and size of the feature space. The depth of the hierarchy tree for LSHTC datasets is 6 and for the IPC dataset is 4.

Table 3 shows the difference in trained model size (actual value of the model size on the hard drive) between the two classification schemes for the four datasets, along with the values defined in Proposition 1. The symbol $\bigtriangledown$ refers to the quantity $\frac{K}{K-b(L-1)}$ of condition 15.

Dataset	TD	Flat	β	b	$\bigtriangledown$
LSHTC1-large	2.8	90.0	1.62	344	1.12
LSHTC2-a	0.46	5.4	1.35	55	1.14
LSHTC2-b	1.1	11.9	1.53	77	1.09
IPC	3.6	10.5	2.03	34	1.17

Table 3: Model size (in GB) for flat and hierarchical models along with the corresponding values defined in Proposition 1. The symbol $\bigtriangledown$ refers to the quantity $\frac{K}{K-b(L-1)}$

As shown for the three DMOZ datasets, the trained model for flat classifiers can be an order of magnitude larger than for hierarchical classification. This results from the sparse and high-dimensional nature of the problem which is quite typical in text classification. For flat classifiers, the entire feature set participates for all the classes, but for top-down classification, the number of classes and features participating in classifier training are inversely related, when traversing the tree from the root towards the leaves. As shown in Proposition 1, the power law exponent β plays a crucial role in reducing the model size of hierarchical classifier.

6. CONCLUSIONS

In this work we presented a model in order to explain the dynamics that exist in the creation and evolution of large-scale taxonomies such as the DMOZ directory, where the categories are organized in a hierarchical form. More specifically, the presented process models jointly the growth in the size of the categories (in terms of documents) as well as the growth of the taxonomy in terms of categories, which to our knowledge have not been addressed in a joint framework. From one of them, the power law in category size distribution, we derived power laws at each level of the hierarchy, and with the help of Heaps's law a third scaling law in the features size distribution of categories which we then exploit for performing an analysis of the space complexity of linear classifiers in large-scale taxonomies. We provided a grounded analysis of the space complexity for hierarchical and flat classifiers and proved that the complexity of the former is always lower than that of the latter. The analysis has been empirically validated in several large-scale datasets showing that the size of the hierarchical models can be significantly smaller that the ones created by a flat classifier. The space complexity analysis can be used in order to estimate beforehand the size of trained models for large-scale data. This is of importance in large-scale systems where the size of the trained models may impact the inference time.

7. ACKNOWLEDGEMENTS

This work has been partially supported by ANR project Class-Y (ANR-10-BLAN-0211), BioASQ European project (grant agreement no. 318652), LabEx PERSYVAL-Lab ANR-11-LABX-0025, and the Mastodons project Garguantua.

8. REFERENCES

[1] R. Babbar, I. Partalas, C. Metzig, E. Gaussier, and M.-R. Amini. Comparative classifier evaluation for web-scale taxonomies using power law. In *European Semantic Web Conference*, 2013.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[3] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *Neural Information Processing Systems*, pages 163–171, 2010.

[4] P. N. Bennett and N. Nguyen. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–18, 2009.

[5] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances In Neural Information Processing Systems*, pages 161–168, 2008.

[6] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 78–87, 2004.

[7] A. Capocci, V. D. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi, and G. Caldarelli. Preferential attachment in the growth of social networks: The internet encyclopedia wikipedia. *Physical Review E*, 74(3):036116, 2006.

[8] O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 27–34, 2004.

[9] S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks with aging of sites. *Physical Review E*, 62(2):1842, 2000.

[10] L. Egghe. Untangling herdan's law and heaps' law: Mathematical and informetric arguments. *Journal of the American Society for Information Science and Technology*, 58(5):702–709, 2007.

[11] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[13] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2072–2079, 2011.

[14] M. M. Geipel, C. J. Tessone, and F. Schweitzer. A complementary view on the growth of directory trees. *The European Physical Journal B*, 71(4):641–648, 2009.

[15] S. Gopal, Y. Yang, B. Bai, and A. Niculescu-Mizil. Bayesian models for large-scale hierarchical classification. In *Neural Information Processing Systems*, 2012.

[16] G. Jona-Lasinio. Renormalization group and probability theory. *Physics Reports*, 352(4):439–458, 2001.

[17] K. Klemm, V. M. Eguíluz, and M. San Miguel. Scaling in the structure of directory trees in a computer cluster. *Physical review letters*, 95(12):128701, 2005.

