An Adaptive Information-Theoretic Approach for Identifying Temporal Correlations in Big Data Sets

Nguyen Ho*, Huy Vo†‡, Mai Vu§
*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
†Center for Urban Science and Progress, New York University, New York, USA
‡Department of Computer Science, the City College of New York, New York, USA
§Department of Electrical & Computer Engineering, Tufts University, Medford, MA, USA
Email: thithao.ho@polimi.it, huy.vo@nyu.edu, mai.vu@tufts.edu

***Abstract*—In the past two decades, new developments in computing, sensing and crowdsourced data have resulted in an explosion in the availability of quantitative information. The possibilities of analyzing this so-called "big data" to inform research and the decision-making process are virtually endless. In general analyses have to be done across multiple data sets in order to bring out the most value of big data. A first important step is to identify temporal correlations between data sets. Given the characteristics of big data in term of volume and velocity, techniques that identify correlations not only need to be scalable, but also need to help users in ordering the correlation across temporal resolutions so that they can focus on important relationships. There is a large body of work in this area, however, most of them either only deal with small data sets, using a fixed temporal resolution, or does not provide a quantifiable measure of a correlation significance. In this paper, we present a method based on mutual information to identify correlations in large data sets. Discovered correlations are suggested to users in an order based on their significance. Our method supports an adaptive streaming technique that minimizes duplicated computation and is implemented on top of Apache Spark for scalability using big data platforms. We also provide a comprehensive evaluation using real-world data sets from NYC Open Data, and compare our findings against a recent study.**

***Keywords*-temporal correlation; mutual information; Big Data; adaptive sliding window; streaming**

I. Introduction

The true value of big data comes from analyzing multiple data sets, in which discovering correlations among them is one of the first steps towards creating new values in big data applications. As data originally reside in individual silos, each of them may serve for a specific and/or limited purpose. However, their combination can, and often does, offer new insights into important problems. In particular, data correlation can result in the identification of individual events and phenomena, as well as the creation of profiles to track these activities even in real-time. Data correlation is also useful in constructing and validating behavioral proxies. For example, demonstrating that the traffic speed is well-correlated with the number of taxi pickups through historical data sources for which the latter are known directly (e.g., through the NYC Taxi & Limousine Commission) would allow accurate measurement of traffic speeds in real-time. More broadly, finding correlation among data sets will allow policy makers to better understand cities, thus, providing better operations and better planning to citizens.

Nevertheless, finding correlations in big data corpuses is a hard problem. Not only the abundance of data makes it impractical to manually determine the correlation between them, their usually large temporal coverage is also a challenge in identifying periods of interest (e.g. an event) at different data resolutions. Considering NYC Open Data [1], with more than 1,500 data sets that have been published and updated since 2009, it would take an immense amount of effort for an analyst just to select a data set that is well-correlated with another data set of interest and combine them together. The challenge still persists even when correlated data sets have been identified. For instance, when a policy adviser wants to study the impact of taxi cabs on 311 complaints, s/he would like to know, e.g., a particular day or week when the two data sets have the highest correlation (if any). It is not only necessary to know whether two data sets are well-correlated, but also when the correlations are the strongest.

Although significant work has been done in finding correlations between data sets, relatively little investigation has been done with adaptive temporal resolution. Most of the work in correlation analysis assume a fixed time resolution for each data set. For example, finding correlation between taxis and weathers will only return well-correlated periods either in hours, in days, or in weeks. However, in practice, it is common to see strong correlations between the two data sets at multiple resolutions ranging from hours (e.g. in raining) to weeks (e.g. during a storm). Additionally, when multiple correlated periods are found, they are often presented equally the same regardless of their significance. Users then have to rank them manually for their analysis.

In this paper, we present a framework for identifying temporal correlations in large data sets with adaptive resolutions. We use mutual information to capture and quantify the co-dependence between variables, thus, we are able to rank the significance of the correlations automatically. This allows us

to suggest "more" interesting temporal periods to users for further investigation. Our method also works in a streaming fashion and minimizes duplicated computations. Finally, our approach has been implemented on Apache Spark to handle the large volume of today's high-demand data sets.

Contributions The main contributions of this paper include:

- We describe a method based on mutual information to identify correlations with adaptive temporal resolutions.
- We add streaming support to a state-of-the-art method in computing mutual information for handling big data sets. We also propose optimizations that minimizes duplicated computations across streaming windows.
- We provide a Spark implementation for scalability.
- We perform a comprehensive evaluation of our techniques on real-world data sets from NYC Open Data.

II. Background

We first review the concept and properties of mutual information (MI) in data analytics and discuss the state of the art estimation method for computing MI values.

A. Mutual Information: Definition and Properties

In information theory, MI has been used as a measure of mutual dependency between variables. It quantifies the amount of information obtained about one variable through the knowledge of other variables [2]. The mutual information, $I(X;Y)$, between two random variables X and Y specifies how much knowledge about X is gained through Y, or how much uncertainty of X is reduced by knowing Y and vice versa. MI is a function of the probability distribution, i.e., the probability density function (pdf) for continuous variables and the probability mass function (pmf) for discrete variables, and can account for both linear and non-linear relationships. Eq. 1 defines the MI of two discrete random variables X and Y:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \quad (1)$$

where $p(x,y)$ is the joint probability of (X,Y), and $p(x)$, $p(y)$ are the marginal probability of X and Y respectively.

From Eq. 1, $p(x,y)$ measures the probability that X and Y are observed together, while $p(x)$ and $p(y)$ are the probability that X and Y separately occur. The fraction $\log(p(x,y)/(p(x)p(y)))$ determines the magnitude of joint occurrence over the individual realization of the variables. The larger is this magnitude, the more likely these two random variables occur together and thus, more likely to be dependent on each other. Intuitively, if the two variables are statistically independent, their MI is zero, meaning that given the knowledge of one variable, it does not reveal anything about the other. On the other hand, if the two variables are statistically dependent, their MI will be greater than zero, and attains larger value as the dependency between the two variables is stronger.

