Introduction to Time Series Search Engine Systems

Chuanlei Zhang
Department of Electrical & Computer Engineering
Ryerson University

Ji'an Luo
State Key Lab. of Industrial Control Tech.,
Zhejiang University,
Zhejiang, China 310027

Shanwen Zhang
Department of Engineering and Technology，Zhengzhou University, China

Chen Zhang
Discover Information Technology Limited
Shanghai , China

***Abstract*—Basic structure of time series search engine system, involving acquisition, data processing and retrieval sub-systems, was elaborated. And options for time series search engine architectures and supported features and interfaces along with the associated tradeoffs for system resource utilization were presented.**

Keywords-time series; search engine; similarity search; data mining

I. INTRODUCTION

Web search activity has previously been shown useful for providing estimates of real-world activity in a variety of contexts, with the most common being health and economics. Examples in health include influenza, acute diarrhea, chickenpox, listeria, and salmonella. Examples in economics include movie box office sales, computer game sales, music billboard ranking, general retail sales, automotive sales, home sales, travel, investor attention11, and initial claims for unemployment [1]. Google recently (June 2011) launched a new labs tool called Google Correlate. Given some data (either via a search term you enter or from data you upload), the system ranks all queries in terms of how well they correlate to the object data – time series. A time series (or time sequence) is a sequence of real numbers, each number representing a value for an interested attribute at a time point. Typical examples include stock prices or currency exchange rates, the volume of product sales, biomedical measurements, weather data, etc., collected over time. Hence, time series database systems supporting fast retrieval of time series data and similarity queries are desired. For example, for time series in finance, time series retrieval can help to identify "pair trading," the goal is to identify pairs (or groups) of stocks whose prices track one another (their difference is a stationary process), it can also help to find correlation of stocks over certain time interval, and to find patterns in temporal data for data mining [2].

A lot of papers have introduced new algorithms and applications on time series data mining [2]-[7]. These issues on fast retrieval of time series data and similarity queries are discussed a lot. However, few of them are actually focused on the time series search engine systems and their compositions. This paper will focuses on the introduction of time series search engine system and description of its each component.

The rest of the paper is organized as follows. Section II introduces the basic concept and architecture of TSSES (Time Series Search Engine System). Section III describes several commonly used ways to acquire time series data. The time series data processing is discussed in details in Section IV. In Section V, we discuss a number of ways for user to query time series data. Section VI describes how a user interacts with a time series search engine. Section VII lists the factors concerning scalability of a time series search engine system. Section VIII describes several commonly used ways to retrieve time series data. Section IX gives typical features of a time series search engine system. Section X makes a brief conclusion.

II. ARCHITECTURE

Most search engine systems share a common architecture at a high level, but vary widely depending on the application and design choices. Generally speaking, there are three main architectural components as we view the system from a time series data flow perspective: data acquisition, data processing (indexing), and retrieval (query processing) (see Figure. 1). In practice, these are typically decoupled independent processes in order to make scaling easy. We will also consider the system from a user activity perspective in which we can consider behaviors and system states [8].

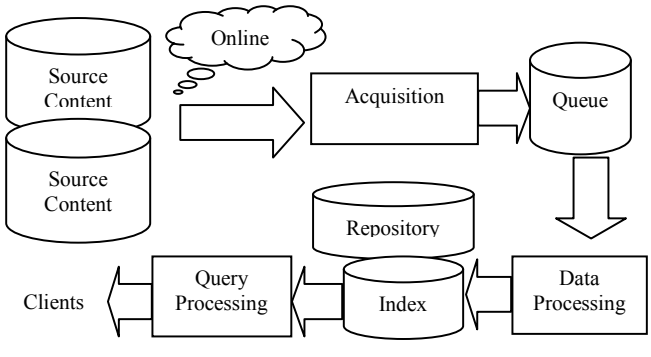

Figure 1. High level architecture of a typical TSSES.

Data acquisition refers to bringing source time series data into the storage for subsequent indexing. This may involve user generated data and online time series data service.

Time series data processing is the next logical stage in the time series flow and involves data pre-processing, feature extraction and indexing methods. The goal is to capture the

978-1-4673-0199-2/12/$31.00 ©2012 IEEE

feature structure in data structures that enable rapid retrieval and time series adaptation.

