Formal Concept Analysis Research Toolbox and Failure Deterministic Finite Automata

Alexey Neznanov, Derrick G. Kourie

National Research University "Higher School of Economics", Moscow, Russia
University of Pretoria, Pretoria, South Africa
ANeznanov@hse.ru, dkourie@gmail.com

Abstract. Formal Concept Analysis Research Toolbox (FCART) is an integrated environment for knowledge and data engineers with a set of research tools based on Formal Concept Analysis (FCA). In the paper we consider main FCA workflow and some applications in the field of the text pattern matching.

Keywords: Pattern Matching, Formal Concept Analysis, Finite Automata, Software.

1 Introduction

Formal Concept Analysis (FCA) [1] is a mature group of mathematical models and good foundation for the methods of intelligent system construction. At present FCA has many applications in various data analysis and knowledge processing tasks [2,3,4].

In previous articles [5,6] we described the stages of development of a software system for information retrieval and knowledge discovery from various data sources based on FCA approach. Formal Concept Analysis Research Toolbox (FCART) was designed especially for the analysis of the complex data including unstructured text collections.

In this paper we will focus on basic FCA tasks and some interesting applications in the field of the text pattern matching.

2 FCART methodology and technology

The underlying methodology of the FCART allows one to obtain new knowledge from data in an iterative ontology-driven process. The software is based on modern methods and algorithms of data analysis, technologies for processing big data collections, data visualization, reporting, and interactive processing techniques. The FCART evolves in the direction of a distributed system based on REST web-services [7]. It implements several basic principles:

1. The iterative process of data analysis using ontology-driven queries and interactive artifacts (such as concept lattices, clusters, implications, etc.).
2. The separation of the processes of *data querying* (from various data sources), *data preprocessing* (of locally saved immutable snapshots), *data analysis* (in interactive visualizers of immutable analytic artifacts), and *results presentation* (in a report builder).
3. The extendibility on three levels: customizing settings of data access components, query builders, solvers and visualizers; writing scripts (macros); developing components (add-ins).
4. The explicit definition of analytic artifacts and their types. It allows one to check the integrity of analytical session data and provides links between artifacts for an end-user.
5. The implementation of the integrated performance estimation tools.
6. The integrated documentation of software tools and methods of data analysis.

The current local version of FCART consists of the following components.

- Core component includes:
  - the multiple-document user interface of research environment with session manager and extensions manager,
  - snapshot profiles editor (SHPE),
  - snapshot query editor (SHQE),
  - query rules database (RDB),
  - session database (SDB),
  - core of report builder.
- Intermediate Data Storage (IDS) for preprocessed data collections in JSON format.
- Internal solvers and visualizers.
- Additional plugins, scripts and report templates.

We use Microsoft and Embarcadero programming environments and different programming languages (C++, C#, Delphi, Python and other). The native executable (the core of the system) is compatible with Microsoft Windows 2000 and later and has not any additional dependences. The important feature of FCART is scripting. Thus, generating and transforming artifacts, drawing, and building reports can be done by scripts. For scripting we use Delphi Web Script and Python languages.

From an analyst point of view, the basic FCA workflow in FCART has four stages (see Fig. 1). On each stage, a user has the ability to import/export every artifact or add it to a report.

1. The filling Intermediate Data Storage (IDS) of FCART from various external SQL, XML or JSON-like data sources (querying external source described by an External Data Query Description – EDQD). EDQD can be produced by some External Data Browser.
2. The loading a data snapshot from the IDS into an analytic session (a snapshot is described by a Snapshot Profile). A data snapshot is a data table with annotated structured and text attributes loaded in the system by accessing IDS.

3. The transforming a snapshot to a binary context (a transformation is described by a Scaling Query).
4. The building and visualizing formal concept lattice and the other artifacts based on the binary context in a scope of analytic session.

