A Data Extraction and Visualization Framework for Information Retrieval Systems

Alessandro Celestini
Institute for Applied Computing, National Research Council of Italy
a.celestini@iac.cnr.it

Antonio Di Marco
Institute for Applied Computing, National Research Council of Italy
a.dimarco@iac.cnr.it

Giuseppe Totaro
Department of Computer Science, University of Rome "Sapienza"
totaro@di.uniroma1.it

ABSTRACT

In recent years we are witnessing a continuous growth in the amount of data that both public and private organizations collect and profit by. Search engines are the most common tools used to retrieve information, and more recently, clustering techniques showed to be an effective tool in helping users to skim query results. The majority of the systems proposed to manage information, provide textual interfaces to explore search results that are not specifically designed to provide an interactive experience to the users.

Trying to find a solution to this problem, we focus on how to extract conveniently data from sources of interest, and how to enhance their analysis and consultation through visualization techniques. In this work we present a customizable framework able to acquire, search and interactively visualize data. This framework is built upon a modular architectural schema and its effectiveness will be illustrated by a prototype implemented for a specific application domain.

Keywords

Data Visualization, Data Extraction, Acquisition.

1. INTRODUCTION

The size of data collected by private and public organizations is steadily growing and search engines are the most common tools used to quickly browse them. Many works, in different research areas, face the problem of how to manipulate such data and to transform them into valuable information, by making them *navigable* and easily searchable. Clustering techniques have been shown to be quite effective to that purpose and have been thoroughly investigated in the past years [17, 18, 2]. However the majority of currently available solutions (e.g., Carrot2[1], Yippy[2]) just supply textual interfaces to explore search results.

In recent years, several works studied how users interact with interfaces during exploratory search sessions, reporting useful results about their behavior [12, 11]. These works show that users spend the majority of their time looking at the results and at the facets, whereas only a neglectable amount of time for looking at the query itself [11] underlining the importance of user interfaces development. According to those works, it is clear that textual interfaces are not very effective to improve exploratory search, so a different solution has to be applied.

Data visualization techniques seem to be well suited to pursue such goals. Indeed, visualization offers an easy-to-use, efficient, and effective method capable to present data to a large and diverse audience including users without any programming background. The main goal of such techniques is to present data in a fashion that supports intuitive interaction to spot patterns and trends, thus making the data usable and informative. In this work we focus on data extraction and data visualization for information retrieval systems, i.e., how to extract data from the sources of interest in a convenient way, and how to enhance their analysis and consultation through visualization techniques. To meet these goals we propose a general framework, presenting its architectural schema composed of four logic units: acquisition, elaboration, storage, visualization. We also present a prototype developed for a case study. The prototype has been implemented for a specific application domain and is available online.

The rest of the paper is organized as follows. Section 2 discusses some frameworks and platforms related to our study. Section 3 presents the framework architectural schema. Section 4 describes a prototype through a case study, and finally, Section 5 concludes the paper suggesting directions for future works.

2. RELATED WORK

In this section we discuss some works proposing frameworks and platforms for data visualization.

WEKA [9] is a Java library that provides a collection of state-of-the-art machine learning algorithms and data processing tools for data mining tasks. It comes with several graphical user interfaces, but can also be extended by using a simple API. The WEKA workbench includes a set of visualization tools and algorithms for classification, regression, attribute selection, and clustering, useful to discover and understand data.

Orange [6] is a collection of C++ routines providing a set of data mining and machine learning procedures which can be easily combined in order to develop new algorithms.

[1] http://project.carrot2.org
[2] http://www.yippy.com/

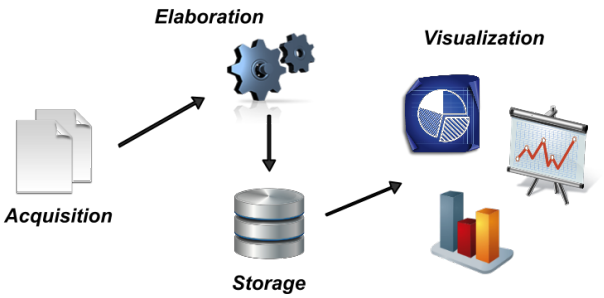

Figure 1: Architectural Schema

The framework allows to perform different tasks including data input and manipulation, methods for developing classification models, visualization of processed data, etc. Orange provides also a scriptable environment, based on Python, and a visual programming environment, based on a set of graphical widgets.