MI owns several properties that make it advantageous when evaluating correlations over other measures such as covariance or the correlation coefficient [3]. First of all, MI equals to zero if and only if the considered variables are statistically independent. Otherwise, it is always greater than zero if the variables hold some kind of dependency. This property makes MI a general metric for measuring correlations and is ideal for noisy data sets, which have a high degree of biases and abnormality, thus, possessing arbitrary and non-linear relationships [4]. Second, MI is invariant under 1-1 transformations, i.e, $I_{XY} = I_{UV}$ if $u = u(x)$ and $v = v(y)$. This property says that under the transformation, if X and Y maintain their distributions, their MI is preserved [3]. This characteristic is well suited for processing spatial-temporal data sets, which are often collected beforehand at different resolutions. For example, in our reference case study, taxi trip records were collected every minute while traffic speeds were recorded every hour.

B. Estimating Mutual Information

Although MI is a powerful measure in discovering relationships among data sets, it is challenging to apply in practice due to the difficulty in estimating probability distributions. Among several estimation methods [5], e.g., histogram, kernel density estimation etc., we choose a popular non-parametric method proposed by Kraskov et al. [6], hereafter called the KSG method, to approximate MI because of the following reasons: (1) This method outperforms other estimators in terms of computational efficiency, accuracy and is especially suitable for long and chaotic time series [7]; (2) The method uses k-nearest neighbor approximation and thus it is data efficient (i.e., it does not require very large samples), adaptive and has minimal bias [6]. This reason makes the method particularly suitable to study spatial-temporal data, where dependency between variables might only occur at specific times or locations.

1) KSG mutual information estimator: The main idea of KSG estimator is that rather than directly compute the joint and marginal probability distributions of considered variables, it estimates the densities of data points in neighborhoods [6]. For each data point, it first searches for k nearest neighbor clusters (k is a pre-defined parameter) and computes the distance d to the k^{th}-neighbor. Then, the population density within distance d is estimated by counting the number of data points that fall inside d. This leads to the computation of MI between X and Y as [6]:

$$I(X;Y) = \psi(k) - 1/k - < \psi(n_x) + \psi(n_y) > + \psi(N) \quad (2)$$

where ψ is the digamma function, k is the number of nearest neighbors, (n_x, n_y) is the number of marginal data points in each dimension falling within the distance d, N is the total number of data points and $< . >$ is the average function.

The intuition behind this estimator is, as MI aims to seek for the knowledge of X based on Y (or vice versa), it looks

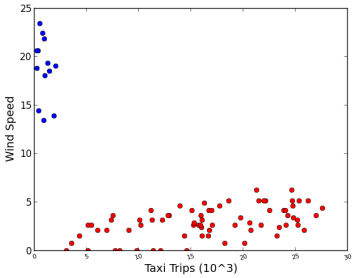


Figure 1. Taxi Trips vs. Wind Speed in normal days (red) and hurricane (blue)

into Y's neighborhood and checks if nearby values y_i result in closely related values x_i. It means that, for a specific data point, if its neighborhood in the (X, Y) space corresponds to similar data, then knowing Y helps predicting X and vice versa, implying a high MI between X and Y. This concept is illustrated in Fig. 1, plotting taxi trips (counted by hour) and wind speed (averaged by hour) in NYC during two different periods: the samples in blue were recorded when hurricane Sandy was happening in NYC from Oct 29^{th} to Oct 30^{th}, 2012; the samples in red were recorded during normal days. Inspecting the blue neighborhood, one can find the high values of wind associated with very low values of taxi, while in the red neighborhood, records are more diverse with low wind values associated with a wider range of taxi values. The blue neighborhood is clearly showing a more apparent pattern, and thus, yielding a higher MI as shown later.

Choosing the value of k***:*** The KSG method requires a free parameter k, i.e., the number of nearest neighbors to be searched for each data point. While there is no theoretical basis for selecting an optimal value for k, in this work, we use our real data sets to tune this parameter until a k value that provides a stable estimation of MI is found. In Section IV, we discuss the result of this tuning process.

III. Searching For Correlations using Adaptive Sliding Window

For time series data, the dependency can either hold for the entire data series, or only hold in certain periods (as the example of taxi trip and wind speed). As we aim to look for correlations between temporal data sets, we develop a search algorithm based on a sliding window method that can adaptively and efficiently compute MI in a incremental manner. The algorithm returns a set of windows of different sizes that show the time durations where considered variables may hold interesting or significant correlations.

A. Using MI to measure correlations in time series

Let $X = \{x_t\}_{t=1}^N$ and $Y = \{y_t\}_{t=1}^N$ be two finite time series of equal length N. The joint time series between them is $(X, Y) = \{(x_t, y_t)\}_{t=1}^N$. Since X and Y are sampled in time, they are either discrete or discretized. Thus the frequencies of combinations of (x, y) pairs can be computed by counting the number of times each pair occurs in the

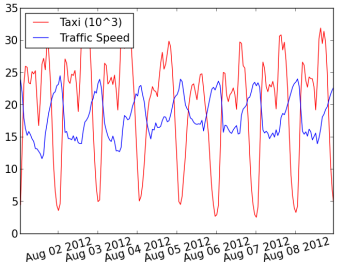


Figure 2. Time series of Taxi Trips and Traffic Speed

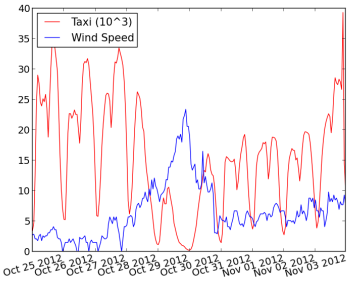


Figure 3. Time series of Taxi Trips and Wind Speed

data, then used to estimate MI. Through estimating the MI of (X, Y), we want to seek answers for the following problems:

Problem 1: Determine if variables X and Y are overall correlated and the strength of their relationship if it exists.

