Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 31 (2014) 201 – 210

Procedia Computer Science

2nd International Conference on Information Technology and Quantitative Management, ITQM 2014

Advancing FCA Workflow in FCART System for Knowledge Discovery in Quantitative Data

Alexey Neznanov[a]*, Dmitry Ilvovsky[a], Andrey Parinov[a]

[a]*National Research University Higher School of Economics, 20 Myasnitskaya Ulitsa, Moscow, 101000, Russia*

Abstract

We describe new features in FCART software system, an integrated environment for knowledge and data engineers with a set of research tools based on Formal Concept Analysis. The system is intended for knowledge discovery from various data sources, including structured quantitative data and text collections. Final version of data transformation from external data source into concept lattice is considered. We introduce new version of local data storage, query language for conceptual scaling of data snapshots as multi-valued contexts, and new tools for working with formal concepts.

Keywords: Data Analysis; Formal Concept Analysis; Software; Multi-valued Formal Context.

Introduction

The goal of developing the software package called "Formal Concept Analysis Research Toolbox" (FCART) is to create a universal extensible integrated environment[1]. The methodology of using this software is based on modern methods and algorithms of data analysis, technologies for manipulating big data collections, data visualization, reporting and interactive processing techniques. It allows one to obtain new knowledge from data with full process tracking and reproducibility. Some ideas of data preprocessing were inherited from the CORDIET project[2].

* Corresponding author. Tel.: +7-495-772-95-90 * 22672; fax: +7-495-772-95-90 * 22668. *E-mail address:* aneznanov@hse.ru.

doi:10.1016/j.procs.2014.05.261

We introduce new functionality of the system in fields of data storage, preprocessing, and querying, as well as new subsystem for working with concepts properties.

1. Formal Concept Analysis

Formal concept analysis (FCA) is a subfield of applied lattice theory, the methods of which are used to solve different problems of data analysis[3].

A formal context is a triple $K = (G, M, I)$, where G is called a set of *objects*, M is called a set of *attributes*, $I \subseteq G \times M$ is a binary relationship between G and M. The following two mappings form a Galois connection between powersets of M and G for any $A \subseteq G, B \subseteq M$:

$$A' = \{m \in M \mid gIm \text{ for all } g \in A\},\ B' = \{g \in G \mid gIm \text{ for all } m \in B\} \quad (1)$$

The first mapping takes a set of objects to a maximal set of attributes possessed by each object from the set. Similarly, the second mapping takes a set of attributes to a maximal set of objects possessing all attributes from the set.

A *formal concept* is a pair (A, B), where $A \subseteq G$ is a subset of objects (called extent), $B \subseteq M$ is a subset of attributes (called intent), $A' = B, A = B'$. The set of all concepts is ordered and this ordered set makes a lattice called *concept lattice*.

2. Formal Concept Analysis Research Toolbox architecture and use cases

We consider the new features of version 0.9 of FCART in the form of local Windows application. We don't consider another line of development in form of Web-based system based on Microsoft .NET platform. FCART is constructed as multicomponent application. We use Microsoft and Embarcadero programming environments and different programming languages (C++, C#, Delphi, Python and other). For inner scripting we use Delphi Web Script and Python. Native executable (the core of the system) is compatible with Microsoft Windows 2000 or later and has not any additional system-level dependences.

Current local version consists of the following components.

- Multiple-document user interface of research environment with session manager.
- Local Data Storage (LDS) for preprocessed data.
- Snapshot profiles editor (SHPE).
- Snapshot (scaling) query editor (SHQE).
- Query rules database (RDB).
- Session database (SDB).
- Report builder.
- Internal solvers and visualizers.
- Additional plugins and scripts.

Current local version intended primarily for analysts who used FCA as a method of knowledge discovery.

2.1. Main FCA workflow overview

Main FCA workflow in FCART has four stages (see Fig. 1) from the analyst point of view.