While WEKA and Orange contain several tools to deal with data mining tasks, our aim is to improve information retrieval systems and user data understanding through visualization techniques. Basic statistical analysis on data, should be implemented by charts through interactions patterns, so that could be performed directly by users.

In [8] authors present FuseViz, a framework for Web-based fusion and visualization of data. The framework provides two basic features: fusion and visualization. FuseViz collects data from multiple sources and fuses them into a single data stream. The joint data streams are then visualized trough charts and maps in a Web page. FuseViz has been designed to operate in a smart environment, where several deployed probes sense the environment in real time, and the data to visualize are live time series.

The Biketastic platform [16] is an application developed to facilitate knowledge exchange among bikers. The platform enables users to share routes and experience. For each route Biketastic captures location, sensed data and media. Such information are recorded while participants ride. Routes' data are then managed by a backend platform that makes visualizing and sharing routes' information easy and convenient.

FuseViz and Biketastic share the peculiarity of being explicitly designed to cope with a specific task in a particular environment. The proposed schemas could be re-implemented in different applications, but there is not a clear extension and adaptation procedure defined (and possibly supported) by the authors. Our aim is to present a framework that: *a*) can be easily integrated with an existing information retrieval system *b*) provides a set of tools to profitably extract data from heterogeneous sources *c*) requires minimum effort to produce new interactive visualizations.

3. FRAMEWORK OVERVIEW

Our framework adheres to a simple and well-known schema (shown in Figure 1) structured in four logic units:

1. **Acquisition**: aims at obtaining data from sources;
2. **Elaboration**: responsible for processing the acquired data to fit operational needs;
3. **Storage**: stores the data previously processed in persistent way and make them available to the users;
4. **Visualization**: provides a visual representation of data.

Actually the framework is mainly focused on the acquisition and visualization stages, whereas the other ones are reported as part of the architecture but are not implemented by us. From an engineering perspective, both middle stages (elaboration and storage) are considered as black-box components: only their input and output specifications must be available. All logic units play a crucial role for visualizing data thus we describe them according to the purposes of our framework.

3.1 Acquisition

This component is in charge of collecting and preprocessing data. Given a collection of documents, possibly in different formats, the acquisition stage prepares data and organizes them to feed the elaboration unit.

Data acquisition can be considered the first (mandatory) phase for any data processing activity that anticipates the data visualization. Cleveland [5] and Fry [7] examine in depth the logical structure of visualizing data by identifying seven stages: *acquire*, *parse*, *filter*, *mine*, *represent*, *refine*, and *interact*. Each stage in turn requires to apply techniques and methods from different fields of computer science.

The seven stages are important in order to reconcile all scientific fields involved in data visualization especially from the logical point of view. However, regarding to our prototype we refer to data acquisition as a software component which is able to collect, parse and extract data in an efficient and secure way. The output of data acquisition will be a selection of well-formed contents that are intelligible for the elaboration unit.

We can collect data[3] by connecting the acquisition unit to data source (e.g., files from a disk or data over a network). The approach to data collection depends on goals and desired results. For instance, forensic data collection requires the application of scientifically sound and proven methods[4] to produce a bit-stream copy from data, that is an exact bit-by-bit copy of the original media certified by a message digest and/or a secure hash algorithm. Thus, data collection in many circumstances has to address specific issues about prevention, detection and correction of errors.

The acquired data must be parsed according to their digital structure in order to extract data of interest and prepare them for an elaboration unit. Parsing is potentially a time-consuming process especially while working with heterogeneous data formats. The parsing stage is necessary also to extract the metadata related to examined data. Both textual contents and metadata are usually extracted and stored in specific data interchange formats like JSON or XML.