The third major functional block from a time series data flow perspective is retrieval where a query engine responds to user requests in a real-time interactive mode. The results are exposed through one or more user interfaces and search results summaries or contextual information may be generated to improve the user experience. In addition to real-time query handling, modules for personalization or data ruining can operate on the stored time series collection in an offline fashion to produce customized views or analytical results for users.

III. DATA ACQUISITION

A time series signal is a set of observations of a variable at successive times. They can be found in applications such as stock market price analysis/prediction, electrocardiogram (ECG) data analysis, weather forecasting, as well as a lot of other industrial process-monitoring systems [9]. From a time series data source perspective, these data can be from user created/collected time series, or from internet download.

A. User Created/Collected

For example, in an engine health monitoring system, indicators of the health of an operating engine will be embedded in patterns of vibration and performance data which are represented as time series. An aero-engine on a civil airliner is capable of generating 1GB of vibration and performance data per engine per flight. A crucial part of engine health monitoring is the detection of earliest possible signs of the presence of features known to be associated with fault conditions and for deviation from a model of the known operation of the engine. In order to find such pattern, collecting and searching over terabytes of data is necessary [9].

B. Online Download

With the accumulation of data on Internet, there is a great amount of on line time series data for users to search and download. For example, a wealth of free data is available on the Web at financial portals such as Yahoo! Finance and Google Finance. These portals source their data directly from the exchanges and from third-party vendors, many whom serve financial professionals. Yahoo! Finance provides downloadable historical finance data for U.S. and international equities, indices, and exchange trade funds (ETFs). You can also find accounting data, analyst estimates, and ownership information for public companies. Other online portals tend to focus on a specific asset class or securities type, such as the Web sites Barchart.com and INO.com that specialize in the futures markets. The wide availability of this information creates opportunities to introduce analysis techniques that are new to the time series data mining [10].

C. Stream Data

An important online time series data source is stream data. Stream data refers to the data that flows into and out of the system like streams. Stream data is usually in vast volume, changing dynamically, possibly infinite, and containing multi-dimensional features. Typical examples of such data include minute price and volume of a specific stock, computer network information flow, web click streams, and satellite data flow. Such data cannot be coped with by traditional database systems, and moreover, most systems may only be able to read a data stream once in sequential order. This poses great challenges on effective mining of stream data [11].

IV. DATA PROCESSING

A. Data Cleaning

Once data has been obtained, it needs to be cleaned up: missing values must be handled in a consistent way such as eliminating incomplete records, manually filling them in, entering a constant for each missing value, or estimating a value. Other data records may be complete but wrong (noisy). These noisy elements must be coped with in a consistent way [11].

1) *Missing data*: A problem that quite often complicates time series analysis is missing data. There are several reasons for this, such as malfunctioning equipment, human error, and environmental conditions. The specific handling of missing data depends on the specific application and the amount and type of missing data. Usually, there are two approaches to deal with missing data:

a) Not filling: For example, in the similarity computation, we can use a method that computes the similarities of two time series by comparing their local slopes. If a segment of data is missing in one of the two time series, we simply ignore that piece in our similarity computation.

b) Filling with an estimate: For example, in the case of a time series and for small numbers of contiguous missing values, we can use data interpolation to create an estimate of the missing values, using adjacent values as a form of imputation. The larger the distance between adjacent values used to estimate the intermediate missing values, the larger the interpolation error. The allowable interpolation error and, therefore, the interpolation distance vary from application to application. The simplest type of interpolation that gives reasonable results is linear interpolation.

The most common ways of interpolation are linear interpolation, spline interpolation and cubic interpolation. Spline interpolation gives superior results to linear interpolation without being too computationally expensive. It uses low order polynomials to approximate the missing intervals.

2) *Remove noise:* Noise is defined as random error that happens in data mining process. It can be caused by a lot of factors, such as faulty measurement equipment and environmental factors. There are two methods of dealing with noise in data mining: binning and moving average smoothing.

In binning, the data are divided into buckets or bins of equal size. Then the data are smoothed by using the mean, the median, or the boundaries of the bin. Moving average smoothing is used frequently in finance to smooth out short term fluctuations in stock prices.