1. Filling Local Data Storage of FCART from various external data sources.
2. Loading a data snapshot from local storage into current analytic session.
3. Transforming the snapshot to a binary context.
4. Building and visualizing formal concept lattice and other artifacts based on the binary context.

We will pay attention to the stages with improved functionality: querying external data, scaling snapshots, and calculating concepts indices.

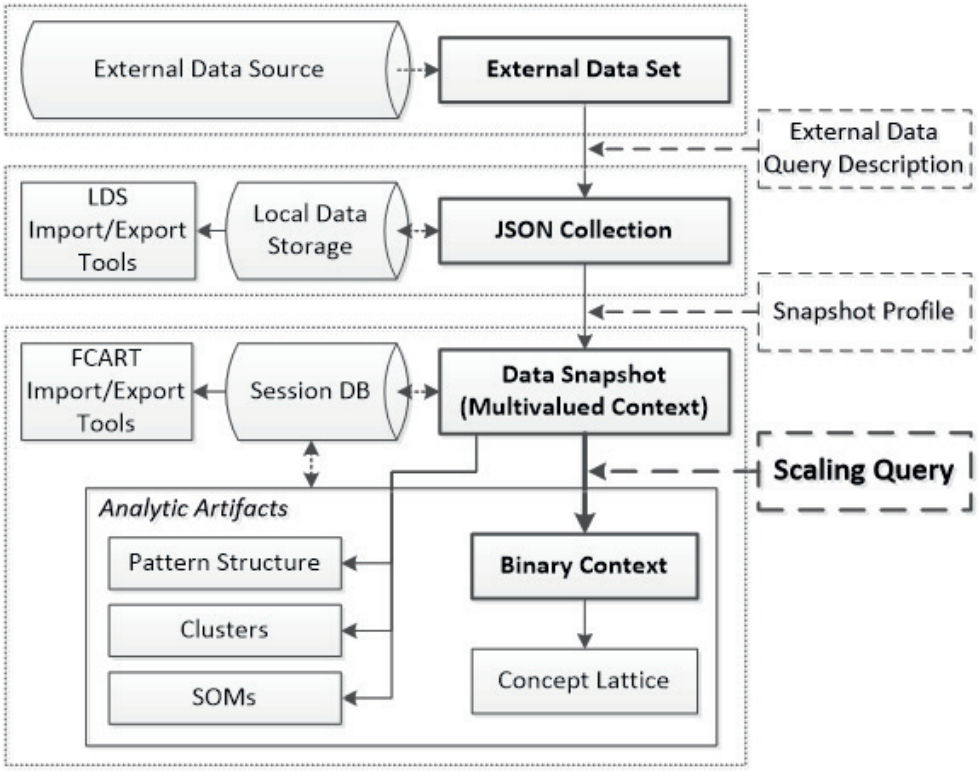

Fig. 1. Main FCA workflow in FCART

2.2. Local Data Storage

FCART Local Data Storage (LDS) plays important role in effectiveness of whole data analysis process because all data from external data storages, session data and intermediate analytic artifacts saves in LDS. Traditional relational databases are not convenient for fast processing, for example, of big amounts of unstructured textual datasets with metadata. In last decade for storing, retrieving and managing big amounts of textual data successfully used document-oriented databases operating with documents in XML or JSON format. XML format is complex and relatively hard to process at the same time. JSON is lightweight and much simpler to use. FCART uses JSON format internally as a main document format.

There are several NoSQL database types: key-value, column-oriented, document-oriented (there are others, but these are the main ones).

1. Key-value systems basically support get, put, and delete operations based on a primary key.
2. Column-oriented systems store data by column as opposed to traditional row-oriented databases. This makes aggregations much easier and faster.
3. Document-oriented systems store structured “documents” such as JSON or XML but have no joins (joins must be handled within your application). It's very easy to map data from object-oriented software to these systems.

