SENSOR: Data-driven Construction of Sketch-based Visual Query Interfaces for Time Series Data

Li Yan
Nanyang Technological University
Singapore
yli055@e.ntu.edu.sg

Nerissa Xu
Nanyang Technological University
Singapore
nerrisa.xu@ntu.edu.sg

Guozhong Li
Hong Kong Baptist University
Hong Kong SAR
csgzli@comp.hkbu.edu.hk

Sourav S Bhowmick
Nanyang Technological University
Singapore
assourav@ntu.edu.sg

Byron Choi
Hong Kong Baptist University
Hong Kong SAR
bchoi@comp.hkbu.edu.hk

Jianliang Xu
Hong Kong Baptist University
Hong Kong SAR
xujl@comp.hkbu.edu.hk

ABSTRACT

Sketching is a common approach to visually query time series data. However, a recent study reported that sketching a pattern for querying is "often ineffective on its own" in practice due to lack of "representative objects" to facilitate *bottom-up search*. In this demonstration, we present a novel *data-driven* sketch-based visual query interface (VQI) construction system called SENSOR to alleviate this challenge. Given a time series dataset, SENSOR automatically constructs its VQI by populating different components from the underlying data. Specifically, it discovers and exposes a set of representative objects in the form of *VST-aware shapelets* to facilitate query formulation. Such data-driven construction has several potential benefits such as empowering efficient *top-down* and *bottom-up* search and portability of the interface across different application domains and sources.

PVLDB Reference Format:
Li Yan, Nerissa Xu, Guozhong Li, Sourav S Bhowmick, Byron Choi, and Jianliang Xu. SENSOR: Data-driven Construction of Sketch-based Visual Query Interfaces for Time Series Data. PVLDB, 15(12): 3650 - 3653, 2022.
doi:10.14778/3554821.3554866

1 INTRODUCTION

Sketch-based visual query interfaces (VQI) [4, 7, 8, 11] for time series data sets (*e.g.,* stock price, ECG) and other data sets (*e.g.,* census data) involving ordered sequences (collectively referred to as data series) enable an end user to convey complex free-form and scale-less patterns of interest freehand, which are then matched against the underlying data to identify regions of interest using some notion of similarity. Although these systems make query formulation effortless, Lee *et al.* [8] observe that *"sketching a pattern for querying is often ineffective on its own. This is due to the fact that sketching makes the assumption that users know the pattern that they want to sketch and are able to sketch it precisely. However this is typically not the case in practice."* They reported that end users follow a *top-down* or *bottom-up* search paradigm for formulating queries. The former refers to search based on user's intuition of how the desired pattern (*i.e.,* query) should look like in theory (*i.e.,* translating a pattern "in-the-head" to a query). The latter refers to search when a user does not have upfront knowledge of what to search for. She learns the key patterns that exist in the dataset through *"representative objects"* to kick start queries. They observed that during query formulation users typically prefer to combine sketching with dragging-and-dropping a recommended pattern onto the canvas, and then modifying it. These patterns act as representative objects of the underlying data to facilitate search. Lee *et al.* [8] reported that bottom-up processes are 40% more commonly used than its top-down counterpart.

At first glance, it may seem that small-sized common shapes (*e.g.,* spike, sink, rise, drop) [5] may qualify as representative objects to aid query formulation. Unfortunately, these patterns are not discriminative or specific to the underlying dataset as they occur in many data series and end users are typically aware of these shapes. On the other hand, representative patterns generated by *zenvisage++* [8, 12] are specific to the underlying dataset and represent a small number of common patterns and outliers. However, they can often be visually complex and large in length (examples can be seen in Fig. 2 in [8]). Consequently, they may impose significant cognitive load on end users to visually interpret them to determine if they are useful (in full or part) for their queries.

We observe that a large number of time series data is associated with class labels [3] as time series classification is a popular task. Even if a time series dataset does not have such labels, any state-of-the-art time series clustering technique [1] can be used to generate them first (*i.e.,* each cluster is one class). Consequently, it opens up the opportunity to use *shapelets* [13], which are discriminative time series subsequences that are maximally representative of a class, to serve as representative objects instead of visually complex and large patterns such as those in *zenvisage++*. Specifically, an end user may browse these shapelets to find patterns of interest, drag-and-drop them on the *Query Panel*, and modify them in any way, if necessary, to sketch a query (*i.e., shapelet-at-a-time* mode).