B. Data Normalization

In data normalization, the data are scaled so that they fall within a specified range, such as [0-1]. Normalization allows data to be transformed to the same "scale" and, therefore, allows direct comparisons among their values. We will examine two ways to do normalization [11]:

Min-max normalization: To do this type of normalization, we need to know the minimum (x_{min}) and the maximum (x_{max}) of the data:

$$x_{norm} = x - x_{min} / x_{max} - x_{min} \quad (1)$$

Z-score normalization: Here, the mean and the standard deviation of the data are used to normalize them:

$$x_{norm} = x - \mu / \sigma \quad (2)$$

Z-score normalization is useful, in cases of outliers in the data, that is, data points with extremely low or highest values that are not representative of the data and could not be due to measure error.

As we can see from above, both types of normalization preserve the shape of the original time series data, but Z-score normalization keeps the shape more closely.

C. Feature Extraction

One of the big challenges of mining time series data is their size and dimensionality. It is usually not enough to look at each point in time sequentially. And also we have to deal with sliding windows of a possibly multi-dimensional time series. By enlarging the context window, this quickly produces very high dimensional vectors, introducing the curse of dimensionality and causing problems with distance metrics. The large amount of data also heavily values of a time series are usually not independent but highly correlated, thus there is a lot of redundancy. Feature extraction should be applied to compress the time series, keeping only the important information while discarding noise and removing correlations. If the right features are extracted, data mining algorithms will not only be speeded up, they will produce better results than on the original data. Especially clustering algorithms will profit from the data cleaning. A reduction down to a few features also increases the interpretability of the results. When using rule generation algorithms, fewer dimensions will produce more understandable rules. Having humans understand, what data mining algorithms find, is the ultimate goal of knowledge discovery, after all [12].

Popular feature extraction techniques for time series include the Discrete Wavelet Transform (DWT) and the Discrete Fourier Transform (DFT). The signal is projected into the frequency domain (DFT) or a tiling of the time frequency plane (DWT). By keeping only the first few coefficients as features, each time series is represented by a rough sketch, because these coefficients correspond to the low frequencies of the signal. But using the first few coefficients is not the best way to compress time series data. Using the largest coefficients instead, will preserve the optimal amount of energy present in the original signal [12]. When dealing with several times series, however, this introduces higher storage demands and/or bookkeeping overhead for the distance computations.

V. Retrieval

After the feature data is extracted and put into the repository ready for indexing, user can retrieve the desired time series sequence with specific similarity measurement. The most common operation of mining time series is searching for similar patterns from time series datasets (similarity searching). The basic form of similarity searching over time series is subsequence matching which assumes that the length of a main series is much longer than that of the query and the query is used to scan through the whole length of the main series. Based on a pre-defined rule, the measure of each match (the score of matching) in a similarity searching provides essential information for the subsequent operation such as classification or prediction [8].

After representing each sequence in a suitable form, it is necessary to define a similarity measure between sequences, in order to determine if they match. An event is then identified when there are a significant number of similar sequences in the database.

An important issue in measuring similarity between two sequences is the ability to deal with outlying points, noise in the data, amplitude differences (scaling problems) and the existence of gaps and other time axis distortion problems. As such, the similarity measure needs to be sufficiently flexible to be applied to sequences with different lengths, noise levels, and time scales; For instance, Euclidean distance may work well on searching for an identical shape (there is no amplitude and time scaling on the query and main series). Correlation is a well known measure for matching signal with scaling on amplitude. For signal with time variation/scaling, dynamic time warping may be a better choice. Hence, there is no single measure that fits all. The use of a measure is the balance of accuracy, performance and many other conditions applied by an application.

In some system, both the amplitude and shape of the engine data are important. In a timer series search engine system, it is often designed to support both correlation and Euclidean distance measures which can be selected by a domain expert before the searching is invoked.

The Extensible Markup Language (XML) is well suited for representing the search results, along with any metadata that accompanies the source asset and any additional metadata that is added over the life of the asset while in the repository. Traditional database systems are optimized to respond to queries on multiple fields and return best matches. A match is well defined, may be extended to include a time range or scope constraints search.