[18] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, 1997.

[19] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD*, 2005.

[20] B. Mandelbrot. A note on a class of skew distribution functions: Analysis and critique of a paper by ha simon. *Information and Control*, 2(1):90–99, 1959.

[21] C. Metzig and M. B. Gordon. A model for scaling in firms' size and growth rate distribution. *Physica A*, 2014.

[22] M. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.

[23] M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 2005.

[24] I. Partalas, R. Babbar, É. Gaussier, and C. Amblard. Adaptive classifier selection in large-scale hierarchical classification. In *ICONIP*, pages 612–619, 2012.

[25] P. Richmond and S. Solomon. Power laws are disguised boltzmann laws. *International Journal of Modern Physics C*, 12(03):333–343, 2001.

[26] H. A. Simon. On a class of skew distribution functions. *Biometrika*, 42(3/4):425–440, 1955.

[27] C. Song, S. Havlin, and H. A. Makse. Self-similarity of complex networks. *Nature*, 433(7024):392–395, 2005.

[28] H. Takayasu, A.-H. Sato, and M. Takayasu. Stable infinite variance fluctuations in randomly amplified langevin systems. *Physical Review Letters*, 79(6):966–969, 1997.

[29] C. J. Tessone, M. M. Geipel, and F. Schweitzer. Sustainable growth in complex networks. *EPL (Europhysics Letters)*, 96(5):58005, 2011.

[30] K. G. Wilson and J. Kogut. The renormalization group and the expansion. *Physics Reports*, 12(2):75–199, 1974.

[31] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 619–626, 2008.

[32] Y. Yang, J. Zhang, and B. Kisiel. A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 96–103, 2003.

[33] G. U. Yule. A mathematical theory of evolution, based on the conclusions of dr. jc willis, frs. *Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character*, 213:21–87, 1925.

Interview: Michael Brodie, Leading Database Researcher, Industry Leader, Thinker

Gregory Piatetsky
KDnuggets
Brookline, MA

gregory@kdnuggets.com

ABSTRACT

We discuss the most important database research advances, industry developments, role of relational and NoSQL databases, Computing Reality, Data Curation, Cloud Computing, Tamr and Jisto startups, what he learned as a chief Scientist of Verizon, Knowledge Discovery, Privacy Issues, and more.

Keywords

Data Curation, NoSQL, Data Curation, Cloud Computing, Verizon, Privacy, Computing Reality.

1. INTRODUCTION

I had a pleasure of working with Michael Brodie when we were both at GTE Laboratories in 1990s, where he was already a world-famous researcher and a department manager. I recently met him at another conference, and our discussion led to this interview. Michael is still very sharp, very active, and busy - he answered these questions while flying from Boston to Doha, Qatar where he is advising Qatar Computing Research Institute.

Parts of this interview were published in KDnuggets [1-3].

2. BACKGROUND

Dr. Michael L. Brodie [4] has served as Chief Scientist of a Fortune 20 company, an Advisory Board member of leading national and international research organizations, and an invited speaker and lecturer. In his role as Chief Scientist Dr. Brodie has researched and analyzed challenges and opportunities in advanced technology, architecture, and methodologies for Information Technology strategies. He has guided advanced deployments of emergent technologies at industrial scale, most recently Cloud Computing and Big Data.

Throughout his career Dr. Brodie has been active in both advanced, academic research and large-scale industrial practice attempting to obtain mutual benefits from the industrial deployment of innovative technologies while helping research to understand industrial requirements and constraints. He has contributed to multi-disciplinary problem solving at scale in contexts such as Terrorism and Individual Privacy, and Information Technology Challenges in Healthcare Reform.

3. INTERVIEW

Gregory Piatetsky: You have started as a researcher in Databases (PhD from Toronto) and had a very distinguished and varied career spanning academia, industry, and government, in US, Europe, Australia, and Latin America over the last 25+ years. From your unique vantage point, what were 3 most important database research advances?