Moreover, security and efficiency aspects have to be considered during the design of a data acquisition unit. However,

[3]We assume to work with static data. Static/persistent data are not modified during data acquisition, while dynamic data refer to information that is asynchronously updated.

[4]`http://dfrws.org/2001/dfrws-rm-final.pdf`

it is beyond the scope of the present work to discuss security and efficiency related issues regardless their important implications for data acquisition.

3.2 Elaboration and Storage

The elaboration unit takes as input the data extracted during the acquisition phase, so it has to analyze and extrapolate information from them. Data analysis for instance, may be performed by a semantic engine or a traditional search engine. In the former case we will obtain, as output, the documents collection enriched with semantic information, in the second case the output will be an index. Moreover, along with the analysis results, the elaboration unit may return analysis of the metadata, related to the documents, which are received as an input.

The main task of the storage unit is to store analysis results produced by the elaboration unit and make them available for the visualization unit. At this stage the main issue is to optimize data access, specifically the querying time, in order to reduce the time spent by the visualization unit retrieving the information to display. Several storage solutions can be implemented, in particular one may choose among different types of data bases [3, 13]. The traditional choice could be a relational database, but there are several alternatives, e.g., XML databases or graph databases.

3.3 Visualization

The visualization unit is in charge of making data available and valuable for the user. As a matter of fact, visualization is fundamental to transform analysis results into valuable information for the user and help her/him to explore data. In particular, the visualization of the results may help the user to extract new information from data and to decide future queries. As previously discussed, the time spent by the user looking at the query itself is negligible, whereas the time spent looking at the results and how they are displayed is long-lasting. Thus, the interface design is crucial for the effectiveness of this unit, and the guidelines outlined in [12] may became a useful guide for the design and implementation of this unit. Given the tight interaction with the user, it is quite important to take into account the response time and usability of the interface. The visualizations provided should be interactive, to enable the user performing analysis operations on data. The same data should be displayed in several layouts to highlight their different aspects. Finally, it is quite important to provide multiple filters for each visualization, in order to offer to the user the chance of a dynamic interaction with the results.

3.3.1 The "Wow-Effect"

A really-effective data visualization technique has to be developed keeping in mind two fundamental guidelines that are abstraction and correlation.

However, scientists often focus on the creation of trendy – but not always useful – visualizations that should arouse astonishment in the users who observe them, causing what McQuillan [14] defines as the *Wow-Effect*. Unfortunately, the Wow-Effect vanishes quickly and results in having stunning visualizations that are worthless for the audience. This effect is also related to the intrinsic complexity of the data generated from acquisition to visualization stage. As shown in Figure 2, the impact of original data into the total amount

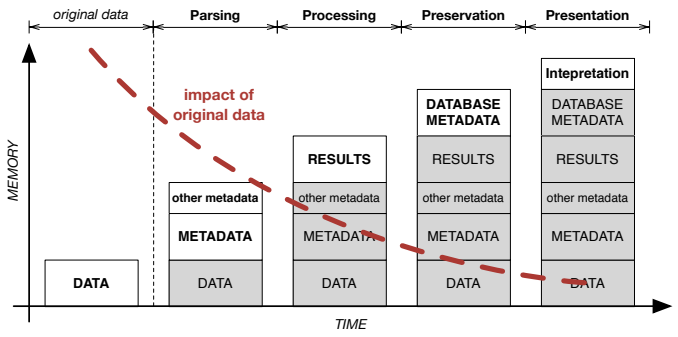

Figure 2: Data enrichment over time

of information decreases over time. Thus, we invested in effort to develop a framework able to overcome the "negative" wow effect by providing visualizations easy to use and effective.

4. CASE STUDY: 4P'S PIPELINE

In this section we present an application of the framework developed for a case study. According to the main task accomplished by each framework unit, we named the whole procedure the *4P's pipeline*: parsing, processing, preservation, and presentation.