The selection of *useful* shapelets for display on a VQI is a challenging problem as there are numerous of them in a dataset and not all facilitate query formulation. Manual selection is prohibitively expensive as it demands a comprehensive knowledge of different shapes in the underlying time series data. In this demonstration, we

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554866

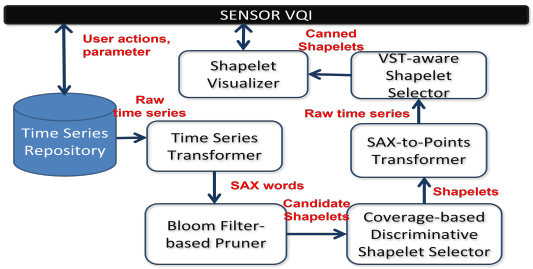

Figure 1: Functional architecture of SENSOR.

Figure 2: Visual interface constructed by SENSOR.

present a novel *data-driven* sketch-based visual time series query interface construction system called SENSOR (**S**ketch-based qu**E**ry i**N**terface con**S**truct**OR**) to address this challenge. Given a time series dataset D with class labels, SENSOR *automatically* populates various panels (*e.g.,* data, patterns) of the VQI from D. Specifically, it selects k *useful* shapelets (*i.e., canned shapelets*) for the *Pattern Panel* that are discriminative, have high *coverage* of D, and within user-specified ease of *visual interpretation* to facilitate sketching. Observe that the recommended patterns of *zenvisage++* are either common patterns or outliers. Hence, they are not selected based on *all* of these criteria. Under the hood, SENSOR extends an efficient shapelet discovery technique [9] for selecting them.

Data-driven selection of VQI components makes SENSOR highly *portable* as it can automatically construct the VQI for *any* application centered around data series. Note that SENSOR *focuses on the interface for query formulation and not on query processing and visualization of results.* One may plug SENSOR on any data series to generate the sketch-based VQI and then install it on top of a state-of-the-art query engine (*e.g.,* [7, 11]). Since SENSOR is independent of any query engine, a sketched query can easily be passed to the underlying engine for its evaluation.

2 DESIGN PHILOSOPHY

SENSOR is designed to give end users the freedom to easily and quickly construct a VQI for any time series data to facilitate top-down and bottom-up search without resorting to coding. Its design is based on the following four principles:

Work with independent data sources. Our data-driven approach should be able to work with any time series data source. SENSOR will offer sufficient benefits to developers and end users by making sketch-based query interface generation effortless.

Focus on shapes. SENSOR focuses on shape formulation during visual querying, which is the most challenging aspect of sketch-based querying. Predicates on shapes (*e.g.,* vertical position, temporal position) and values can easily be added by augmenting a *Query Panel* with either menu icons (*e.g.,* [11]) or an additional panel (*e.g.,* [8, 12]), and is beyond the scope of this work.

Useful shapelets selection. It is paramount to select shapelets that are potentially useful for facilitating top-down and bottom-up search paradigms. Intuitively, shapelets with high *coverage* and *diversity* are desirable as they can support a wide variety of queries. Furthermore, accuracy of freehand sketching varies widely [7]. For instance, complex shapelets (*e.g.,* with many rise and drop) are hard to replicate accurately by sketching due to bio-mechanical and cognitive constraints on humans. A recent study [11] reported that freehand sketches *"preserve visually salient perceptual features but often with non-uniform scale and locally distort a pattern"*. However, the more inaccurate the sketched pattern is from the desired pattern in the dataset, the higher is the chance for the underlying query engine to retrieve irrelevant matches [7]. Hence, it is important to select high-coverage and discriminative shapelets that also alleviate the challenge of sketching a shape accurately while ensuring that end users are exposed to shapelets that are visually easy to interpret during query formulation.

Query log-oblivious. Although query logs can provide rich information of shapes of past queries, in practice, such information is often not publicly available. Hence, SENSOR should be able to generate useful shapelets without demanding query logs as input.

3 SYSTEM OVERVIEW

Figure 1 depicts the architecture of SENSOR. It consists of the following key modules. Interested readers may refer to [9] for detailed discussion related to some of these modules.

The GUI module. Figure 2 depicts a screenshot of the SENSOR GUI. *Panel 1* enables a user to select a time series data source D (with class labels), specify input parameters C for the data-driven construction, and load previously generated shapelets. Note that the parameters enable her to customize the interface according to her requirements. *Panel 2* (*Data Panel*) shows the selected time series data. A user interacts with it via zooming, panning, and navigating iteratively. *Panels 3* and *4* (*Pattern Panel*) visualize common shapes [5] that occur in most datasets and k shapelets selected from D, respectively. We refer to the former as *common shapelets* and the latter as *canned shapelets*. Similar to [11], *Panel 5* (*Query Panel*) is used for freehand and scale-less sketching of a query. A user may drag-and-drop shapelets to sketch a query in shapelet-at-a-time mode. These shapelets can be modified in *Panel 5* through shrinking, stretching, inserting, or removing irrelevant sections. A user may also simply sketch a query freehand.

