VisualFacts: A Platform for In-Situ Visual Exploration and Real-time Entity Resolution

George Papastefanatos[1], Giorgos Alexiou[1,2], Nikos Bikakis[1], Stavros Maroulis[1,2] and Vasilis Stamatopoulos[1]

[1]ATHENA Research Center, Greece
[2]Nat. Techn. Univ. of Athens, Greece

Abstract

VisualFacts is an open-source data visualization platform for big geo-located data. VisualFacts combines in-situ visualization with real-time entity resolution capabilities to address scenarios where users wish to visual explore and efficiently perform analytic operations directly on raw data files, which are aggregated from multiple overlapping data sources. VisualFacts is based on an adaptive index for efficiently scaling up to big volumes; it combines a grid with a tree and a blocking-based structure for efficiently processing spatial, categorical and overlapping data, respectively. It is progressively created based on the user interaction, adapting to the areas and details of the user exploration. This paper provides the architecture of the platform and presents a demonstration of its main features.

Keywords

Data Visualization, Visual Analytics, Adaptive Indexing, In-situ Processing, Entity Resolution

1. Introduction

In-situ visual exploration and analysis has become a common practise due to the availability of big datasets in raw formats (e.g., csv/json). In-situ techniques attempt to avoid the overhead of fully loading and indexing the data in a DBMS, and improve performance by progressively building an index during data exploration. In this work, we focus on in-situ visual analysis of data that is collected and aggregated from multiple sources (e.g., scholarly data, POIs, etc) and thus they contain dirty/duplicate entities, i.e., multiple records in the file may refer to the same real-world entity. Furthermore, the data can be visualized and explored on a 2D layout (e.g., map/scatter plot) and the types of analysis include faceted filtering over categorical attributes, spatial clustering of datapoints, computation of aggregate uni/bivariate statistics which are visualized in charts and, finally real-time entity resolution (ER) [1]. The latter refers to the analysis-aware detection of matching entities between the data points, included in the visualized area and the analysis of the properties, which characterize them as duplicates. Many commercial and research tools offer the functionality to visualize data from raw files [2]; however many of them suffer from large initialization times and poor interactivity performance, when the files become too large to fit in memory or when complex analytic tasks, such as real-time ER, are employed.

Contribution. In this demo paper, we present VisualFacts, a visual analytics platform for big geo-located data that assists users perform analysis of raw data files of varying quality (with duplicates or missing data) in rich visual ways. VisualFacts combines in-situ visualization with real-time ER capabilities. It allows users to use their own data file(s) and start visually interacting with the data on a map without loading or indexing the data in a database. The backbone of the platform is *a visual aware in-memory index*, which is constructed on-the-fly and adjusted to user interaction, as well as an *engine which offers on-the-fly visual ER* and clustering of duplicate data. Specifically, VisualFacts combines and integrates technologies from RawVis [3] and QueryER [4]. The first is a visual tool for in-situ visual exploration and the second offers the ER query engine. **The novelty of this work lies in the extensions we made in the underlying in-situ index to accommodate new structures that can speed up real-time ER in visual exploration scenarios, the integration of the ER query engine in the in-situ query engine, and finally the novel UI modules that enhanced VisualFacts to combine visual analytics on geo-located data with ER analysis tasks**. With these enhancements, the platform can scale up the visualization, interactive exploration and ER analysis to millions of data points on a map, using commodity hardware.

VisualFacts as well as the individual technologies are open source and can be used independently or integrated into existing data management or visual analysis systems.

Published in the Workshop Proceedings of the EDBT/ICDT 2022 Joint Conference (March 29-April 1, 2022), Edinburgh, UK
0000-0002-9273-9843 (G. Papastefanatos); 0000-0002-6307-4053 (G. Alexiou); 0000-0001-6859-1941 (N. Bikakis); 0000-0003-2816-4368 (S. Maroulis); 0000-0002-9044-796X (V. Stamatopoulos)