The prototype is a browser based application available online[5]. The data set used for testing the 4P's pipeline is a collection of documents in different file formats (e.g., PDF, HTML, MS Office types, etc). The data set was obtained by collecting documents from several sources, mainly related to news in English language.

4.1 Parsing task

The acquisition unit is designed to effectively address the issues discussed in Section 3.1. Parsing is the core task of our acquisition unit and for its implementation we exploited the Apache Tika[6] framework. The Apache Tika is a Java library that carries out detection of document type and the extraction of both metadata and structured textual content. It uses existing parser libraries and supports most data formats.

4.1.1 Tika parsing

Tika is currently the de-facto "babel fish", performing automatic text extraction and content analysis of more than 1200 data formats. Furthermore there are several projects that aim at expanding Tika to handle other data formats.

Document type detection is based on a taxonomy provided by the IANA media types registry[7]that contains hundreds of officially registered types. There are also many unofficial media types that require attention, so Tika has its own media types registry that contains both official registered types and other, widely used albeit unofficial, types. This registry maintains information associated to each supported type. Tika implements six methods for type detection [4] respectively based on the following criteria: filename patterns, `Content-Type` hints, magic byte prefixes, character encodings, structure/schema detection, combined approaches.

[5] `http://kelvin.iac.rm.cnr.it/interface/`
[6] `http://tika.apache.org/`
[7] `http://tools.ietf.org/html/rfc6838`

The `Parser` interface is the key concept of Apache Tika. It provides a high level of abstraction hiding the complexity of different file formats and parsing libraries. Moreover, it represents an extension point to add new parser Java classes to Apache Tika, that must implement the Parser interface. The selection of the parser implementation to be used for parsing a given document may be either explicit or automatic (based on detection heuristics).
Each Tika parser allows to perform text (only for text-oriented types) and metadata extraction from digital documents. Parsed metadata are written to the `Metadata` object after the `parse()` method returns.

4.1.2 *Acquisition unit in detail*

Our acquisition unit uses Tika to automatically perform type detection and parsing, against files collected from data sources, by using all available detectors and parser implementations. Although Tika is, to the best of our knowledge, the most complete and effective way to extract text and metadata from documents, there are some situations where it could not accomplish its job, for example when Tika fails to detect the document format or, even if it correctly recognizes the filetype, when an exception occurs during parsing. The acquisition unit handles both situations by using alternative parsers which are designed to work with specific types of data (see figure 3):

- Whenever Tika is not able to detect a file because either it is not a supported filetype or the document is not correctly detectable (for example, it has a malformed/misleading `Content-Type` attribute), the examined file is marked as `application/octet-stream`, i.e., a type used to indicate that a body contains arbitrary binary data. Therefore, the acquisition unit processes documents whose the exact type is undetectable by using a customized set of ad-hoc parsers, each one specialized to handle specific types. For instance, Tika does not currently support Outlook PST files, so they are marked as `octet-stream` subtypes. Then, the acquisition unit analyzes the undetected file by using criteria as extension pattern or more sophisticated heuristics and finally it sends the binary data to an ad-hoc parser based on the *java-libpst*[8] library.
- During parsing, even though a document is correctly detected by Tika, some errors/exceptions can occur, interrupting the extraction process related to the target file. In this case, the acquisition unit tries to restart the parsing against the file that has caused a Tika exception by using, if available, a suitable parser selected from an ad-hoc parsers list.

The acquisition unit extracts metadata from documents according to a unified schema based on basic metadata properties contained in the `TikaCoreProperties` interface, which all (Tika and ad-hoc) parsers will attempt to extract. A unified schema is necessary in order to have a unique experience with searching against metadata properties. A complete and more complex way to address "metadata interoperability" consists in applying schema matching techniques in order to provide suitable metadata crosswalks.