Problem 2: If overall correlation does not hold for X and Y, then search for time windows $w_{ij} = [t_i, t_j]$, $1 \leq i < j \leq N$, where X and Y are highly correlated.

A positive MI between variables over the entire data series indicates that they are correlated in general. Thus, *Problem 1* can be answered by computing MI for the entire (X, Y) data series. The relationship strength is determined by the MI magnitude, where a larger value corresponds a stronger relationship. As an example, considering two variables: the number of taxi trips (X) and the traffic speed (Y) in NYC, shown in Fig. 2 (the data are taken from NYC Open data sets). We might observe that whenever the number of taxi trips x_i is high, traffic speed y_i is low and vice versa, implying a pattern that a high number of taxi might slow the traffic. Thus, if we ever wonder whether these two variables have any correlations, we can take their entire time series data and compute the MI. If it is positive, then there are correlations between these variables. On the other hand, consider taxi trips and wind speed (Fig. 3), there is no clear patterns between these two except for a period from Oct 29^{th} to Oct 30^{th}, where we observe a significant drop in taxi trips associated with an extreme high wind speed. This period is when hurricane Sandy was causing abnormally high winds in NYC. Hence, we may conclude that the correlation between these two might not exist during regular days, but only in extreme events. In this case, MI of the pair (taxi trips, wind speed) might be very low during regular days but significantly high in the time window [Oct 29^{th}, Oct 30^{th}]. Looking for these temporal windows throughout the time series is our objective in answering *Problem 2*.

B. Adaptive sliding window search

Our goal now is to efficiently search for correlations over time series data sets. One possible approach is to compute MI for every possible segment of the data. However, this is computationally expensive and requires a large amount of memory. To address this issue, we adaptively slide windows of different sizes over the data series to iteratively filter data portions that might contain interesting correlations. Next, we describe the terminology used in our algorithm.

Time window contains a set of timestamped data points collected over a continuous time period and are sorted in chronological order.

Window granularity is a temporal unit representing time scale of a window. For example, a window containing data for an entire year has a *one-year* granularity; a window containing data of just 10 days has a *10-day* granularity.

Granularity size represents the number of data points contained in one unit of granularity. This size depends on the data resolution. For example, if granularity units are measured in *days* and the data resolution is in hours, i.e. a sample is collected every hour, the granularity size is 24.

Window size is number of data points contained in a window and is computed by taking the product of *window granularity* and *granularity size*. A window must have proper size, i.e., contain enough samples, in order to report any significant correlations.

Threshold σ is a non-negative real number representing the minimum value of MI. A window with an $MI \geq 0$ is said to contain significant correlations.

The adaptive sliding window search procedure is composed of several layers. At the highest level, the search is performed in a hierarchical top-down manner: it starts with the largest granularity and iteratively reduces the size if significant correlations are not found, thus, creating different window sizes. At each level, windows of the same size are moved along the time series as MI is computed for each of them. The MI computation for consecutive windows is performed incrementally to minimize the computational cost. In the following, we detail each computation step.

1) Top-down search using adaptive window sizes: A main benefit of the top-down search is to minimize the search space. Rather than computing MI for all possible windows, it assumes that data sets are only well-correlate at a certain window size or above. This assumption is often valid, for example when correlation links to weather events which only last for some time. We allow users to define the maximum and minimum granularity they want to perform on the data. Then starting with the maximum granularity, data are partitioned into windows of the same size. Two consecutive windows can be disjoint or overlapped depending on whether correlation exists in one of them (details are described in the next section). MI is computed for each window and is compared against the threshold σ. Windows that hold significant correlations (i.e., MI $\geq \sigma$) are selected. After the first scan, a set of windows which pass the threshold test are returned and another set contains windows that do not pass (called *left out data*) will be used for the next scan, but with a smaller granularity. The granularity decreasing step is also customizable. The search stops when we exhaust all data points or the granularity reaches the minimum size. Clearly, the top-down approach provides the user with flexibility to test the dependency in different time scales. It can test the overall correlation of variables by setting the maximum granularity to cover the entire data series, while decreasing the granularity to a smaller scale helps to uncover correlations in time windows. Algorithm 1 illustrates this hierarchical search.

Algorithm 1 Adaptive top-down search

```
1: Inputs: < x, y >: pair of time series variables
2: Parameters: max_g: maximum granularity, min_g: minimum granularity,
   size_g: granularity size, step: granularity decreasing step
3: current_g = max_g
4: left_out = < x, y >
5: while left_out AND current_g > min_g do
6:     (windows, left_out) = search_windows(left_out, size_g, current_g)
7:     current_g = current_g - step
8: end while
9: return windows
```

2) Sliding window with filtering: This step works at each granularity layer. At each granularity layer, the procedure *search_windows* moves windows of the same size along the data series and filter those that hold significant correlations. We illustrate this movement in Fig. 4.

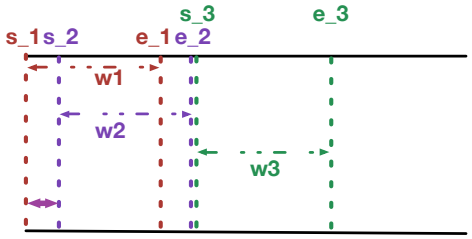


Figure 4. Sliding window search with filtering

Each window w_i is identified by the start index s_i and end index e_i (indicating the first and the last data point of w_i). Since each data point is associated with a timestamp, the start index and end index also indicate the start time and end time of the window.