Michael Brodie: Three most important database research advances:

1. Ted Codd's Relational model of data (1970) is the most important database research advance as it launched what is now a $28 BN/year market still growing at 11% CAGR with over 215 RDBMSs on the market. More important to me it launched four decades of amazing research advances starting with query optimization (Selinger) and transactions (Gray) and innovation that has probably grown at 20% CAGR.
2. The next most important research advance or stage was a change in perspective that specific domains require their own DBMS such as graph databases, array stores, document stores, key-value stores, NoSQL, NewSQL, and many more to come. DB-Engines.com lists twelve DBMS categories thus bumping the database world from managing 8% of the world's data to about 12% but due to the growth of non-database data back to 10%. Soon, due to the role of data in our digitized world there will be data management systems for many more domains. While this is amazingly cool, how do we solve multi-disciplinary (multi-data domain) problems in a consistent rather that disjoint way?
3. The next most important research advance is just emerging and is mind blowing. I call it **Computing Reality**, acknowledging that every datum (every real world observation) is not definitive but probabilistic. Unlike conventional databases and more like reality, Computing Reality has *no single version of truth*. How do we model such worlds, more realistic worlds and compute over them? The simple answer is that it is already in Big Data sources. There are many related attempts to address Computing Reality including social computing, probabilistic computing, probabilistic databases, Open Worlds in AI, Web Science, Approximate Computing, Crowd Computing, and more. Perhaps this will be the next generation of computing.

GP: What about the most important database industry developments?

MB: Alas the database industry, like all industries, has a legacy problem that stifles innovation. It has taken over 30 years to emerge from the relational era. The most important recent database industry development came from outside the database industry, it is Big Data and its marketing arm called MapReduce and its data sidekicks, Hadoop and NoSQL. **Frankly, the database industry has been insular and protected its relational turf for FAR too long.** Smart folks at Yahoo!, Google and other places saw value in data, non-database data, and thus emerged MapReduce, Hadoop, and NoSQL- generally crappy database ideas but it woke up the database industry[1]. Hadoop and NoSQL are growing in demand. In time it will be seen that they are amazing for a very specific problem domain, embarrassingly parallel problems, but it is a money pit for everything else. The importance of MapReduce is that it forced the database industry to get out of their hammocks.

GP: What is the role of Relational Databases, NoSQL databases, Graph databases, and other databases today?

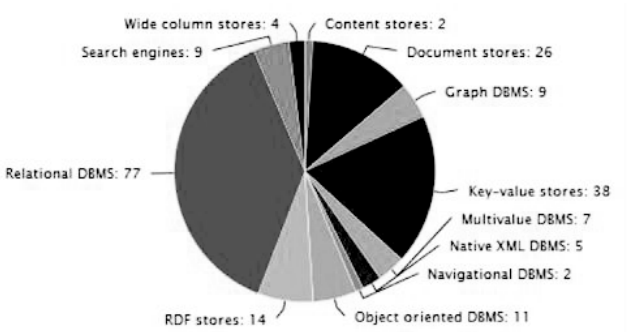
Relational Databases have two extremely well established roles. Conventional row stores serve the OLTP community as the backbone of enterprise operations. These blindingly fast transaction processors are moving in-memory. OLTP stores are modest in number and size (< 1 TB) growing and declining in lock step with business growth and decline. Column stores, OLAP, are the backbone of data warehouses and until recently business intelligence. In general there are huge numbers of these, often of very large size in the Petabyte and Exabyte range. This is where Big Data battle lines are being drawn. What fun!!

This is also where we turn from polishing the relational round ball [5] and focus on the other dozen or so other DBMS categories. Taking over is relative; none of the 12 other categories has more than 3% of the database market. Graph databases serve graph applications like networking in communications, telecom, social networks, and of course NSA applications! But what is wonderful about these emerging classes of data-domain specific DBMSs is that we are only now discovering the rich use cases that they serve.

The use cases define the DBMSs and the DBMSs help formulate the use cases. SciDB is a superb example of managing scientific data and computation at scale. It is awkward for both communities – database folks who don't speak linear algebra or matrices, and scientists who only speak R. Exciting times. For a little fun look at the database-engines list [6].

Database Engines

DB-Engines lists 216 different database management systems, which are classified according to their database model (e.g. relational DBMS, key-value stores etc.). This pie-chart shows the number of systems in each category. Some of the systems belong to more than one category.



Popularity changes per category, April 2014, over 1 year

- Graph DBMS – growing dramatically 3.5X
- Wide column stores – 2X
- Document stores 2X
- Native XML DBMS – 1.5X
- Key-value stores – 1.5X
- Search engines – 1.5X
- RDF stores – 1.5X
- Object oriented DBMS - flat
- Multivalue DBMS - flat
- Relational DBMS - flat

GP: You have held an amazing variety of positions in academy, industry, government organization, VC firms, and start-ups, in US, Brazil, Canada, Australia, and Europe. Which 3 positions were most satisfying to you and why?