Observe that the contents of *Panels* 1 and 5 are provided by a user. On the other hand, the contents of *Panels* 2, 3, and 4 depend on the data. Hence, the goal of SENSOR is to populate these three panels. In particular, common shapelets occur in most datasets and hence are provided as default for all. Consequently, populating *Panels 2* and *3* is straightforward. In the sequel, we focus on populating *Panel 4*.

The time series transformer module. The raw time series data is first transformed by PAA [6] and SAX [10], which are a part

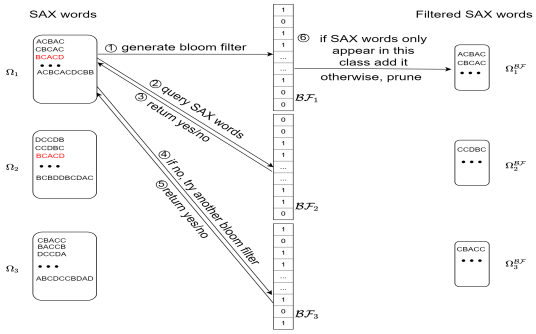

Figure 3: Bloom filter-based pruning [9]

.

of a class called symbolic representations. These methods smooth and discretize the raw data into equal-size segments and aggregate them into a new representation. For example, consider a raw time series with length $n = 8$: [0.156, 0.782, 1.551, 0.349, -0.199, -0.78, 1.058, 0.759]. Let the *compression ratio* be 2 and the alphabets for symbolic representation are {A, B, C, D} (*i.e., alphabet size = 4*). The compression ratio indicates that every two data points are merged by taking their average value. Thus the PAA sequence is of length 4 and the raw time series is converted to upto 3 decimal places: [0.469,0.95,-0.4895,0.9085]. Each value in the PAA sequence is mapped to one of the alphabets based on a lookup table. Thus, the raw time series is transformed into a SAX subsequence (a.k.a SAX word) CDBD. A sliding window method is adopted to generate numerous SAX words.

The bloom filter-based pruning module. Bloom filters that use 13 independent MD5 hash functions are constructed for each class of time series to efficiently prune the same SAX words that also exist in other classes. Such SAX words have less discriminative power. Since bloom filters do not produce false negatives, the SAX words of a class that do not exist in the bloom filters of other classes are definitely not in those other classes. Figure 3 depicts the key steps of using bloom filters $\mathcal{BF}$ to prune the SAX words set Ω. It assumes there are three classes - a bloom filter is generated for each class and takes one SAX word e from Ω_1 as the query for the other two bloom filters generated by Ω_2 and Ω_3. If the SAX word only appears in this class, then it has high discriminative power and thus can pass through the bloom filter and is added to the candidate set. Otherwise, the SAX word has low discriminative power (*i.e.,* also present in other classes) and thus is pruned. Next, the module realizes a non-metric distance-based pruning of similar SAX words, which eliminates similar SAX words that exist in all classes. In this step, similar SAX words exist in different classes are pruned. In addition, for words in the same class, only one representative word is kept for further processing.

The coverage-based discriminative shapelet selection module. For each set of similar SAX words of a class, we consider its term frequency as a weight. Then, this module creates *weighted bitmap structures* of SAX words to quantify the quality of canned shapelet candidates, which are utilized to discover a set of SAX words that has the maximal weight and represents all instances in each class. Specifically, a bitmap structure is constructed for each SAX word e. The length of bitmap is the number of instances in

Figure 4: Shapelets and visual interpretation.

the class and the initial value of each bit is set to 0. If e exists in the sequence T_j, then the value of jth position of the bitmap is set to 1. Thus, $e.bitmap[j]$ indicates whether the SAX word candidate covers T_j. The larger the number of 1s in the bitmap of e, the more instances e covers and the better the quality of e. The *shapelet selection problem* is then formalized as a weighted bitmap cover problem, which can be reduced from the classical weighted set cover problem. A heuristic algorithm is implemented to solve it.

The SAX-to-points transformer module. This module transforms the SAX words back to their raw representations.