Initially, the search starts from the leftmost of the data series, and compute MI for the first window $w_1 = [s\_1, e\_1]$. Assuming that $mi_1 < \sigma$, then w_1 does not hold significant correlation and its indices $[s\_1, e\_1]$ are inserted into $left\_out$ list. Next, the search shifts to the right, creating the second window $w_2 = [s\_2, e\_2]$. The shifting step from s_1 to s_2 indicates how far the window is moved every time the previous window does not pass threshold test, and is also user-defined. The MI is computed for data points belonging to w_2. Supposing that this time, w_2 passes the threshold test. In this case, the indices $[s\_2, e\_2]$ are inserted into $windows$

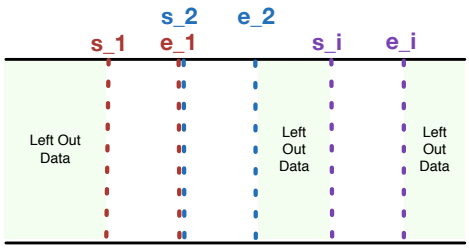


Figure 5. Time series after the first scan

list, and at the same time, the current entry in $left\_out$ list (i.e., $[s\_1, e\_1]$) is updated to $[s\_1, s\_2]$, indicating only the data portion from s_1 to s_2 is left out. Next, the search moves on to the third window $w_3 = [s\_3, e\_3]$ with s_3 is right after e_2 of w_2. The procedure repeats for the rest of the data series.

Fig. 5 illustrates the results after the scanning. A set of non-overlapping windows that pass the threshold test is stored in $windows$ list, and a set of disjoint partitions containing left out data is stored in $left\_out$ list. These left out data partitions will become the inputs for the next scan with reduced granularity.

3) Boxed-assisted algorithm with incremental computation: Up to this point, we have illustrated how sliding window can be applied to the search for correlations in time series, progressively. In this section, we describe how to compute MI for each window in an incremental manner.

As mentioned earlier, the KSG estimator aims to estimate MI based on the neighborhood population. For each data point i, it counts marginal points within k-nearest distance. Different methods can be used to search for k-nearest neighbor (e.g., k-D tree, boxed-assisted, projection method [8]). We choose to use the boxed-assisted method because it outperforms other methods for low dimensional data [8].

In the sliding window algorithm, computing MI for each window by visiting all data points is considerably expensive. Noticing that there could be many overlapping data between consecutive windows, we propose a method that is able to track changes across windows to reuse computation of shared data points. The shift from w_1 to w_2 in the sliding window would result in three different sets of data: (1) data points from s_1 to s_2 in w_1 but are removed in w_2; (2) overlapping data from s_2 to e_1; and (3) newly added data from e_1 to e_2. Our optimized boxed-assisted algorithm can minimize the computation of the old data ($[s\_1, s\_2]$) and the new data ($[e\_1, e\_2]$) in such a way that only the computation for new or affected data has to be performed.

In standard boxed-assisted algorithm, the search space is divided into equal size boxes. Each data point is projected into exactly one box. Each box maintains a list of points within its bounds. When searching for k-nearest neighbors of point i, the box containing point i is identified first. The search then extends to the neighborhood from the reference box until k nearest points are found. Next, the distances (in each dimension) to the k^{th}-neighbor are determined and the marginal points are computed by counting the number of points within these distances. In addition to the box-array structure in the standard version, we employ a supplementary storage to keep track of previous computation results. For each data point i, we add 3 components to its data structure to store: (1) the index of the k^{th}-nearest neighbor, (2) the distances in each dimension to its k^{th}-nearest neighbor, (3) the number of marginal points in each dimension. To track changes caused by old and new data, we define an influenced region for each data point i. An *influenced region* of point i is a rectangular bounding box $R_i = (l_i, r_i, b_i, t_i)$ where l_i, r_i, b_i, t_i are its left, right, bottom, top-most indices and are computed by using indices of the box where point i is located plus/minus $(+/-)$ the distance d to its k^{th}-neighbor. The *influenced region* maintains an area where any point j either falling into or being removed from this region will potentially affect point i. The effect can be either changing its k^{th}-neighbor or its marginal counts. In this case, point i requires a re-evaluation.

With the introduction of influenced regions, the computation for each data point i is enhanced as follows. If i is:

- A new point: (Step 1.1) Follow the standard algorithm to compute its marginal points. (Step 1.2) Re-evaluate every point j whose influenced region contains i.
- An old (removed) point: (Step 2.1) Remove point i and its corresponding data structures. (Step 2.2) Re-evaluate every point j whose influenced region contains i.
- In the overlapping region, no computation is required.

As the result of this incremental computation method, for each window a minimum searching region is determined containing only new points and a minimum updating region containing only points affected by adding new points and by removing old points.

Space complexity: The space complexity of standard boxed-assisted algorithm is $O(n)$ where n is number of samples in a window. With m additional data structures used in the optimized version, the space complexity is $O(mn)$, thus linear in the data size.

Time complexity: With a standard boxed-assisted algorithm, the time to compute MI for each window is $O(n \log(n))$ where n is the window size. The sliding window approach with w windows results in the complexity of $O(wn \log(n))$. With the incremental computation, however, only the computation and updates for new and affected points, respectively, are needed. If the data are sparse, i.e, only few are affected through insertions or removals of points, the time complexity would be much smaller.

Setting the threshold σ We provide users the flexibility to specify the threshold σ, depending on the requirement of relationship strength. The value of σ decides how many windows will show up in the results. Usually, a larger value of σ results in fewer windows. One way to set σ is to base on data coverage. Data coverage represents the amount of data covered in the selected windows, and is computed as:

$$\textit{Data Coverage} = \frac{\#samples\_in\_selected\_windows}{\#total\_samples} \quad (3)$$

Data coverage can be interpreted as the percentage of interesting data points over the entire data series. Data coverage of 20% implies that we want to look for significant correlations in 20% of the entire data series.