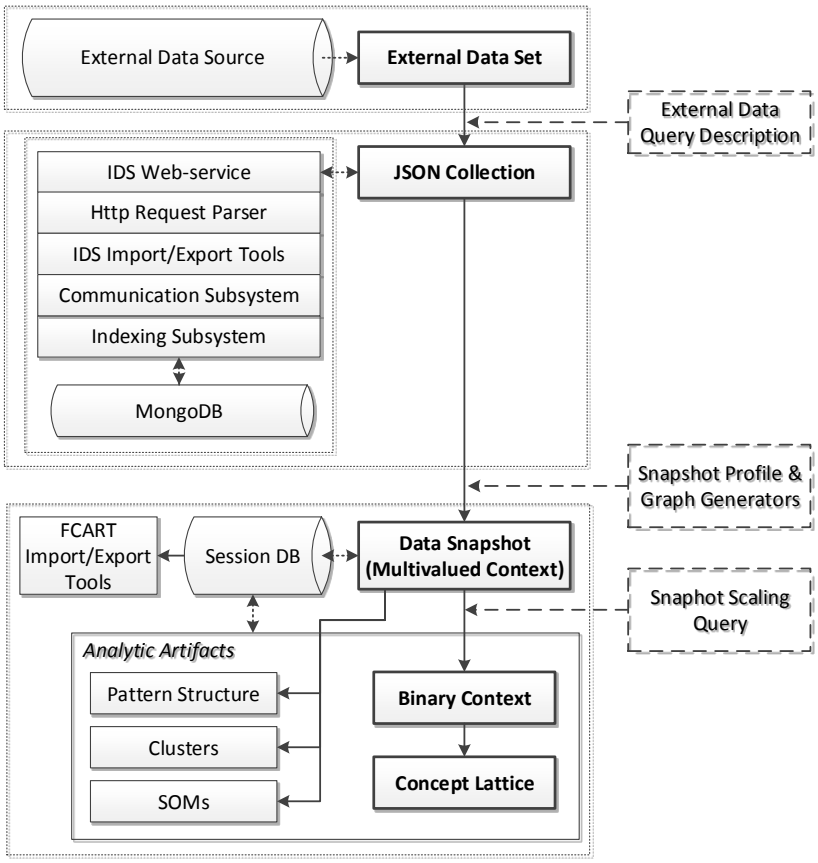

Fig. 1. Main workflow in FCART

3 Working with analytical session

3.1 Analytical session structure

Some of FCA entities appear to be fundamental to the information representation. In the methodology, we use the term "*analytic artifact*" to denote the definition of an abstract interface, describing the entity of the analytic process. The basic artifact for FCA-based methods is that of the "formal context", i.e., object-attribute representation of a subject domain. The most important artifacts also include the "concept lattice" and the "formal concept".

Artifacts' instances can be linked by "origination" relationship. For example, we can generate a concept lattice from the given formal context. In this case, the formal concept will be an "origin artifact" or simply "origin" for the lattice. Another example is the pair of a lattice and an "association rules set": the lattice is the origin of the rules set. Any artifact's instance is *immutable*. It means that any instance cannot be changed after creation, but can be copied and visualized in various ways.

If we have the predefined set of artifacts, in the most cases we can use the term "artifact" instead of "artifacts' instance" without ambiguity. The collection of all artifacts in current analytic cycle forms so-called "*analytical session*".

An "*Initial artifact*" is an artifact without origin in a current session. For example, after loading a formal context from a file on a disk we obtain an initial artifact. Such an artifact becomes one of the roots in a current session.

3.2 Solvers, visualizers, and reports

An analyst works with analytic artifacts using solvers, visualizers and reports.

Solvers. All types of artifacts are generated by solvers. Each solver requires one or many artifacts' instances of preassigned types as an input and produces one artifact's instance of preassigned type as an output. The parameters profile of a solver allows to save/load parameters and automatically create dialogs for a user input. Each parameter has Id, title, data type, hint, group and default value.

Having the predefined types of artifacts and links (assigned by solvers) between immutable artifacts' instances we can check the integrity of data of particular analytical session. Without explicit user action a session cannot lose any artifact instances and links, and guarantees the integrity of a session.

A solver without input artifacts is called "*Generator*". Generator can use a script or a plugin for automatic building artifact from scratch.

Visualizers. Artifact visualizer is a special solver that generates the user-oriented visual representation of input artifact (or several artifacts) instance. From a technical point of view, visualizer produces an interactive or non-interactive window with some elements of user interface. Of course, one artifact can have different kinds of visual appearance.

Each artifact type has *default visualizer*, which is invoked by clicking on an artifact in the session window. The parameters profile of visualizer defines common visual properties and drawing parameters (for example, coordinates of concepts in a concept lattice). In interactive visualizer, a user has the set of tools for changing the drawing parameters "on the fly". Usually, visualizer is the last in a chain of solvers. However, we can get a visual representation of each artifact in the current session. For example, formal concept lattice visualizer (lattice browser) draws a diagram of the lattice and allows a user to manipulate the diagram, but this browser itself does not generate new artifacts. We need to distinguish the generation of new artifact and drawing of an existing artifact for various cases: working in the batch mode, increasing efficiency of long chains of solvers, benchmarking, etc. Of course, a complex visualizer can help invoking next solvers by simplifying select of some input artifacts as parts of the visualized artifact.