[8] `https://code.google.com/p/java-libpst/`

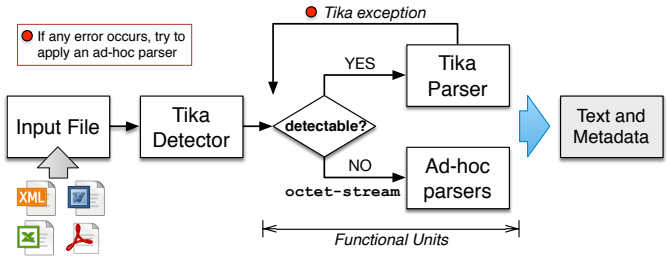

Figure 3: Acquisition unit

4.2 Processing and Preservation tasks

The second and the third tasks are respectively the processing and the preservation of data. The elaboration and storage units which perform these tasks are tightly coupled. All processed data must be stored in order to preserve the elaboration results in a persistent way. They work by using a simple strategy like *Write-Once-Read-Many* pattern, where the visualization unit plays the reader role.

4.2.1 *Elaboration unit*

The elaboration unit is formed by the semantic engine Cogito[9]. Cogito analyzes text documents, and is able to find hidden relationships, trends and events, transforming unstructured information into structured data. Among the several analysis it identifies three different types of entities (people, places and companies/organizations), categorizes documents on the basis of several taxonomies and extract entities co-occurrences. Notice that this unit is outside the framework despite we included it in the architectural schema. Indeed, we do not take care of the elaboration unit design and development, we consider it as given. This unit is the entity with which the framework interacts and to which the framework provides functionalities, i.e., text extraction and visualization.

4.2.2 *Storage unit*

As storage unit we resorted to BaseX[10], an XML data base. BaseX is an open source solution released under the terms of the BSD License. We decided to use an XML data base because the results of the elaboration unit are returned in XML format. Moreover, the use of an XML data base helps to reduce the time for XML documents manipulation and processing, compared to a middleware application [10, 15]. An XML data base has also the advantage of not constraining data to a rigid schema, namely in the same data base we can add XML documents with different structures. Thus, the structure of the elaboration results can change without effecting the data base structure itself.

4.3 Presentation task

For the development of the visualization unit we used D3.js[11] [1], a JavaScript library. The library provides several graphical primitives to implement visualizations and uses only web standards, namely HTML, SVG and CSS. With D3 it is possible to realize multi-stage animations and interactive visualizations of complex structures.

[9] `http://www.expertsystem.net`
[10] `http://basex.org`
[11] `http://d3js.org`

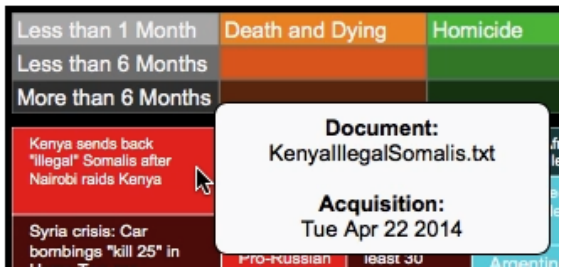

Figure 4: Treemap with category zooming

Figure 5: Geographic visualization and country selection

To improve data retrieval, we realized several visualization alternatives that exploit Cogito's analysis results. Figure 4 shows a treemap visualization that displays a documents categorization, notice that the same document may fall in different categories. Not all categories are displayed, only eight among the most common ones. The categories reported are selected on the basis of the number of documents contained in the category itself. The treemap visualization is quite effective in providing a global view of the data set. Our implementation enables also a category zooming to restrict the set of interest, i.e., clicking on a document the visualization displays only the documents in the same category. Moreover, the user is able to retrieve several information such as the document's name, part of the document content and the document's acquisition date, directly from the visualization interface. Figure 5 shows a geographic visualization that displays a geo-categorization of documents. The countries appearing in the documents are rendered with a different color (green), to highlight the difference respect to the others. The user can select each green country to get several