C. Exploiting parallelism

To accelerate the search process, we exploit the parallelism of big data platform using Apache Spark. We partition input data into overlapping chunks, where each is processed as a separate process in the Spark cluster. The overlapping portion ensures that the search is continuous between chunks and equals to the shifting step of two consecutive windows.

Algorithm 2 illustrates the parallel procedure. In line 2, data indices are partitioned into overlapped chunks. Each chunk contains w windows and two consecutive chunks are overlapped by a shifting step. Line 3 maps each partition into a node in the Spark cluster and the top-down search procedure is applied to it (lines 5-7).

Algorithm 2 Paralellise Top Down Search

```
1: Parameters: shift: shifting step, size: window size, w: number of windows in
   each chunk, N: total number of samples
2: rdd = sc.parallelize([(i-shift,i+w*size) for i in xrange(0,N,w*size)])
3: rdd.mapPartitions(compute_MI)
4:
5: def compute_MI(chunks):
6:       for chunk in chunks:
7:            top_down_search(chunk)
```

IV. Experimental Evaluation

We evaluate our method using NYC Open Data Sets [1]. For its effectiveness and performance, we assess the correlation of the extracted windows, and the scalability of running against different data sizes, respectively.

A. Data and Infrastructure

The NYC Open Data Sets: include spatial-temporal data containing information related to different activities which took place in New York City. Table I shows the summary of data sets. The data preprocessing and analysis were conducted using the data facility at our research center with a 1200-core cluster running Cloudera Data Hub 5.4 and Apache Spark 1.6. The cluster consists of 20 high-end nodes, each with 64 cores, 256GB of RAM, and 24TB of storage.

B. Data Cleaning and Preprocessing

Data cleaning aims to remove entries with no reported value and duplicated entries. The missing value can be in the form of $null$ (empty) or a default value. For example, weather data use default values (e.g., 9999 or 9999.99) to record missing entries while the missing values in the taxi data is empty. The duplicate entries are removed by checking entry ids to ensure the uniqueness of reported information.

After the cleaning process, the preprocessing step takes place to put data into the form of variables that we are interested in. This is done by using available metadata associated with each data set. For example, the taxi data set consists of 22 attributes, however, we are interested in the number of taxi trips, which is not an attribute. Thus, we compute this variable by counting the number of taxi trips under different time scales, e.g., minute, hour, day, week etc. To deal with the large volume of raw data, we perform the preprocessing process using our Spark cluster. It took approximately 45 mins to process the taxi data set and less than 10 mins to process other data sets.

C. Design of Experiments

Variables and Resolutions: From the data sets in Table I, we extract variables listed in Table II, considering various time resolutions (D: Day, H: Hour, 15M: 15 Minutes).

Parameters setting: Before applying the search algorithm, we need to assign values to several parameters. We report the selection of their values as follows.

- Value of k: we use different k to compute MI for different pairs of variables, e.g., k is in the range from 1 to 20 with step 1. We found that with our data sets, $k = 6$ gives the most stable results comparing to others. The MI changes dramatically between $k = 1$ to $k = 4$. Then it is more stable with k in the range from 5 to 10. Thus, we selected $k = 6$ for our search method.
- σ threshold: this parameter ties up closely with the nature of the relationship between variables. For those that are naturally correlated, higher or lower σ results in lower or higher data coverage. For those that are naturally not correlated, even low σ still results in very low coverage. Though it is not always the case, a σ resulting in 20% data coverage is used for the majority pairs of variables in this work.
- Window granularity: to test the overall correlation, we set the granularity to $year$ to cover the entire data series. To search for time windows, we start with the *month* granularity, then split into day and then $hour$.

D. Running Experiments

Experiments Descriptions: We apply our search algorithm for each pair of variables that we are interested in. We use the same resolution for variables of the same pair. The search is performed on the Spark cluster. Our Spark's application is written in Python, however, the MI computation code is written in C for performance reason. In order to communicate between the application and the MI computation in worker nodes, we use the Spark's *pipe()* command.

Performance and Scaling: We report here the execution times of our method comparing to a brute force version on different data sizes. A brute force search will compute the MI for every window without using incremental computation. To have a fair comparison, we only report the results in which the two algorithms run on one node. We do not compare the parallel implementation as parallelism

Table I
SUMMARY OF NYC OPEN DATA SETS

Data sets	Size	#Attributes	#Records	Time Duration	Description
Taxi Trips	115 GB	22	900 M	2009-2014	Information of taxi trips in the city
Traffic Speed	18 GB	4	400 M	2009-2012	Record the speed of vehicles at certain location and time in NYC
Collisions	49 MB	22	400 K	2012-2014	Record the incidents happened in NYC
Weather	318 MB	1008	65 K	2010-2014	Contain information of weather condition in the city with more than 1000 attributes.
311 Complaints	600 MB	10	80 M	2003-2014	Include direct complaints from citizens from 311 telephone.

Table II
VARIABLES AND RESOLUTIONS

Variables	Resolutions	Description
#Taxi Trips	D, H, 15M	Number of taxi trips
Average Taxi Fare	D, H, 15M	Average fare of taxi trip
Average Trip Duration	D, H, 15M	Average duration of taxi trip
Average Wind Speed	D, H, 15M	Average speed of wind
Average Rain Precip.	D, H, 15M	Average amount of rainfall
Average Visibility	D, H, 15M	Average distance of visibility
Average Traffic Speed	D, H, 15M	Average speed of traffic
# Injured Motorists	D, H	Number of injured motorists
# Killed Motorists	D, H	Number of killed motorists
# Injured Cyclists	D, H	Number of injured cyclists
# Killed Cyclists	D, H	Number of killed cyclists
# Injured Pedestrians	D, H	Number of injured pedestrians
# Killed Pedestrians	D, H	Number of killed pedestrians
# Collisions	D, H	Number of collisions
# 311 Complaints	D, H	Number of complaints