The VST-aware shapelet selection module. The preceding modules select shapelets that have high coverage and are discriminative. A user needs to visually inspect them in *Panel 4* to determine if a shapelet is relevant to her search. Depending on the shape of a shapelet, this may demand different degree of cognitive efforts. For example, consider the two shapelets in Figure 4. *Shapelet 1* is intuitively easier to visually interpret as the small bumps (up-down turns) are not visually salient features. On the other hand, the up-down turns in *Shapelet 2* are visually salient. Hence, in order to visually interpret it for querying, a user needs to mentally compute and match the number of such turns and its overall shape.

This module selects k canned shapelets in *Panel 4* by considering the cognitive effort required for visual interpretation. Given the selected shapelets in their raw time series format (*i.e.,* output of *SAX-to-points transformer* module), a user-specified *VST threshold* $\tau \geq 0$ and k, it selects k canned shapelets with *normalized VST count* $C_s \geq \tau$ for display in *Panel 4*. It first counts the total number of *visually salient turns* (VST) in a shapelet and then utilizes it to compute C_s. In order to detect and compute number of turns in a shapelet, we first need to compute the difference between consecutive data points. Assume a simplified shapelet with only 3 data points: [t_1, t_2, t_3]. Consider two scenario shapes of the shapelet: the time series data increases (resp. decreases) from t_1 to t_2 then decreases (resp. increases) from t_2 to t_3. Let $\Delta_1 = t_2 - t_1$ and $\Delta_2 = t_3 - t_2$. It is obvious that when the product of Δ_1 and Δ_2 is negative, there is a turn. Hence, we can compute the number of turns in a shapelet of length ℓ iteratively by incrementing t_i by 1 and computing the product of adjacent Δs. If the product is negative, it increments the number of turns by 1.

Not all turns are visually perceptive to humans (*e.g., Shapelet 1*). Furthermore, when the number of turns increases in a shapelet of length ℓ, the gaps between turns are narrower. This increases the difficulty to visually interpret a shapelet. Hence, we only count *visually salient turns* (VST) in a shapelet. Given a shapelet of length ℓ, a turn is a VST if its *salient ratio* r_s is greater than or equal to a *saliency threshold* σ. The salient ratio is given as $r_s = \Delta Y/\ell$ where ΔY is the difference in y-values of the turn. Observe that r_s gives us the flexibility to determine if minor fluctuations in a time series data source for a specific application are important. We can simply set σ to a very low value in this case.

Lastly, it counts the number of VSTs, denoted by V_s, in each shapelet S. Then the *normalized VST count* of S is given as $C_s = V_s/\ell$. If $C_s \geq \tau$, where τ is a user-defined *VST threshold*, then S is selected for display in *Panel 4*. Notice that τ, V_s, and σ can be interactively varied by an end user to select canned shapelets with different visual complexities.

The parameters k, σ, τ, and V_s are specified through the configuration panel in *Panel 1*. Observe that we take a *lazy* approach of selecting canned shapelets based on VSTs. That is, we perform the selection *after* the coverage-based discriminative shapelet selection. Alternatively, an eager approach may prune the original time series based on VSTs before the transformation of raw series to SAX words. We chose the former to facilitate end users to interactively change $\tau/V_s/\sigma$ to view canned shapelets with different visual complexities without recomputing discriminative shapelets. The eager approach will result in (partial) recomputation of coverage-based discriminative shapelets as these parameters change.

The shapelet visualizer module. This module facilitates the visualization of common and canned shapelets on the VQI. Users can view them by scrolling the relevant panels.

4 RELATED SYSTEMS

Many studies focus on efficient and scalable processing of time series queries, *e.g.,* [6, 7]. SENSOR is complimentary to these efforts. Most existing sketch-based visual querying systems, *e.g.,* [4, 7, 11] *do not* suggest representative patterns beyond simple shapes in a data-driven manner. The work most germane to SENSOR is *zenvisage++*[8, 12], which generates representative patterns to represent common patterns and outliers. As remarked in Section 1, these patterns can be large and visually complex that can be hard to visually interpret.

There are several recent efforts on the data-driven construction of VQIs for subgraph query formulation [2]. These systems expose small subgraphs on a VQI that have high coverage and diversity, and low cognitive load to facilitate bottom-up search. SENSOR is inspired by this spirit of data-driven VQI construction. However, data series and sketch-based querying are fundamentally different from visually querying graph data.

5 DEMONSTRATION OVERVIEW

SENSOR is implemented in Java JDK 1.8. Our demonstration will be loaded with several real time series data with varying number, type, and length from the popular UCRArchive [3] (*e.g., Coffee*, *ECGFiveDays*). Example queries that can be constructed using the canned shapelets will be presented for formulation. Users can also sketch their own ad-hoc queries. A **video** to illustrate the demonstration scenarios is available at https://youtu.be/uxF-_v7oiRg. The key objectives are to enable one to experience the followings.