Fig.3 shows the user interface of the standard formal concept lattice visualizer.

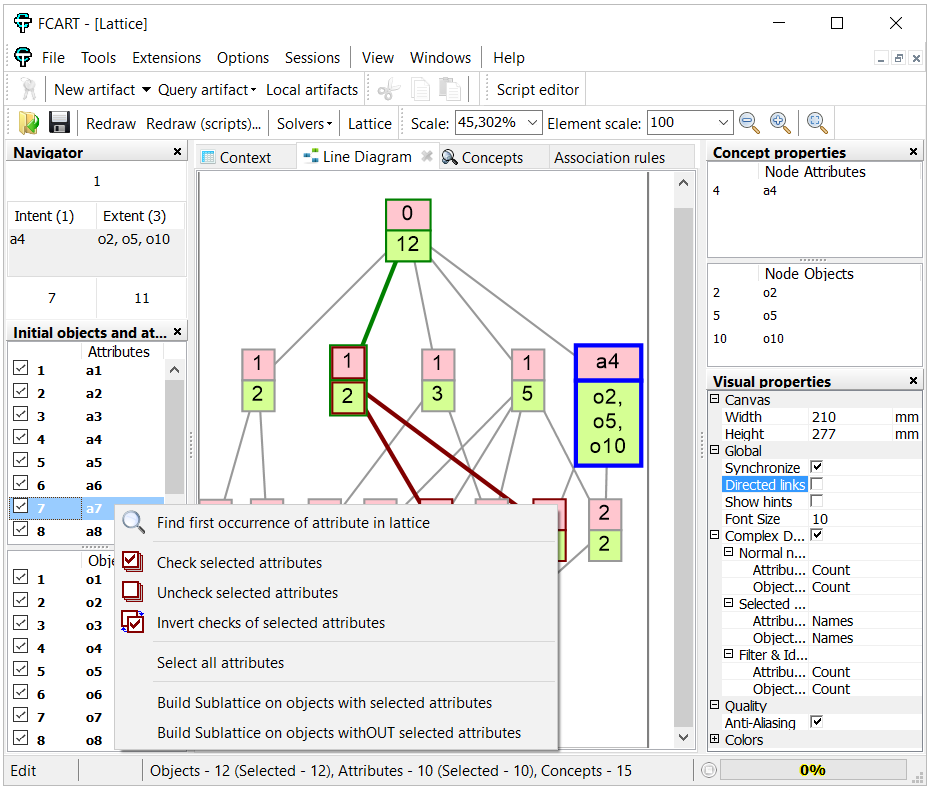

Fig. 3. FCART formal concept lattice visualizer

Reports. Report is a final representation of the research results. Every scientific environment should provide a rich text editor of a report with additional functionality to avoid mistakes while converting and moving multiple artifacts with metadata into an external editor. The main feature of the FCART report builder is the automatic insertion of the fully decorated artifact representation in the resulting report. An artifact can support several representations in form of a text as of a vector or a bitmap image as well.

3.3 Interactive editing of artifacts

What is sources of the artifacts? An initial artifact can be generated by:

1. The executing a script or a plugin (with type “Generator of …”) without an input artifact.
2. The loading an artifact from a session or an external file
3. The querying to the IDS or some External Data Source.

4. The opening one of the *interactive editor*.

The result of a generator or a query becomes immutable. It is also obvious that artifact is mutable while editing by an interactive editor. A user can send a copy of an artifact to an interactive editor, if FCART has the registered editor for such an artifact's type. Fig. 4 shows the user interface of the standard formal context editor. After execution of the "Build lattice…" command the immutable formal context is added to the session and becomes the origin of the formal concept lattice.