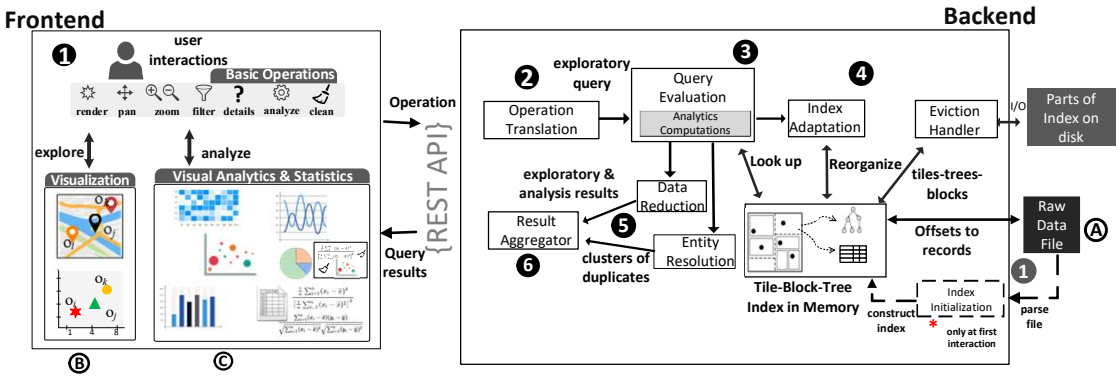

Figure 1: VisualFacts Architecture. Users perform visual operations at the Frontend; Backend evaluates operations on the index and (when needed) the data files. ER is performed during query evaluation and each query results in index adaptation.

More resources are publicly available on GitHub[1] and the project's website[2].

Related Work. Techniques for progressive loading and indexing exist for generic in-situ analysis [5, 6, 7, 8], and visualization [9, 10, 11]. Also, query-driven ER techniques have been proposed [1, 12] in the context of the generic problem of analysis-aware data processing [13]. To the best of our knowledge, this is the first work and system that combines these two problems, and considers real-time ER in in-situ visual analytics settings.

2. Platform Overview

Exploration Scenario and Operations. Fig. 1 provides an overview of our platform. The exploration scenario considers that a user visually interacts with data stored in a single *data file* Ⓐ on disk using a 2D visualization technique (e.g., map, scatter plot) Ⓑ, and analyzes it using visual (e.g., bar, line, pie charts, and heatmaps), and *statistical* methods Ⓒ. The file contains records that correspond to duplicate entities and there is no attribute (e.g., id) that explicitly identifies duplicate records in the file. Attribute values are numerical, categorical, or textual. Two of the numerical ones (e.g., longitude, latitude) are selected explicitly by the user or implicitly by the system (based on the datatype) for the X and Y axis of the 2D visualization.

A visual operation performed by the user is characterized by a window area and a set of parameters (filters, analytic operations, requested statistics in charts, etc.). At the backend, a visual operation is mapped to the query semantics of our framework. An *exploratory query* [9] is an SQL-like expression which includes the data access operations (e.g., a window selection on the X and Y attributes, filtering conditions on the categorical attributes, aggregate functions, etc) for evaluating a visual user action. It also denotes whether the user has requested to perform an ER task on the query results.

Architecture Overview. The web-based frontend (Fig. 1) communicates with the backend via REST API calls. ❶ The user selects a data file Ⓐ and an initial operation on its contents (e.g., select and render the points within a specific area on the map) to start the analysis. VisualFacts parses the file and creates an initial version of the index, called Tile-Tree-Block Index (*Index Initialization*). During the index construction, the results corresponding to the first user request are also fetched. ❷ The user follows a sequence of visual exploration and analysis operations (i.e., interactions), which are sent to the backend and translated to exploratory queries (*Operation Translation*). ❸ Queries are evaluated over the index structure (*Query Evaluation*) to compute and fetch the results (data objects evaluated by the query). ❹ The query parameters are used by the *Index Adaptation* component to adjust the index, i.e., re-organize its tile structure and update its stored statistics. ❺ The results are further processed (Data Reduction component) and reduced (e.g., via spatial clustering) such that over-plotting issues are properly addressed at the frontend visualization. Simultaneously, the *ER* component considers the query results to find and group duplicates into clusters. ❻ The output of these two components (spatial and duplicate clusters) are combined at the *Result Aggregator* component and sent to the frontend as the final query results. Note that, during the index construction or the query evaluation, the index structure may not fit in main memory, in such cases, the *Eviction Handler* component stores parts of the index structure on the disk.

At the frontend, the results are visualized on a map Ⓑ, statistics are rendered in various charts Ⓒ and duplicate

[1] https://github.com/VisualFacts
[2] https://visualfacts.imsi.athenarc.gr/software.html

clusters are also rendered on the map as well as on separate charts. The fontend offers various visual operations for interacting and analyzing the data. (Sec.3).