For NoSQL databases there are three primary concerns: consistency, availability, and partition tolerance[4].

- Consistency means that each client always has the same view of the data.
- Availability means that all clients can always read and write.
- Partition tolerance means that the system works well across physical network partitions.

Consistent, Available (CA) systems have troubles with partitions and typically deal with it with replication. Examples of CA systems include: Traditional RDBMSs (like Postgres, MySQL, etc), Vertica (column-oriented), Aster Data (relational), Greenplum (relational)

Consistent, Partition-Tolerant (CP) systems have trouble with availability while keeping data consistent across partitioned nodes. Examples of CP systems include: BigTable (column-oriented/tabular), Hypertable (column-oriented/tabular), HBase (column-oriented/tabular), MongoDB (document-oriented), Terrastore (document-oriented), Redis (key-value), Scalaris (key-value), MemcacheDB (key-value), Berkeley DB (key-value)

Available, Partition-Tolerant (AP) Systems achieve "eventual consistency" through replication and verification. Examples of AP systems include: Dynamo (key-value), Voldemort (key-value), Cassandra (column-oriented/tabular), CouchDB (document-oriented), SimpleDB (document-oriented), Riak (document-oriented)
Other FCART components access LDS through a web-service on top of MobgoDB[5].

Despite on trouble with availability MongoDB is the best choice for several reasons.

1. FCART intended to work with big collections of text fragments and complex structured objects.
2. Any structured data can be transformed into JSON document without information loss.
3. There are many third-part tools for querying and analyzing stored data.

Inserting to LDS from external data source can be done by sending to the web-service in the http request elements of "external data query description" (EDQD). EDQD query consists from:

1. Id.
2. Type ("FS Folder", "FS File", "SQL Source", "REST JSON WS", "SOAP WS", etc.).
3. URL with a protocol and local path.
4. ConnectionString.
5. Native Query.

There are specific "visual browsers" which can be used by FCART to build EDQD and load data from external data storage to the LDS. At this moment LDS was tested for storing up to 1 TB data.

The http-request to web-service constructed from two parts: prefix part and command part. Prefix part contains domain name and local path (e.g. http://zeus.hse.ru/lds/). Command part describes what LDS has to do and represents some function of web-service API. For example, http-request "http://zeus.hse.ru/lds/<DbName>**.iter**|NUMBER**:**NUMBER" gets one element or slice from the collection.

3. FCA workflow step by step

3.1. Filling local storage

Local storage of FCART can be filled from various data sources. System supports SQL, XML and JSON sources, so it can load data from most databases and web-services. For describing specific query to data source, we use EDQD. EDQD became a part of collection metadata. Here is the algorithm of web-service (WS) operation (See WS definition in the previous section):

1. User of FCART sends http-request with EDQD elements to the WS.
2. WS parses http-request. If the request is valid then WS will extract the native query and will send it to the specific data storage.
3. If WS successfully gets data from specific data storage by native query then WS will create collection of JSON documents and will add metadata to this collection.
4. WS adds information about success or fail to the technical part of LDS.
5. WS sends response to the user.

3.2. Taking a data snapshot from local storage

Data snapshot (or simply "snapshot") is a result of querying data from local storage. Snapshot is represented as a set of records with structured (atomic and complex) and unstructured (long text) fields. Snapshot is described by a set of metadata: snapshot profile, local path, link to external data source, time stamp, author, and comment. Profile contains definitions of all fields. It can be treated as a "type" of snapshot.

Each field is defined by the following main properties.

- Field Id (unique identifier of the field).

- Path (path in initial XML or JSON definition of collection element – may be empty).
- Name (user-friendly name of the field).
- Group (for visual grouping of fields).
- Comment (any – may be empty).
- Data type (Boolean / Integer / Float / Text / Binary / DateTime).
- Is Unstructured? (field can be interpreted as unstructured text, if this is true then field definition can contain additional properties).
- Is Multivalued? (field is a set or an vector of atomic values, if this is true then field definition can contain additional properties).