MB: What a great question. Thank you for asking because it caused me to think about what I have really enjoyed over 40 years. Somehow CSAIL at MIT and the Faculty of Computing and Communications at EPFL jump to mind.

There are scary smart people at those places. Like climbing mountains it both scares and exhilarates me. To be frank my jobs at big enterprises in hindsight are confusing. I guess I was window dressing because my role did not feel like it had impact. So getting motivated and scared at MIT and EPFL are probably top, so there's number one. Why? Just look down 5,000 feet and ask why am I here?

Second is a combination of Advisory roles at US Academy of Science, DERI, STI, ERCIM, Web Science Trust, and others because they gave me a sense of collaborating, challenging, and contributing. How cool is that?

Third would be working at startups like Tamr and Jisto. Imagine

[1] On June 25, 2014 Google launched Cloud Dataflow replacing MapReduce and marking the decline of MR and Hadoop as predicted at launch in 2010 by Mike Stonebraker in 2010.

waking up in the morning and thinking you might change the world. That requires that I conceive the world not just differently, but so that it solves someone's REAL problem. Even more cool.

Gregory Piatetsky: Currently you are an adviser at a startup called Tamr [7], co-founded by another leading DB researcher and serial entrepreneur Michael Stonebraker. What can you tell us about Tamr and its product?

Michael Brodie: Consider the data universe. Since the 1980's I have said in keynotes that the database and business worlds deal with less than 10% of the world's data most of which is structured, discrete, and conforms to some schema. With the Web and Internet of Things in the 1990s massive amounts of unstructured data began to emerge with a growth rate that was inconceivable while shrinking database data to less than 8%. EMC/IDC claims [14] that our Digital Universe is 4.4 zettabytes and will double every two years until 2020 when it will be 44 zettabytes.

[**If you are constantly amazed at the growth of the Digital World, you don't understand it yet** – A profound, casual comment of my departed friend, Gerard Berry, *Academie Francais.*]

In 1988 or so you, Gregory, and a few others saw the potential of data with your knowledge discovery in databases – then a radical idea. Little did others, including me, realize the potential of this, now named Big Data. Even though Big Data is hot in 2014, almost 30 years later, it's application, tools, and technologies are in their infancy, analogous to the emergence of the Web in the early 1990s. Just as the Web has and is changing the world, so too will Big Data.

Compared with database data, Big Data is crazy. It's largely not understood hence it is schema-less or model-less. **Big Data is inconceivably massive, dirty, imprecise, incomplete, and heterogeneous beyond anything we've seen before. Yet it trumps finite, precise, database data in many ways hence is a treasure trove of value.** Big Data is qualitatively different from database data that is a small subset of Big Data - EMC/IDC claims 1% as of 2013. It offers far greater potential thus value and requires different thinking, tools, and techniques. Database data is approached top-down. Telco billing folks know billing inside out so they create models that they impose, top-down on data. Data that does not comply is erroneous. Database data, like Telco bills must be precise with a *single version of truth*, so that the billing amount is justifiable. Due in part to scale, Big Data must be approached bottom up. More fundamentally, we should let data speak; see what models or correlations emerge from the data, e.g., to discover if adding strawberry to the popsicle line-up makes sense (a known unknown) or to discover something we never thought of (unknown unknowns). Rather than impose a preconceived, possibly biased, model on data we should investigate what possible models, interpretations, or correlations are in the data (possibly in the phenomena) that might help us understand it.

Hence, the new paradigm is to approach Big Data bottom-up due to the scale of the data and to let the data speak. Big Data is a different, larger world than the database world. The database world (small data) is a small corner of the Big Data world. Correspondingly Big Data requires new tools, e.g., Big Data Analytics, Machine Learning (the current red haired child), Fourier transforms, statistics, visualizations, in short any model that might help elucidate the wisdom in the data. But how do you get Big Data, e.g., 100 data sources, 1,000, 100,000 or even 500,000, into these tools? How do you identify the 5,000 data sources that include Sally Blogs and consolidate them into a coherent, rationale, consistent view of dear Sally? When questions arise in consolidating Sally's data, how do you bring the relevant human expertise, if needed, to bear – at scale on 1 million people? Many successful Big Data projects report that this data curation process takes 80% of the project resources leaving 20% for the problem at hand. Data curation is so costly because it is largely manual hence it is error prone. That's where Tamr comes to the rescue. It is a solution to curate data at scale.