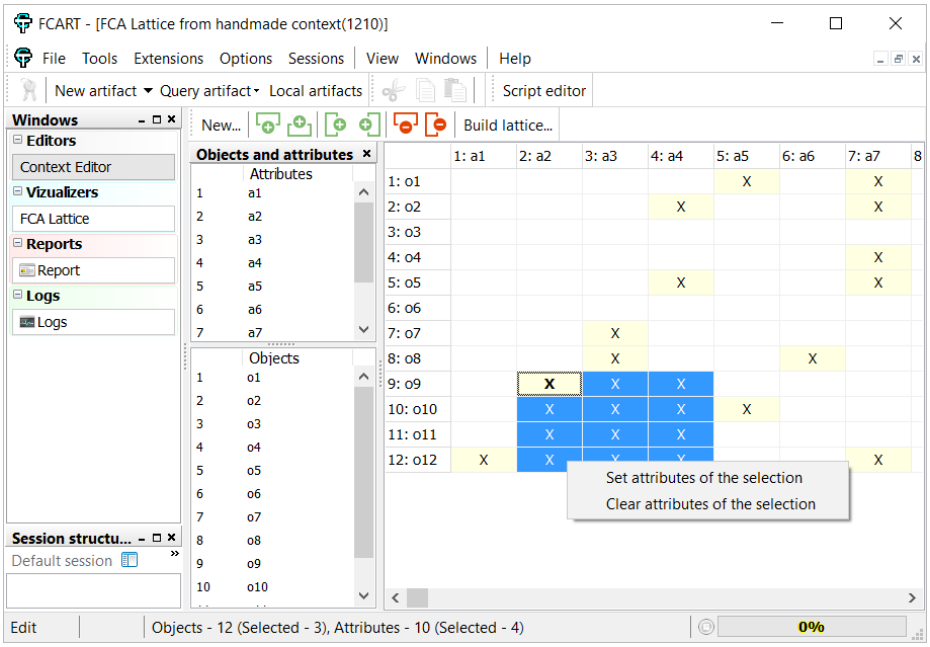

Fig. 4. FCART interactive editor of a context

4 FCART and Failure Deterministic Finite Automata

The Formal Concept Analysis approach is used for transform a Deterministic Finite Automata (DFA) into a Failure Deterministic Finite Automata (FDFA). FDFA is proposed and defined by Kourie D.G., Watson B.W. and others in 2012 [8].

Madoda N., Kourie's master student, had used the local version of FCART in 2014 for efficient generation of the state/out-transition concepts and interactive analysis of the state/out-transition (formal concept) lattice [9]. FCART had been integrated in the DFA – Homomorphic Algorithm (DHA) workflow without any specific input/output formats or plugins. There are three stages of FCART usage in DHA:

1. Loading state/out-transition context for the given DFA from a CSV file.
2. Building formal concept lattice for a given DFA.
3. Saving PAR concepts into an XML file.

Optionally a user can check the context and analyse the lattice that also saved in XML format.

The next step should be the development of a new web-solver for distributed version of FCART [7]. It will rise the efficiency of DHA implementation and will allow using DHA as a REST web-service.

5 Conclusion and future work

In this paper we have described the FCART software and its application in research of failure deterministic finite automata. The demo-version of used local version 0.9.4 of the FCART can be downloaded from http://cs.hse.ru/en/ai/issa/proj_fcart.

We will try to advance some functionality in the distributed version of FCART.

Acknowledgements

This work was carried out by the authors within the project “Data mining based on applied ontologies and lattices of closed descriptions” supported by the Basic Research Program of the National Research University Higher School of Economics.

References

1. Ganter, B., Wille R. Formal Concept Analysis: Mathematical Foundations, Springer, 1999.
2. Priss U. Formal Concept Analysis in Information Science, Annual Review of Information Science and Technology, 40, 2006, pp. 521-543.
3. Poelmans J., Kuznetsov S.O., Ignatov D.I., Dedene G. Formal Concept Analysis in knowledge processing: A survey on models and techniques, Expert Systems with Applications, 40 (16), 2013, pp. 6601-6623.
4. Poelmans J., Ignatov D. I., Kuznetsov S.O., Dedene G. Formal concept analysis in knowledge processing: A survey on applications, Expert Systems with Applications, 40 (16), 2013, pp. 6538-6560.
5. Neznanov A.A., Ilvovsky D.A., Kuznetsov S.O. FCART: A New FCA-based System for Data Analysis and Knowledge Discovery, Contributions to the 11th International Conference on Formal Concept Analysis, 2013, pp. 31-44.
6. Neznanov A., Parinov A. FCA Analyst Session and Data Access Tools in FCART, 16th International Conference “Artificial Intelligence: Methodology, Systems, and Applications” (AIMSA-2014), 2014, pp. 214-221.
7. Neznanov A.A., Parinov A.A. Distributed Architecture of Data Analysis System based on Formal Concept Analysis Approach, Intelligent Distributed Computing IX, 2015, pp. 265-271.
8. Kourie D.G., Watson B.W., Cleophas L., Venter F., Failure Deterministic Finite Automata, Proceedings of the Prague Stringology Conference 2012б, 2012, pp. 28-41.
9. Madoda A.N. An Assessment of Selected Algorithms for Generating Failure Deterministic Finite Automata. Master’s dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, 2015.