Table III
EXECUTION TIMES OF OPTIMIZED SLIDING WINDOW ALGORITHM AND BRUTE FORCE ALGORITHM

Data Size (#Samples)	Brute Force	Optimized SW
1000	3.97 sec	3.15 sec
4000	32 sec	16 sec
8000	95 sec	40 sec
12000	4 mins 9 sec	2 min 3 sec
16000	11 mins 32 sec	6 mins 25 sec
18000	13 mins 35 sec	7 mins 16 sec
20000	15 mins 34 sec	8 mins 45 sec
25000	20 mins 52 sec	10 mins 51 sec
31000	27 mins 50 sec	12 mins 2 sec

only divides data into smaller sizes. As shown in Table III, our implementation can achieve a speedup of 2 with the optimized version, and the larger the data size, the larger the speedup. Note that the speedup will change depending on the data distribution. For example, if the data series are well-correlated, the search will stop earlier while if the data are not well-correlated, the search will need longer to search for significant windows.

E. Summary of Correlation Findings

We report our findings when searching for correlations in the NYC Open data sets. Due to space limitation, we will plot only a few extracted windows when explaining our findings. We compare our method and findings to those found in [9]. The work in [9] uses a topology-based approach to represent and identify relationships among the data sets.

1) Weather and Taxi: The findings listed here come from variables extracted from *Weather* and *Taxi* data sets.

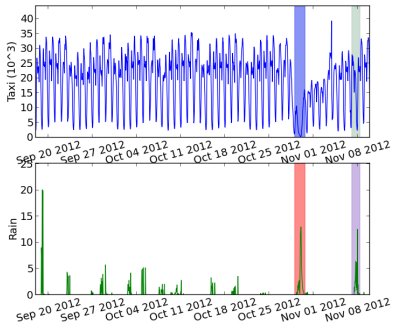


Figure 6. Taxi Trips vs. Rain Precipitation

Taxi Trips and Rain Precipitation: We search for correlation between the number of taxi trips and average rain precipitation in an hour resolution. In certain extracted windows, we found a negative relation between two variables, shown in Fig. 6. Two highlighted windows show a drop in the taxi trips associated with abnormally high rain. To better understand the windows, we look into their data. The first window last for two days, from 29th Oct 2012 to 30th Oct 2012, the period when hurricane Sandy approached NYC. In the second window, however, the drop of the taxi trips happens at midnight. Noticing that taxi trips have a daily pattern, showing a high number of trips during rush hours (7AM-9AM, 1PM-3PM, 6PM-8PM) and a low number of trips in non-rush hours, especially at midnight. Thus, the association between high rain and low taxi trips seems likely to be coincident in the second window.

Moreover, it should also be noted that there is a short period on the far left of Fig. 6, where an abnormally high rain is also linked with a drop of the taxi trips. This window, however, is not shown in our search results. We found that the abnormal increase in precipitation occurred in a very short time period. Only 7 data samples were reported in this period, which were not enough to hold significant correlation and, thus, is not captured by our search.

The findings in [9] report a strong negative relationship between the two variables considered here: ”When rain precipitation is high, number of taxi trips is low, implies the difficulty to find a taxi in rainy days”. In our findings however, it appears that this pattern only holds in very extreme events, such as during a hurricane, in which many other factors may combine together with the measured

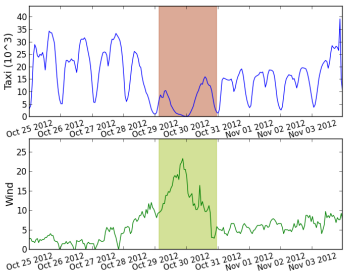


Figure 7. Taxi Trips vs. Wind Speed

Table IV
TOP RANKINGS WINDOWS BETWEEN TAXI AND WIND SPEED

From	To	MI	Event
2012-Oct-29	2012-Nov-02	0,651174	Sandy Hurricane
2012-Jul-27	2012-Jul-28	0,604444	Tornado hits NYC
2012-Jan-20	2012-Jan-21	0,578166	Snow storms
2012-Nov-10	2012-Nov-11	0,542242	Snow storms
2012-Dec-23	2012-Dec-24	0,493842	Storms
2012-Nov-06	2012-Nov-07	0,487898	Snow storms
2012-Aug-28	2012-Aug-29	0,471387	Hurricane Irene
2012-Sep-08	2012-Sep-09	0,424399	Tornado
2012-Sep-19	2012-Sep-20	0,420836	Storms

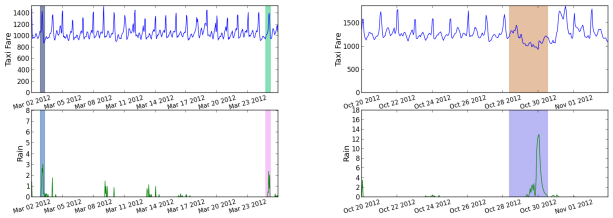
rainfall to cause a drop in the taxi trips.

Taxi Trips and Wind Speed: We test the pair of (taxi trips, wind speed) variables, in both an hour and 15 mins resolutions. We found a negative relationship between these two in both resolutions. When the wind speed readings were exceptionally high, major drops in taxi trips occurred (e.g. Fig. 7). Our finding is similar to the results reported in [9] indicating a negative relationship between the taxi trips and the average wind speed.

However, the discovered relationship is also linked with very extreme events. To test our hypothesis, we performed further analysis by ranking extracted windows according to their MI magnitude and select the top windows. As expected, many of these top windows were associated with extreme events that happened in NYC. Table IV reports these events information, which we found through news feed, along with each highly ranked time period.

Taxi Fare and Rain Precipitation: A finding reported in [9] suggests a positive relationship between Taxi Fare and Rain Precipitation (taxi drivers increase earnings when it rains). The phenomenon was explained due to the fact that taxi drivers were the target earners. However, we experienced different results. We found a window where the two are negatively related, shown in Fig. 8(b), whose time coincides with the hurricane Sandy. Whereas in other periods, as for example in Fig. 8(a), when the two windows (highlighted) are found at times when the rain was higher than normal, but the taxi fare was still relatively similar to other periods.