Tile-Tree-Block Index. The Tile-Tree-Block (TTB) is an in-memory index. The TTB index is an extension of the VETI index [9] with structures that allow real-time ER analysis. VETI is a grid index, which *organizes the data objects in non-overlapping rectangle tiles*. The grid is defined over the domains of the X and Y numeric attributes, and each tile in the grid is defined from intervals in the two domains, respectively. Each tile *encloses the objects*, whose values for the two axis attributes fall within the tile intervals. Every tile is associated with a tree, which organizes the data objects based on values from the categorical attributes. Each level in the tree corresponds to a categorical attribute (e.g, assume two categorical attributes for `Country`, `Gender`) and the nodes of a level correspond to the distinct values found in the objects contained in this tile (e.g., `Greece`, `Italy` are nodes in `Country` level). Note that, different values capturing the same concept (e.g., `Greece` vs. `GR`) in a categorical attribute are not sanitized or resolved in the tree; thus, a level may contain multiple nodes for the same concept. The reference to a data object is kept in the leaves and it is the offset of the record in the data file.

The TTB extends the VETI index with *an additional blocking index per tile* which groups the offsets of objects in a tile in blocks. It considers all the possible attribute values of the objects in a tile (thus increasing the recall of the ER) and associates each value (blocking key) with the list of objects it appears (e.g., `Greece`: $\{o_1, o_3, ..., o_n\}$ is the block for 'Greece' value with reference to the objects containing this value in the 'Country' attribute). Blocking is a common index structure used in ER Papadakis et al. [14] [14] for reducing the number of pairwise comparisons that need to be performed between entities.

Finally, a global link index is used to store the duplicate relations between the objects, which are detected within the exploration session of the user. It is used to avoid performing comparisons and speed up ER tasks for areas previously visited by the user.

Index initialization and Query Evaluation. The raw file is parsed and the index is initialized with an initial set of tiles. Details on the initialization policy are presented in [9]. The index progressively adjusts itself to the user interactions, by splitting visited tiles into more fine-grained ones. Thus, the grid becomes more dense in the area that the user explores. The tree and block indexes are spatially bounded within a tile (which keeps them small in size) and they are rebuilt when a tile is split into new ones, during index adaptation.

In brief, an exploratory query is evaluated over the TTB index as follows: Based on the requested window, it first identifies the tiles overlapping with the query window and uses the tree structure within each tile to evaluate any filter conditions and retrieve the objects answering the query (details in [9]). These results may contain duplicate entities, which (in case the user has requested for ER) are further processed by the ER component on the block index of each tile. The objects evaluated by the query are matched against the tile block index, and a set of candidate comparisons is formed. This step allows the selection of objects in the tile that were not selected by the query but share the same blocks with the query results. If the number of comparisons is high, a metablocking [12] step eliminates unnecessary blocks and redundant comparisons; finally the ER performs the comparisons between the objects selected by the query and new objects selected from the block index. For the actual comparisons, we follow a schema-agnostic approach, and we compare the values of all corresponding attributes between object pairs (more details on ER processing in [4]). In the final step, the resolved records are amended in the Global Link Map, the query is evaluated, and the results are sent to front end along with the resolved duplicates.

3. User Interface Functionality

This section outlines VisualFacts's user interface (Fig. 2) visualizing data about hotels in US (see Section 4). The basic features include:

Dataset Selection. The user can select to explore one of the already added datasets or upload a new CSV file Ⓐ. For the exploration and analysis of a new CSV file, the schema (e.g., latitude and longitude fields, categorical attributes) is automatically detected by the UI and can be further customized by the user.

Map-based Visual Exploration. The user is able to explore and analyze data over different geographical areas using operations like panning and zooming, or focus on a specific area (e.g., neighborhood) by drawing a rectangle over the map Ⓑ. The data objects are clustered on the map to avoid overplotting issues. By clicking on a green cluster, the cluster breaks down into an arachnoid of points, where each point represents a single data object. By clicking on a single point the user can see the details about that entity Ⓒ.

Faceted Filtering. Faceted filtering enables users to define multiple filters over the categorical attributes via the Filtering dropdown Ⓓ. For example, in Fig. 2, the user has selected to view and analyze the 4-star hotels that provide fitness facilities. Active filters are summarized below the filtering dropdown.

Statistics. In the statistics panel Ⓔ, the user can examine univariate (e.g., mean, variance, standard deviation) or