We call it collaborative data curation because it optimizes the use of indispensable human experts. **Data Curation is for Big Data what Data Integration is for small data.**

Data Curation is bottom up and Data Integration is top down. It took me about a year to understand that fundamental difference. I have spent over 20 years of my professional life dealing with those amazing Data Integration platforms and some of the world's largest data integration applications. Those technologies and platforms apply beautifully to database data – small data; they simply do not apply to Big Data.

To emphasize what is ahead, here is a prediction. Data Integration is increasingly crucial to combining top-down data into meaningful views. Data Integration is a huge challenge and huge market that will not go away. Big Data is orders of magnitude larger than small or database data. Correspondingly Data Curation will be orders of magnitude larger than Data Integration. The world will need Data Curation solutions like Tamr to let data scientists focus on analytics, the essential use and value of big data, while containing the costs of data preparation. In addition to Tamr there are over 65 very cool data curation products contributing to addressing the growing need and creating a new software market. What is also cool about data curation is that it can be used to enrich the existing information assets that are the core of most enterprise's applications and operations. Of course, the really cool potential of data curation is that it makes Big Data analytics more efficiently available to allow users to discover things that they never knew! How cool is that?

For more on Data Curation at scale, see Stonebraker [8].

GP: You also advise another startup Jisto. What can you tell us about your role there?

MB: I am having a blast with Jisto [9]– some amazingly talented young engineers [PhDs actually] with lots of energy and a killer idea. Jisto is an exceptional example of the quality you ask about in the next question.

Cloud computing enabled by virtualization is radically changing the world by reducing the cost and increasing the availability of computing resources. Can you imagine that only 50% of the world's servers are virtualized?

Pop quiz [do not cheat and read ahead].

What is the average CPU utilization of physical servers, worldwide? Of virtual servers?

Answer: Virtual machine CPU utilization is typically in the 30-50% range while physical servers are 10-20%, due to risk and scheduling ,but mostly cultural challenges.

Jisto enables enterprises to transparently run more compute-intensive workloads on these paid-for but unused resources whether on premises or in public or private clouds, thus reducing costs by 75–90% over acquiring more hardware or cloud resources.

Jisto provides a high-performance, virtualized, elastic cloud-computing environment from underutilized enterprise or cloud computing resources (servers, laptops, etc.) without impacting the primary task on those resources. Organizations that will benefit most from Jisto are those that run parallelized compute-intensive applications in the data center or in private and public clouds (e.g., Amazon Web Services, Windows Azure, Google Cloud Platform, IBM SmartCloud).

Jisto is currently looking for early adopters for its beta program who will gain significant reduction in the cost of their computing possibly avoiding costly data center expansion.

GP: You are also a Principal at First Founders Limited. What do you look for in young business ventures - how do you determine quality?

MB: There are armies of people who evaluate the potential of startups. The professional ones are called "Venture Capitalists (VCs). The retired ones are called Angels. Like any serious problem there is due diligence to determine and evaluate the factors relevant to the business opportunity, the technology, the business plan, etc. as the many books [10] and formulas suggest.

If you are reading a book, then you don't know. **Ultimately it comes down to good taste developed over years of successful experience.** Andy Palmer, a serial entrepreneur, good friend, and very smart guy said "Do it once really well then repeat." Andy ought to know, Tamr is about his 25th startup.

At First Founders I can do some technology, Jim can do finance, Howard can do business plans. Collectively we make a judgment. But good VC's are the wizards. They have Rolodexes. When their taste says maybe they refer the startup to the relevant folks in their network who essentially do the due diligence for them. Like at First Founders, the judgment is crowd sourced, actually what we call at Tamr, it is *expert sourced*. I have a growing trust of the crowd and especially of the expert crowd.

GP: You were a Chief Scientist at Verizon for over 10 years (and before that at GTE Labs which became part of Verizon). What were some of the most interesting projects you were involved in at GTE and Verizon?

MB: The technical challenge that stays with me is that addressed by the Verizon Portal, Verizon's solution for Enterprise Telecommunications – providing Telecommunication services to enterprise customers, such as Microsoft. Verizon, like all large Telcos, is the result of the merger & acquisition of 300+ smaller Telcos. Each had at least 3 billing systems; hence Verizon acquired over 1,000 billing systems. Billing is only one of over a dozen systems categories, including sales, marketing, ordering, and provisioning. Providing a customer like Microsoft with a telephone bill for each Microsoft organization requires integrating data potentially from over 1,000 databases. As is the case for most enterprises, Verizon and Microsoft reorganize constantly complicating the sources to be integrated, like Microsoft, and the targets, Verizon's changing businesses, e.g., wireline and FiOS. Every service company faces this little-discussed massive challenge.