VI. User Perspective

We need to know how a user interacts with a time series search engine. We can model this as a set of behavioral states as shown in Figure.2. The states are as follows [8]:

- Q: Query - the user is presented with a query interface and must formulate or express the query to the system.
- B: Browse - the service presents a set of rank ordered results in response to the query. Metadata and raw data are displayed as a list and the user can interact via scrolling, paging, etc.
- V: View - the user has selected a particular item and the system is ready to do operations on this result.
- A: Annotation - the user may tag, rate, review or otherwise comment on the search result.
- S: Save - the user use both the time series and the search results to save as a query for later use.
- U: Upload - the user publishes their time series time series and provides directives for intended users, time series categories, etc.

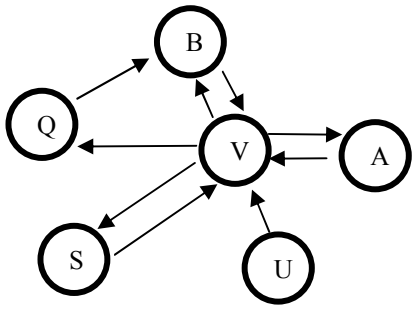

Figure 2. User activities during time series search and uploading

In the figure, only the primary flows are indicated and sites typically allow users to navigate among all the states at will.

VII. FACTORS CONCERNING SCALABILITY

In the process of designing a time series search engine system, factors influence the scalability of the system and basic choice have great effect on the cost required to support a given user base. The following list of scalability factors relates to a wide range of time series search applications [8]:

A. Data Acquisition

Time series data update frequency/ variability: On average, how a lot of datasets are posted to the system for indexing for a given time interval? How does this vary over the course of a day or week? What is the peak arrival rate during busy hours?

- Time series dataset average length: The length of the data effects the processing time.
- Aggregate Time series dataset size to search: Low size will reduce required time to finish search.

B. Data Processing

- Real-time factor: as mentioned above, the types of processing indexing operations can vary widely, from simple metadata ingest to sophisticated feature extraction and time series analysis. Once a particular palette of processing operations is selected, there are tradeoffs within each in terms of accuracy vs. speed.
- Latency constraints: The number of processing servers required is a function of not only real-time factor, time series updating frequency and variance, but also the maximum allowable delay from time series acquisition to being searchable. Continuous time series acquisition and processing applications such as monitoring are characterized by fixed system latency, but for user generated sources, the service can be configured to accommodate the average time series updating frequency. Users who post time series during busy periods will experience more delay in processing, and the perceived quality of service is a function of the maximum and typical processing delays that a user experiences. Also, processing priority schemes can improve the user experience, at the expense of a small number of users. For example, instead of a first come, first served (FIFO) policy, short duration time series can be processed before longer content, or new content from a traditionally popular source can be given higher priority.

C. Data Storage

Regarding storage, the single most important factor by far is the size of the bulk time series. If the original time series is preserved, the policies regarding uploaded time series such as allowable update frequency and whole length.

Alternative time series representations need to be taken into consideration. Are feature data of each time series to be extracted to support a wide range of time resolution to retrieve? For the best user experiences, multiple scale feature data are required.

D. Retrieval

- Peak simultaneous users: As in any Web application, the primary metric determining scalability of the retrieval subsystem is the number of simultaneous users at peak usage times.
- User activity cycle: The retrieval user interface can be crafted such that the user spends varying amounts of time performing the actions of query, browse, and viewing time series (see Figure2. User activities during time series search and uploading) for a given session, and these states consume different sets of system resources. For example, viewing time series puts no load on the query engine, but taxes the data delivery subsystem.
- Number of monitoring searches: monitoring searches need continuously search with the time series datasets updated. If the service is designed such that users rapidly get the monitored search results, as opposed to passively search by user, the load on the server and the total output bandwidth will be increased.

VIII. Retrieval Interfaces

Like other search engine systems, time series search systems may support several interfaces to allow developers to create new applications. Systems may support interfaces including [8]:

A. Web Services (WS)

The simple object access protocol (SOAP) used over HTTP and described by the Web Services Description Language (WSDL) provides a standard framework for interaction with services such as time series retrieval, but the latitude enabled in architecture and parameter semantics implies that time series search Web services are not interoperable in general. Some amount of new interface code is typically required when switching from one Web service to another.