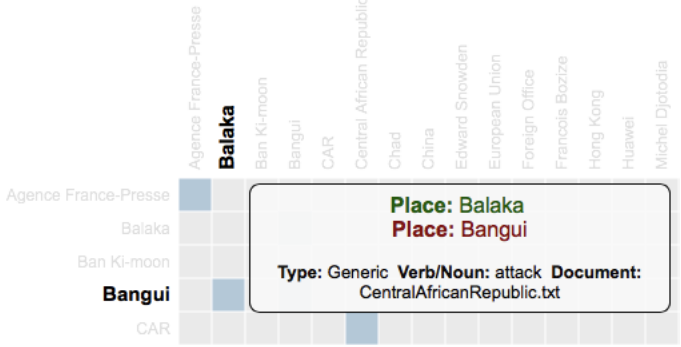

Figure 6: Co-occurrences matrix.

information that are reported inside a tooltip as shown in figure. For each country are reported general information such as capital's name, spoken languages, population figures, etc. Such information do not come from the Cogito analysis, but are added to enrich and enhance the retrieval process carried out by users. The tooltip reports also the list of documents in which the country appears and the features detected by Cogito. Features are identified according to a specific taxonomy and for each country are reported all the features detected inside the documents related to that country. Moreover, this visualization displays geographic locations belonging to the country, possibly identified during the analysis, e.g. rivers, cities, mountains, ecc. Figure 6 shows the visualization of entities co-occurrence (only a section of the matrix is reported in figure). Three types of entities are identified by Cogito, that are places, people, organizations. All entities are listed both on rows and columns, when two entities appear inside the same document the square at the intersection is highlighted. The color of the squares is always the same, but the opacity of each square is computed on the basis of the number of co-occurrences. Thus, the higher the number of co-occurrences, the darker the square at the intersection. Furthermore, a tooltip for each highlighted square reports the type of the two entities, information about the co-occurrence and the list of documents in which they appear. Specifically, the tooltip reports the verb or noun connecting the entities and some information about the verb or noun used.

Figure 7 shows a force directed graph that displays the relations detected among the entities identified in the documents. Each entity is represented by a symbol denoting the entity's type. An edge connects two entities if a relation has been detected between them, self-loop are possible. Edges are rendered with different colors based on relations' type. The legend concerning edges and nodes is reported on top of the visualization. A tooltip reports some information about the relations. In particular, for each edge is reported the sentence connecting the entities, the verb or noun used in the sentence and the document's name in which the sentence appear. Instead for each node a tooltip reports the list of document in which the entity appears. Furthermore, for each visualization, the user may apply several filters. In particular, we give the possibility to filter data by acquisition date, geographic location, nodes' types (co-occurrence matrix and force directed graph), relations' type (force directed graph), categories (treemap).

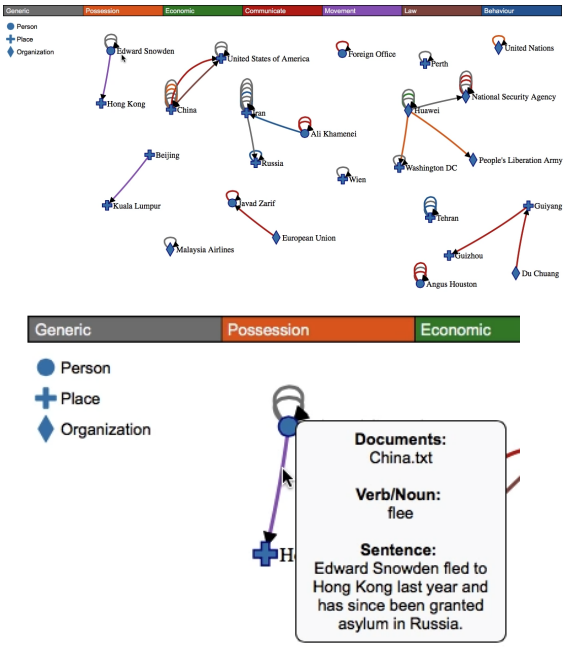

Figure 7: Entity-relations force directed graph

5. CONCLUSIONS

The interest in data visualization techniques is increasing, indeed these techniques are showing to be a useful tool in the processes of data analysis and understanding. In this paper we have discussed a general framework for data extraction and visualization, whose aim is to provide a methodology to conveniently extract data and facilitate the creation of effective visualizations. In particular, we described the framework's architecture, illustrating its components and its functionalities, and a prototype. The prototype represents an example of how our framework can be applied when dealing with real information retrieval systems. Moreover, the online application demo provides several visualization examples that can be reused in different contexts and application domains.