Integrating 1,000s of operational systems is a backward looking problem. The cool forward-looking problem was Verizon IT's Standard Operating Environment (SOE). Prior to cloud platforms and cloud providers, Verizon IT (actually one team) sought to develop an SOE onto which Verizon's major applications (0ver 6,000) could be migrated to be managed virtually on an internal cloud. What a fun challenge. When the team left Verizon as a group over 60 major corporate applications, including SAP, had been migrated. Smart folks, good solution that failed in Verizon. **In industry, challenges are 80-20; 80% political, 20% technical**. The SOE is being reborn in the infrastructure of another major infrastructure corporation.

Finally, the next most interesting and yet unsolved industry challenge was getting over the legacy that Mike Stonebraker and I addressed in [11].

How do you keep a massive system up to date in terms of the application requirements and the underlying technology or migrate it to a modern, efficient, more cost effective platform?

Enterprises tend to invest only in new revenue generating opportunities often leaving the legacy problem to grow and grow. So existing systems like billing languish and accumulate. **It's like a teenager never tidying their room for 60 years.** Now where are my blue shoes? I suggested to Mike Stonebraker that we rewrite our 1995 book. He did not even respond to the email, suggesting that it is largely a political problem and not technical, no matter the brilliant technical solution provided.

Lesson: If you are a CIO, *clean up your goddamn room; you're not going out until you do!*

GP: Around 1989 when you were a manager at GTE Labs and I was a member of technical staff there, you were somewhat skeptical of the idea I proposed for research into Knowledge Discovery in Databases (then called KDD or Data Mining, and more recently Predictive Analytics, and Data Science). The field has progressed significantly since then. From your point of view, what are the main successes and disappointments of KDD/Data Mining/Predictive Analytics and can Data Science become an actual science?

MB: My current research concerns the scientific and philosophical underpinnings of Big Data and Data Science. With Big Data we are undergoing a fundamental shift in thinking and in computing. Big Data is a marvelous tool to investigate *What* – correlations or patterns that suggest that things might have or will occur.

Big Data's weakness is that it says nothing about *Why* – causation or why a phenomenon occurred or will occur.

A pernicious aspect of *What* are the biases that we bring to it. On a personal note, my biased recall of 1989 was how marvelous your ideas were and the amazing potential of data mining. I accept your view that I was skeptical rather than enthusiastic as I recall. You see I modified reality to fit my desire to be on the winning side, which I was not then. Hence, what we think that we thought may bear little resemblance to reality or, more precisely other people's reality. As Richard Feynman said,

"The first principle is that you must not fool yourself - and you are the easiest person to fool."

That said, I see the main successes of this trend as a nascent trajectory along the lines of Big Data, Data Analytics, Business Intelligence, Data Science, and whatever the current trendy term is. The World of *What* is phenomenal – machines proposing potential correlations that are beyond our ability to identify. Humans consider seven plus or minus 2 variables at a time, a rather simple model, while models, such as Machine Learning, can consider millions or billions of variables at a time. Yet 95% (or even 99.99999%) of the resulting correlations may be meaningless. For example, ~99% of credit card transactions are legitimate with less than 1% that are fraudulent, yet the 1% can kill the profits of a bank. So precision and outlier cases, called anomalies in science can matter. So it pays to search for apparently anomalous behavior – as it is happening!

We have already seen massive benefits of Big Data in the stock market, electoral predictions, marketing success, and many more that underlie the Big Data explosion. **Yet there is a potential Big Data Winter ahead if people blindly apply Big Data and more specifically Machine Learning.** The failures concern limited models of phenomena and the human tendency of bias. People can and do use *What* (Big Data, etc.) to support their biases and limited models, e.g., used to support the claim of the absence of climate change or lack of human impact on climate change, rather than letting the data speak to suggest directions and models that we may never have thought of. As it has always been, it takes courage to change from a discrete world of top-down models [I know how this works!] to an ambiguous, probabilistic world [What possible ways does this work?].

Those are natural successes and limitations of an emerging field. The direction, opportunities, and changes are profound. I experience a mix of fear and tingles thinking of asking the data to speak. Hoping that I can be open to what it says and distinguishing s..t from Shinola.