B. SQL/ XML Query

A time series collection may expose an SQL interface and applications may connect using ODBC. The Microsoft Indexing Service is an example of a service that supports an SQL interface but with an underlying architecture that is more indexing oriented than a traditional database. Several XML query languages are available for use with metadata stored in XML.

C. Notification via Email

Systems can support persistent queries stored on the server and notify users via Email. Users can select to be notified once a day rather than multiple times as new matching time series arrives. Servers must efficiently manage stored queries from a potentially large set of users, and possibly pool similar queries.

D. Notification via SMS

Systems can also provide notification via SMS (Short Messages). Especially, in some areas, where SMS is very popular and it is an efficient notification way.

IX. Typical System Features

We indicated notification as an interface, but this may be thought of as more of a "service" or feature than an API. Other features found in time series search services include [8]:

- Favorites - saved searches where results may appear on the search engine landing page freshly executed with each visit.
- History - access to time series recently searched and viewed.
- User preferences - capture and support user preferences such as explicit time series filtering.
- Download - The ability for users to download the time series.
- Annotation - ability for users to post persistent comments, rate or tag time series and make these annotations sharable.

X. Conclusions

We've seen that the basic structure of time series search involves acquisition, data processing and retrieval systems. We presented options for time series search engine architectures and supported features and interfaces along with the associated tradeoffs for system resource utilization. Architectures have evolved as new technologies have become practical and widespread and we will no doubt continue to see advances in time series search architectures going forward.

Acknowledgment

The authors would like to thank all the reviewers of the manuscript. Chuanlei Zhang is a post-doc with Department of Electrical & Computer Engineering, Ryerson University, Toronto, Canada. His research interest includes Time Series Data Mining, Embedded Software.

References

[1] Matt Mohebbi, Dan Vanderkam, Julia Kodysh,Rob Schonberger, Hyunyoung Choi & Sanjiv Kumar, “Google Correlate Whitepaper," http://www.google.com/trends/correlate/whitepaper.pdf, June 2011.

[2] Franky Kin-Pong Chan, Ada Wai-chee Fu and Clement Yu, “Haar Wavelets for Efficient Similarity Search of Time-Series:With and Without Time Warping,” IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 3, May/June 2003, pp. 686–705.

[3] Qingguo Wang, Dmitry Korkin, and Yi Shang, "A Fast Multiple Longest Common Subsequence (MLCS) Algorithm," IEEE Trans. on Knowledge and Data Engineering, Vol. 23, No. 3, March 2011, pp.321-334.

[4] Faraz Rasheed, Mohammed Alshalalfa, and Reda Alhajj, "Efficient Periodicity Mining in Time Series Databases Using Suffix Trees," IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 1, Jan. 2011, pp.79-94.

[5] Xiang Lian, Lei Chen, Jeffrey Xu Yu, "Multiscale Representations for Fast Pattern Matching in Stream Time Series," IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 4, April 2009, pp. 568-581.

[6] Khanh Vu, Kien A. Hua, Hao Cheng, and Sheau-Dong Lang, "Bounded Approximation: A New Criterion for Dimensionality Reduction Approximation in Similarity Search," IEEE Transactions on Knowledge and Data Engineering, Vol. 20, No. 6, June 2008, pp. 768-783.

[7] O¨mer Egecioglu, Hakan Ferhatosmanoglu, and Umit Ogras, "Dimensionality Reduction and Similarity Computation by Inner-Product Approximations," IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 6, June 2004, pp.714-726.

[8] David C. Gibbon, Zhu Liu. Introduction to video search engines. Berlin: Springer, 2008.

[9] Liang, B. and Austin, "*A neural network for mining large volumes of time series data*," IEEE International Conference on Industrial Technology (ICIT) 2005 (14-17 December 2005, City University of Hong Kong). IEEE , New York, pp. 688-693.

[10] Dongsong Zhang and Lina Zhou, "Discovering Golden Nuggets: Data Mining in Financial Application," IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, Nov. 2004, pp. 513-522.

[11] Theophano Mitsa. Temporal data mining. Boca Raton, FL : Chapman & Hall/CRC, 2010.

[12] Fabian Mörchen, "Time series feature extraction for data mining using DWT and DFT," Technical Report, No. 33, Philipps-University Marburg 200