Taxi Trips and Visibility: We found a correlation between the number of taxi trips and visibility. During periods when



(a) (b)

Figure 8. Taxi Fare vs. Rain Precipitation

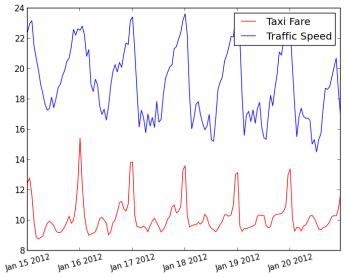


Figure 9. Daily Patterns of Taxi Fare and Traffic Speed

the visibility is low, there is a shortage of taxi trips. This may be due to the increased danger of driving under this condition. This result is similar to the finding in [9]. In addition, our findings also suggests that this phenomenon mostly happens at midnight, when we think could be due to other factors beside visibility, causing a shortage of taxis.

2) Taxi and Traffic Speed: We analyze variables extracted from the *Taxi* and *Traffic Speed* data sets. We compute their MI using the entire data series of the year 2012 in an hour resolution. We found a dependency between taxi trips and traffic speed with a significantly high MI. We then split the data into daily windows, and compute the MI for each day. The results also yield a high MI, implying that the variables might correlate with a daily pattern. This confirms the finding in [9], which also reports a strong relationship between the two variables, even in regular days.

We test another relationship between the taxi fare and the traffic speed in an hour resolution. We obtain a relatively high MI for the entire data series in 2012. This implies a correlation between the two variables. Particularly, when we split the yearly data into daily pattern, we gain even higher MI, implying the correlation is stronger in a daily pattern. Fig. 9 shows the pattern of this correlation, where the traffic speed and the taxi fare were both low and high at similar times. This suggests that taxi drivers are likely to earn less with a slow traffic, similar to the findings in [9].

3) Collisions and Weather: We analyze the *Collisions* and *Weather* data sets. One of the findings in [9] reports a strong positive relationship between rainfall and the number of motorists killed, as well as the number of injured pedestrians. However, we could not find these correlations using our method. The MI values between these pairs of variables

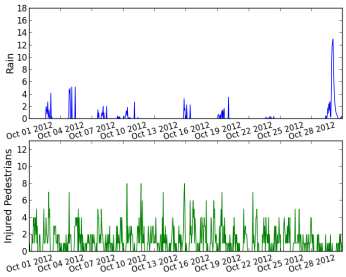


Figure 10. Rain vs. Injured Pedestrian

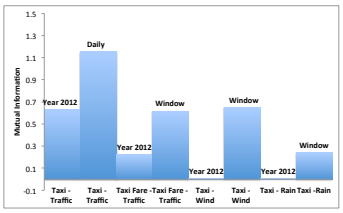


Figure 11. Strength of Different Relationships

are almost zero, and we could not extract any significant windows from the data series, even with a low σ, across all resolutions. We made further investigations by manually inspecting the results, e.g. the plots in Fig. 10. We were not able to confirm a clear pattern between these variables.

4) Collisions and Taxi: We study variables extracted from the *Collisions* and *Taxi* data sets. Similar to the previous experiment, our method did not find a correlation between the taxi trips and the number of collisions, where [9] reports a strong positive relationship.

5) Collisions and 311: We test the relationship between number of collisions and number of 311 complains. A finding in [9] suggests that there is a strong positive relationship between these two variables. However, we could not find this correlation using our method.

F. Verify the Strength of the Relationships

We verify the relationship strength between different pairs of variables by comparing their MI magnitude. The larger its magnitude, the stronger the relationship. Fig. 11 shows some of them. For taxi trips and traffic speed, the MI is high using entire data series of the year 2012 and significantly high using daily data. Similarly for taxi fare and traffic speed. Whereas for taxi trips and wind speed, or taxi trips and rain precipitation, the overall MIs are almost zero but significantly higher in time windows.

V. Related Work

Understanding the dependency between data sets is one of the most desirable things one has when dealing with data. Such understanding can help forecast trends, make predictions and uncover root causes of certain phenomena. However, this is a difficult problem. In the past, many research works (e.g., [10], [11], [12]) have used traditional statistical metrics such as covariance, correlation coefficients (Pearson, Spearman) to identify correlations among data. These metrics, however, are limited to linear dependency. As data sets today are growing larger, more diverse and complex, the problem becomes more challenging and requires new techniques or approaches to address. Recent works such as [13], [14], [15], [16] attempt to approach the problem from high level. Sarma et al. [13] define the concept of relatedness (based on the results of queries) to find related tables in databases. In [14], Pochampally et al. propose to model correlations between different data sources using *joint precision* and *joint recall* as indicators. Whereas the work in [15] relies on the history and schema of data sets to map and link them together. In [16], Roy et al. use the concept of intervention (i.e, changes in the values of inputs affect the outputs) to look for causal explanation for the answers of SQL queries. These works differ from our approach as we aim to look for relationships not only between data sets but also the time windows where the data are well-correlated.

Recently, Chirigati et al. [9] proposes a topology-based framework to identify relationships between spatial-temporal data sets. The notion of topological features, whose interestingness is captured by critical points (maximum and minimum), is defined to represent the data sets and identify relationships between them. The technique is able to find both regular relationships and relationships at extreme events. Our work aims to achieve similar goals, but at the same time, trying to pick out the exact time windows where the correlations occur. Our method should be used in conjunction with their work to create a comprehensive framework in data exploratory. Indeed, in this paper, we closely compare our findings to those in [9]. There are some differences in the findings between the two methods, thus, a deeper analysis in the future could be helpful to uncover those differences and better explain the results.