I call the vision *Computing Reality*. It may be the Next Generation of Computing.

GP: In your very insightful report of the White House-MIT Big Data Privacy Workshop [12] you have a quote "Big data has rendered obsolete the current approach to protecting privacy and civil liberties". Will people get used to much less privacy (as the digitally-savvy younger people seem to be) or will government regulation and/or technology be able to protect privacy? How will this play in US vs. Europe vs. other regions of the world?

MB: As an undergraduate at the University of Toronto, I was extremely fortunate to have had Kelly Gotlieb, the Father of Computing in Canada, as a mentor. I was a student in his 1971 course, *Computers and Society*, later to become the first book on the topic. Kelly and the issues, including privacy, have resonated with me throughout my career. Kelly observed that privacy, like many other cultural norms, varies over time. So yes, privacy will fluctuate from Alan Westin's notion of determining how your personal information is communicated to the Facebook-esk "Get over it".

While personal privacy is undergoing significant change, disclosure of information assets that are part of the digital economy or of government or corporate strategy may have very significant impacts on our economy and democracy. Hence, this raises issues of security, protection, and cultural and social issues too complex to be treated here.

However, there are a number of very smart people looking at various aspects. The quote you cite is from Craig Mundy [13] who explores changes that Big Data brings debating the balancing of economic versus privacy issues.

Very smart folks, like Butler Lampson and Mike Stonebraker, are commenting on practical solutions to this age-old problem. Their arguments are along the following lines. Due to the massive scale of Big Data, and what I call Computing Reality, previously top-down solutions for security, such as anticipating and preventing security breaches, will simply not scale to Big Data. They must be augmented with new approaches including bottom-up solutions such as Stonebraker's logging to detect and stem previously unanticipated security breaches and Weitzner's accountable systems.

To beat the Heartbleed bug and others like it, "Organizations need

to be able to detect attackers and issues well after they have made it through their gates, find them, and stop them before damage can occur," Gazit, a leading cyber security expert said recently. "The only way to achieve such a laser-precision level of detection is through the use of hyper-dimensional big data analytics, deploying it as part of the very core of the defense mechanisms.

Big Data has rendered obsolete the current approach to protecting privacy and civil liberties."

Hence, Big Data requires a shift from a focus on top-down methods of controlling data generation and collection to a focus on data usage. Not only do top-down methods not scale, "Tightly restricting data collection and retention could rob society of a hugely valuable resource [13]". Adequate let alone complete solutions will take years to develop.

GP: What interesting technical developments you expect in Database and Cloud Technology in the next 5 years?

MB: I call the Big Picture *Computing Reality* in which we model the world from whatever reasonable perspectives emerge from the data and are appropriate, e.g., have veracity, and make decisions symbiotically with machines and people collaborating to optimize resources while achieving measures of veracity for each result.

One subspace of this world is what we currently know with high levels of confidence, the type of information that we store in relational databases. Another encompassing space is what we know but forgot or don't want to remember (*unknown knowns*) and a third is what we speculate but do not know (*known unknowns*), these are all the hypotheses that we make but do not know in science, business, and life.

The rest of the data space – *unknown unknowns* - is infinite; otherwise learning would be at an end. That is the space of discovery.

I am investigating Computing Reality to investigate the entire space with the objective of accelerating Scientific Discovery. This is practically interesting because very little of our world is discrete, bounded, finite, or involves a single version of truth, yet that is the world of most computing. With Computing Reality we hope to be far more pragmatic and realistic. This is technically and theoretically interesting because we have almost no mathematical or computing models in these areas. Those that exist are just emerging or are massively complex. How cool is that? You see what old retired guys get to do?

GP: What do you like to do in your free time? What recent book you liked?

MB: Free time – what a concept! My yoga teacher, Lynne recommended that I should try to do nothing one day, and I will. I will. Soon. Really. Life is such a blast; it's hard to keep still.

(Michael Brodie and his son on a peak in New Hampshire)

My activities include the gym (4 times a week); hiking/climbing ~75 mountains USA, Nepal, Greece, Italy, France, Switzerland, and even Australia; 42 of the 48 4,000 footers in NH (most with Mike Stonebraker); cooking (daily and special occasions with my son Justin, an amazing chef and brewer, when he's not doing his PhD), travel, and my garden; all of these – except the gym and garden - with family and close friends.