Currently we're experimenting our prototype for digital forensics and investigation purposes, aiming at providing to law enforcement agencies a tool for correlating and visualizing off-line forensic data, that can be used by an investigator even if she/he does not have advanced skills in computer forensics. As a future activity we plan to release a full version of our prototype. At the moment the elaboration engine is a proprietary solution that we cannot make publicly available, hence we aim at replacing this unit with an open solution. Finally, we want to enhance our framework in order to facilitate the integration of data extraction and data visualization endpoints with arbitrary retrieval systems.

Acknowledgements

We would like to express our appreciation to Expert Systems for support in using Cogito. Moreover, financial support from EU projects HOME/2012/ISEC/AG/INT/4000003856 and HOME/2012/ISEC/AG/4000004362 is kindly acknowledged.

6. REFERENCES

[1] M. Bostock, V. Ogievetsky, and J. Heer. D^3 Data-Driven Documents. *IEEE TVCG*, 17(12):2301–2309, Dec 2011.

[2] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):1–17, Jul 2009.

[3] R. Cattell. Scalable SQL and NoSQL Data Stores. *SIGMOD Rec.*, 39(4):12–27, May 2011.

[4] M. Chris and J. Zitting. *Tika in Action*. Manning Publications Co., 2011.

[5] W. S. Cleveland. *Visualizing data*. Hobart Press, 1993.

[6] J. Demšar, T. Curk, A. Erjavec, v. Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, and B. Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14(1):2349–2353, Jan 2013.

[7] B. Fry. *Visualizing Data: Exploring and Explaining Data with the Processing Environment*. O'Reilly Media, Inc., 2007.

[8] G. Ghidini, S. Das, and V. Gupta. FuseViz: A Framework for Web-based Data Fusion and Visualization in Smart Environments. In *Proc. of IEEE MASS '12*, pages 468–472, Oct 2012.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov 2009.

[10] S. Jokić, S. Krco, J. Vuckovic, N. Gligoric, and D. Drajic. Evaluation of an XML database based Resource Directory performance. In *Proc. of TELFOR '11*, pages 542–545, Nov 2011.

[11] B. Kules, R. Capra, M. Banta, and T. Sierra. What do exploratory searchers look at in a faceted search interface? In *Proc. of JCDL '09*, pages 313–322, 2009.

[12] B. Kules and B. Shneiderman. Users can change their web search tactics: Design guidelines for categorized overviews. *Information Processing & Management*, 44(2):463–484, Mar 2008.

[13] K. K.-Y. Lee, W.-C. Tang, and K.-S. Choi. Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage. *Computer Methods and Programs in Biomedicine*, 110(1):99–109, Apr 2013.

[14] A. G. McQuillan. Honesty and foresight in computer visualizations. *Journal of forestry*, 96(6):15–16, Jun 1998.

[15] M. Paradies, S. Malaika, M. Nicola, and K. Xie. Comparing xml processing performance in middleware and database: A case study. In *Proc. of Middleware Conference Industrial Track '10*, pages 35–39, 2010.

[16] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava. Biketastic: Sensing and Mapping for Better Biking. In *Proc. of SIGCHI '10*, pages 1817–1820, 2010.

[17] O. Zamir, O. Etzioni, O. Madani, and R. M. Karp. Fast and intuitive clustering of web documents. In *Proc. of KDD '97*, pages 287–290, 1997.

[18] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *Proc. of SIGIR '04*, pages 210–217, 2004.