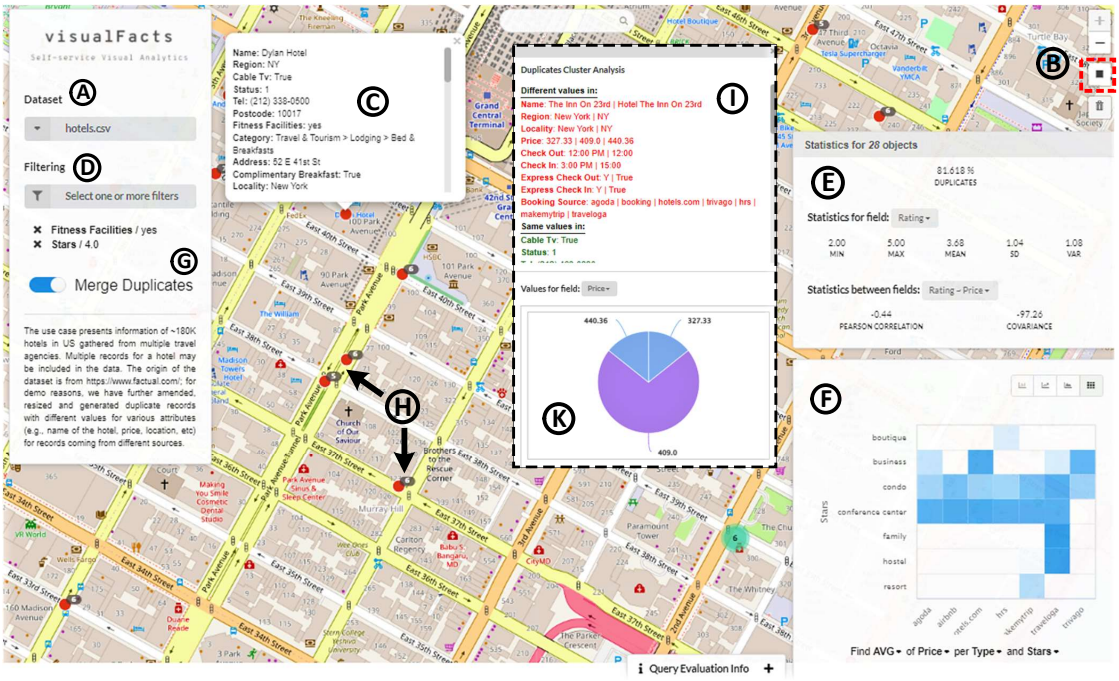

Figure 2: VisualFacts UI Overview

bivariate statistics (e.g., the Pearson correlation, covariance). The statistics are computed for the data points being visible in the entire window or the selected rectangle area and refreshed following every user action on the map (e.g., pan, zoom). In Figure 2, univariate statistics are presented for the hotel rating field, and bivariate for the rating and price fields.

Visual Analysis. The user is able to visually analyze the data by selecting the most suitable visualization type and metrics for their analysis Ⓕ. For example, in Fig. 2, the user has selected a heatmap to visualize the average hotel price w.r.t. the type and stars of the hotels. Following a user action, charts are refreshed following the visible data points on the map.

Visual-based ER Analysis. The ER analysis is activated via a toggle button (*merge duplicates*) Ⓖ. When enabled, the ER operation is performed for the visible points on the map and the duplicate records are clustered. A duplicate cluster is visualized by a red sign annotated with the number of duplicates Ⓗ.

ER Statistics. With data deduplication enabled, the Statistics Ⓔ and Analysis Ⓕ panels on the right are updated to present statistics and metrics evaluated over the deduplicated data.

Attribute-based ER Analysis. The user can select and analyze a specific duplicate cluster on the map, examining its details and the different and common attribute values that appear in its duplicate objects Ⓘ. Additionally, a pie chart presents the percentage of each attribute value over the objects of the cluster Ⓚ.

4. Demonstration Outline

In this section, we outline our demonstration scenario. The tool is available at:

https://visualfacts.imsi.athenarc.gr/platform/visualize/hotels.

The attendees will be able to interact with the tool and analyze two real-world datasets, regarding Hotels and the Restaurants, respectively. The Hotel dataset contains records for hotels in NY (about 180K hotels). Each hotel is described by several attributes, such as name, address, price, type. We assume that the dataset contains data retrieved from different booking platforms (e.g., Booking, Trivago), so, multiple records for a hotel may be included in the data. In brief, we generate this dataset by using as "base" collection, hotel entries retrieved from the public factual API at 2015[3]. Based on these entries, we generate duplicate records by using different values for various attributes (e.g., name of the hotel, price, location), representing records coming from different booking platforms (details are omitted due to lack of space). The Restaurant dataset contains about 180K restaurants from Europe and is provided by the TripAdvisor[4].

[3] www.factual.com

[4] www.kaggle.com/stefanoleone992/tripadvisor-european-restaurants

Users will be able to interact with the prototype and perform several operations such as:

- Interact with the map to pan, zoom in/out to find areas of interest and filter the visualized objects.
- Focus on and analyze specific areas by using the rectangle selection functionality.
- Select the statistics to examine during the exploration.
- Select the visualization type, the attributes and the metrics that will be generated in order to support their analysis tasks
- Use the ER functionalities to: (1) on-the-fly detect and visually present clusters of duplicate entries; (2) examine aggregated statistics w.r.t. the duplicate entities; and (3) analyze the (common and different) attribute's values that appear in the duplicate entities.