Very cool Big Data Books:

Big Data: A Revolution That Will Transform How We Live, Work, and Think by Viktor Mayer-Schonberger, Kenneth Cukier, Houghton Mifflin Harcourt confused and inspired me, then

The Signal and the Noise: Why So Many Predictions Fail-but Some Don't, by Nate Silver, Penguin Press, inspired me.

Real books

- Ken Follett's *The Pillars of the Earth; Century Trilogy* (Fall of Giants, Winter of the World and Edge of Eternity)
- Henning Mankell's *The Fifth Woman (A Kurt Wallander Mystery)*

GP: You just returned from Doha, Qatar where you were advising the Qatar Computing Research Institute (QCRI) - quite far from Silicon Valley, New York, or Boston. What is happening there and what computing research are they doing?

MB: This was my first visit to Qatar that was remarkable culturally an intellectually. Culturally I saw spectacular result of hydrocarbon wealth and vision, e.g., amazing architecture emerging from the dessert. Intellectually I saw the beginnings of Qatar's National Vision 2030 to transform Qatar's economy from hydrocarbon-based to knowledge-based.

One step in this direction by the Qatar Foundation was to create the Qatar Computing Research Institute (QCRI). In less than three years QCRI has established the beginnings of a world-class computer science research group seeded with world-class researchers in strategically important areas such as Social Computing, Data Analysis, Cyber Security, and Arabic Language Technologies (e.g., Machine Learning and Translation) amongst others. Each group already has multiple publications over several years in the leading conferences in their areas, e.g., SIGMOD and VLDB for Data Analysis. I spent my time reviewing with them what I consider to be some of the most challenging issues in Big Data.

4. REFERENCES

[1] Gregory Piatetsky, Exclusive Interview: Michael Brodie, Leading Database Researcher, Industry Leader, Thinker, in KDnuggets, April 2014,

http://www.kdnuggets.com/2014/04/michael-brodie-database-researcher-leader-thinker.html

[2] Gregory Piatetsky, Interview (part 2): Michael Brodie on Data Curation, Cloud Computing, Startup Quality, Verizon, in KDnuggets, May 2014,

http://www.kdnuggets.com/2014/04/interview-michael-brodie-2-data-curation-cloud-computing-verizon.html

[3] Gregory Piatetsky, Interview (part 3): Michael Brodie on Industry Lessons, Knowledge Discovery, and Future Trends, in KDnuggets, May 2014,

http://www.kdnuggets.com/2014/05/interview-michael-brodie-3-industry-lessons-knowledge-discovery-trends-qcri.html

[4] www.michaelbrodie.com/michael_brodie.asp

[5] Michael Stonebraker. Are We Polishing a Round Ball? Panel Abstract. ICDE, page 606. IEEE Computer Society, 1993

[6] Database Engines List, http://db-engines.com/en/blog_post/23 http:/db-engines.com/en/blog_post/23

[7] Gregory Piatetsky, Exclusive: Tamr at the New Frontier of Big Data Curation, KDnuggets, May 2014,

http://www.kdnuggets.com/2014/05/tamr-new-frontier-big-data-curation.html

[8] Stonebraker et al, Data Curation at Scale: The Data Tamer System In CIDR 2013 (Conference on Innovative Data Systems Research).

[9] http://jisto.com/

[10] R. Field, "Disciplined Entrepreneurship: 24 Steps to a Successful Startup by Bill Aulet", Journal of Business & Finance Librarianship, vol. 19, no. 1, pp. 83-86, Jan. 2014.

[11] M. Brodie and M. Stonebraker. Legacy Information Systems Migration: The Incremental Strategy, Morgan Kaufmann Publishers, San Francisco, CA (1995) ISBN 1-55860-330-1

[12] M. Brodie, White House-MIT Big Data Privacy Workshop Report, KDnuggets, Mar 27, 2014.

http://www.kdnuggets.com/2014/03/white-house-mit-big-data-privacy-workshop-report.html

[13] Craig Mundy, Privacy Pragmatism: Focus on Data Use, Not Data Collection, Foreign Affairs, March/April 2014.

[14] The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. IDC/EMC, April 2014.

About the authors:

Gregory Piatetsky is a Data Scientist, co-founder of KDD conferences and ACM SIGKDD society for Knowledge Discovery and Data Mining, and President and Editor of KDnuggets. He tweets about Analytics, Big Data, Data Science, and Data Mining at @kdnuggets.