Regarding the use of mutual information, the metric has been broadly used in many domains to achieve different goals, e.g., features selection ([17], [18]), clustering and mining ([19], [20]), network inference and construction ([21], [22]). In term of searching for correlations, the work in [23] has been used time-delayed mutual information to interpret temporal correlations in glucose measurements time series data. Chen et al. [24] use mutual information to discover spatial temporal dynamic of the magnetosphere during geospace storms. However, these works do not consider the context of big and diverse data sets where correlations can hold in entire data sets or only in specific time durations.

In the context of big data, the use of mutual information is relatively new. A recent work of Su et al. [25] proposes a framework and a set of algorithms to analyze relationships of massive scientific data sets. They also use mutual information as one of the metrics to measure correlations between queries. However, they only consider overall correlations and focus the work on data indexing to efficiently compute correlation in parallel and distributed setting.

In the context of data streaming, in another recent work [26] Keller et al. propose an algorithm to estimate MI for data streams. It also uses the KSG method and proposes a data structure called query anchor to keep track of marginal counts. A main difference in our work is that we use a top down approach to minimize the search space, while at the same making sure to capture significant correlations. In particular, we introduce influenced regions to keep track of changes in the data, a technique that is not used in [26].

VI. Conclusion

In this paper we present a method based on information theory to study the temporal relationships in the context of big data sets. With the presented results, we believe that the approach opens a new and important perspective on how we should deal with big data. In an era where data are massive, diverse and complex, a wise treatment for it is not to process data blindly but to select the most interesting and informative data portions. By reducing the amount of data that potentially are not informative, different mining and learning algorithms can be beneficial in terms of computation efficiency and accuracy.

In the future, we intend to expand our work to capture the spatial components and discovering relationships across more than two variables. We also plan to perform a deeper analysis of our results and [9] to validate the different findings between the two methods.

VII. Acknowledgements

We would like to thank Prof. Barbara Pernici for providing insightful comments and discussions throughout the work. We also thank the authors of [9] for making their data available and sharing valuable insights with us. This work was supported in part by a CUNY IRG Award.

References

[1] Nyc open data. [Online]. Available: https://nycopendata.socrata.com

[2] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[3] D. R. Brillinger, "Some data analyses using mutual information," *Brazilian Journal of Probability and Statistics*, pp. 163–182, 2004.

[4] S. de Siqueira Santos, D. Y. Takahashi, A. Nakata, and A. Fujita, "A comparative study of statistical methods used to identify dependencies between gene expression signals," *Briefings in bioinformatics*, p. bbt051, 2013.

[5] L. Paninski, "Estimation of entropy and mutual information," *Neural computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

[6] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, 2004.

[7] A. Papana and D. Kugiumtzis, "Evaluation of mutual information estimators for time series," *International Journal of Bifurcation and Chaos*, vol. 19, no. 12, pp. 4197–4215, 2009.

[8] M. Vejmelka and K. Hlaváčková-Schindler, "Mutual information estimation in higher dimensions: A speed-up of a k-nearest neighbor based estimator," in *ICANNGA Proc.*

[9] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire, "Data polygamy: the many-many relationships among urban spatio-temporal data sets," in *SIGMOD Proc.*, 2016.

[10] W. E. Dean Jr and R. Y. Anderson, "Application of some correlation coefficient techniques to time-series analysis," *Journal of the International Association for Mathematical Geology*, vol. 6, no. 4, pp. 363–372, 1974.

[11] H.-C. Huang, S. Zheng, and Z. Zhao, "Application of pearson correlation coefficient (pcc) and kolmogorov-smirnov distance (ksd) metrics to identify disease-specific biomarker genes," *BMC Bioinformatics*, vol. 11, no. 4, p. 1, 2010.

[12] G. Chamberlain, "Analysis of covariance with qualitative data," 1979.

[13] A. Das Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu, "Finding related tables," in *SIGMOD Proc.*, 2012, pp. 817–828.

[14] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava, "Fusing data with correlations," in *SIGMOD Proc.*, 2014.

[15] A. Alawini, D. Maier, K. Tufte, and B. Howe, "Helping scientists reconnect their datasets," in *SSDBM Proc.*, 2014.

[16] S. Roy and D. Suciu, "A formal approach to finding explanations for database queries," in *SIGMOD Proc.*, 2014.

[17] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[18] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Trans. on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[19] N. Slonim, G. S. Atwal, G. Tkačik, and W. Bialek, "Information-based clustering," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 51, pp. 18 297–18 302, 2005.

[20] Y. Ke, J. Cheng, and W. Ng, "An information-theoretic approach to quantitative association rule mining," *Knowledge and Information Systems*, vol. 16, no. 2, pp. 213–244, 2008.

[21] P. E. Meyer, K. Kontos, F. Lafitte, and G. Bontempi, "Information-theoretic inference of large transcriptional regulatory networks," *EURASIP journal on bioinformatics and systems biology*, vol. 2007, no. 1, pp. 1–9, 2007.

[22] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano, "Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC bioinformatics*, vol. 7, no. Suppl 1, p. S7, 2006.

[23] D. J. Albers and G. Hripcsak, "Using time-delayed mutual information to discover and interpret temporal correlation structure in complex populations," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 1.

[24] J. Chen, A. Sharma, J. Edwards, X. Shao, and Y. Kamide, "Spatiotemporal dynamics of the magnetosphere during geospace storms: Mutual information analysis," *Journal of Geophysical Research: Space Physics*, vol. 113, no. A5, 2008.

[25] Y. Su, G. Agrawal, J. Woodring, A. Biswas, and H.-W. Shen, "Supporting correlation analysis on scientific datasets in parallel and distributed settings," in *HPDC Proc.*, 2014.

[26] F. Keller, E. Müller, and K. Böhm, "Estimating mutual information on data streams," in *SSDBM Proc.*, 2015.