Users will be also presented with specific use case scenarios and analytic tasks that will provide better insights into VisualFacts capabilities. Using the Hotel dataset, we assume a scenario where a data analyst, working for a consulting company, assists hotels to advertise their business and offerings across booking platforms. The analyst uses data from different booking platforms (e.g., Booking, Trivago) and compares them against hotel's location, amenities, rating, and prices to match clients' presence and offerings (e.g., price per night) with the most appropriate platform. Assuming that the analyst's company specializes in 4-stars hotels, the following tasks are considered. Initially, the analyst gains an overview of the booking platforms, which have many 4-star hotels. She navigates to the location of interest, filters out the 4-stars hotels and generates a chart with the number of the hotels per data source. Then, she inspects in a heatmap the platforms' coverage for different types (e.g., business, motel) of hotels and amenities and decides the one(s) which covers most of her clients. Finally, using the ER functionalities, she detects and analyzes how the same hotels are presented in different platforms, selects the hotels she has as clients and investigates on the differences and similarities of their offers in the different booking platforms.

VisualFacts has been assessed by an evaluation study of 40 users, who were requested to perform visual tasks similar to the aforementioned and provide their feedback on the usability and interactivity of the tool. Participants reported that the tool provides real-time interaction, the exploratory tasks were easy to perform, and the online ER task could saved time from manually cleaning and preparing the data for visualization. Due to lack of space we omit the details of the study and its findings.

Acknowledgements. The VisualFacts project (1614) has been funded by the Hellenic Foundation for Research and Innovation (ELIDEK) and by the General Secretariat for Research and Technology (GSRT).

References

[1] H. Altwaijry, S. Mehrotra, , D. V. Kalashnikov, Query: A framework for integrating entity resolution with query processing, in: Proceedings of the VLDB Endowment, volume 9(3), 2015.

[2] N. Bikakis, Big Data Visualization Tools, in: Encyclopedia of Big Data Technologies, Elsevier, 2019.

[3] S. Maroulis, N. Bikakis, G. Papastefanatos, P. Vassiliadis, RawVis: A System for Efficient In-situ Visual Analytics, in: ACM SIGMOD, 2021.

[4] G. Alexiou, G. Papastefanatos, V. Stamatopoulos, G. Koutrika, N. Koziris, Queryer: A framework for fast analysis-aware deduplication over dirty data (2022). arXiv:2202.01546.

[5] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, A. Ailamaki, Nodb: Efficient Query Execution on Raw Data Files, in: ACM ACM SIGMOD, 2012.

[6] M. Karpathiotakis, M. Branco, I. Alagiannis, A. Ailamaki, Adaptive Query Processing on Raw Data, PVLDB 7 (2014).

[7] M. Olma, M. Karpathiotakis, I. Alagiannis, M. Athanassoulis, A. Ailamaki, Slalom: Coasting through Raw Data Via Adaptive Partitioning and Indexing, PVLDB 10 (2017).

[8] M. Olma, M. Karpathiotakis, I. Alagiannis, M. Athanassoulis, A. Ailamaki, Adaptive partitioning and indexing for in situ query processing, VLDBJ (2019).

[9] S. Maroulis, N. Bikakis, G. Papastefanatos, P. Vassiliadis, Y. Vassiliou, Resource-Aware Adaptive Indexing for In-situ Visual Exploration and Analytics, VLDB Journal (2022 - to appear).

[10] N. Bikakis, S. Maroulis, G. Papastefanatos, P. Vassiliadis, In-situ Visual Exploration over Big Raw Data, Information Systems 95 (2021).

[11] S. Maroulis, N. Bikakis, G. Papastefanatos, P. Vassiliadis, Y. Vassiliou, Adaptive indexing for in-situ visual exploration and analytics, in: DOLAP, 2021.

[12] G. Alexiou, G. Papastefanatos, Query driven entity resolution in data lakes, in: Springer, Cham, volume International Workshop on Information Search, Integration, and Personalization, 2019, pp. 117–130.

[13] S. Giannakopoulou, M. Karpathiotakis, A. Ailamaki, Cleaning denial constraint violations through relaxation, in: ACM SIGMOD, 2020.

[14] G. Papadakis, G. Alexiou, G. Papastefanatos, , G. Koutrika, Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data, in: PVLDB, 9(4), 2015.