Data-driven construction of VQI. Through SENSOR's interface (Figure 2), the audience will be able to select a time series dataset and configure the aforementioned parameters through *Panel 1* to automatically construct the VQI containing shapelets in *Panels 3* and *4 within few seconds*. One will be able to interactively select different time series datasets as well as vary the parameters to appreciate the portable and data-driven nature of SENSOR. Note that canned shapelets can be different for different datasets. Furthermore, by varying $\tau/V_s/\sigma$, the audience can experience the trade-off between easy and hard-to-sketch shapelets and their impact on cognitive efforts required to interpret them.

Top-down and bottom-up sketch-based search. The audience can experience the benefits of SENSOR in sketch-based visual querying using the generated VQI. In particular, in existing sketch-based interfaces one has to browse the underlying time series dataset to find pattern-of-interest to trigger bottom-up search as without a clear "pattern-in-head" it is hard to initiate a meaningful sketch [8]. As remarked earlier, this can be a cumbersome and laborious endeavour. We shall demonstrate that SENSOR alleviates this challenge. An audience can browse the canned shapelets and use them as representative objects to trigger bottom-up search. She may (a) simply drag-and-drop one or more shapelets-of-interest to the *Query Panel* and modify them in any way; or (b) view a shapelet-of-interest in *Panel 4* and then replicate it or a portion of it by freehand scale-less sketching in *Panel 5*. Given that browsing the canned shapelets is more efficient than browsing the underlying time series data, through our demonstration the audience will be able to appreciate that SENSOR not only facilitates bottom-up search but also enables more efficient sketch-based query formulation effortlessly. Furthermore, they will able to discern that the shapelets support more accurate sketch generation w.r.t the desired pattern, thereby paving the way for a query engine to retrieve results that are closer to user's perception of similarity match.

ACKNOWLEDGMENTS

Nerissa Xu and Sourav S Bhowmick are supported by AcRF Tier-2 Grant MOE2019-T2-1-029. Guozhong Li is jointly supported by IRCMS/19-20/H01 and HK RGC RIF R2002-20F.

REFERENCES

[1] S. Aghabozorgi, A. S. Shirkhorshidi, T. Y. Wah. Time-series clustering – a decade review. *Information Systems*, 53: 16-38, 2015.
[2] S. S. Bhowmick, B. Choi . Data-driven Visual Query Interfaces for Graphs: Past, Present, and (Near) Future. *In Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*, 2441-2447, ACM, 2022.
[3] Y. Chen, et al. The UCR time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/, Last accessed 20th June, 2022.
[4] M. Correl, M. Gleicher. The Semantics of Sketch: Flexibility in Visual Query Systems for Time Series Data. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 131-140, IEEE CS, 2016.
[5] M. Gregory, B. Shneiderman. Shape identification in temporal data sets. In *Expanding the Frontiers of Visual Anal. and Visualization*, 305-321, Springer, 2012.
[6] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems* 3(3), 263-286, Springer, 2001.
[7] C. Fan, K. Matkovic, H. Hauser. Sketch-Based Fast and Accurate Querying of Time Series Using Parameter-Sharing LSTM Networks. *IEEE Trans. Vis. Comput. Graph.*, 27(12),4495-4506, IEEE CS, 2021.
[8] D.J.L. Lee, et al. You Can't Always Sketch What you Want: Understanding Sensemaking in Visual Query Systems. *IEEE Trans. Vis. Comput. Graph.*, 26(1): 1267-1277, IEEE CS, 2020.
[9] G. Li, et al. Efficient Shapelet Discovery for Time Series Classification. *IEEE Trans. Knowl. Data Eng.*, 34(3), 1149-1163, IEEE CS, 2022.
[10] J. Lin, E. Keogh, S. Lonardi, B.l Chiu. A symbolic representation of time series, with implications for streaming algorithms. *Proc. of the 8th ACM SIGMOD workshop on Res. issues in Data Mining and Know. Discovery*, 2-11, ACM, 2003.
[11] M. Mannino, A. Abouzied. Expressive Time Series Querying with Hand-drawn Scale-free Sketches. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1-13, ACM, 2018.
[12] T. Siddiqui, et al. Effortless Data Exploration with zenvisage: An Expressive and Interactive Visual Analytics System. *Proceedings of the VLDB Endowment*, 10(4), 457-468, 2016.
[13] L. Ye, E. Keogh. Time series shapelets: a new primitive for data mining. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 947–956, ACM, 2009.