FCART provides one with a snapshot profiles editor (SHPE). Fig. 2 shows a variant of SHPE for XML/JSON filtering. The left pane called "XML/JSON Structure" displays a sample of first JSON-document from a LDS collection. A user can select any element from the document, add it to profile as a new field and set additional properties of the profile field (right pane).

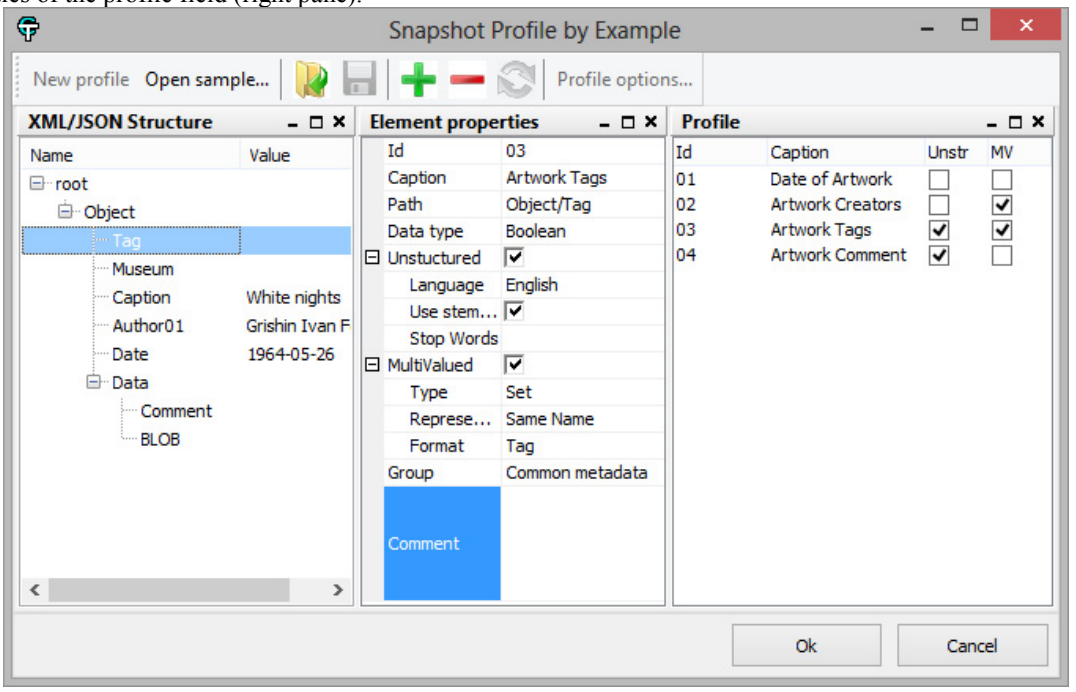

Fig. 2. Snapshot Profile Editor by example of first element in LDS collection

Multivalued field definition additionally contains the following properties.

- The significance of the order of values (Set / Vector).
- Type of multivalued presentation ("delimited content" / "same path" / path in form of "name + number").
- Format of tags and delimiters (separators).

Consider the following example of XML fragment.

```
<Data>
    // ...
  <Genre>Lounge</Genre>
  <Genre>Easy listening</Genre>
    // ...
</Data>
```

In this example field "Genre" is multivalued and have multivalued presentation type "same path" (Path="<Data>/<Genre>"). But in other source we can have type "name + number" (Path="<Data>/<Genre%d>").

```
<Data>
    // ...
  <Genre01>Lounge</Genre01>
  <Genre02>Easy listening</Genre02>
    // ...
</Data>
```

Unstructured field definition additionally contains the following properties.

- Language (main language of text).
- SW (list of stop words).
- Stemmer (not required in current version because we use snowball stemmer from Lucene).

It is very useful for dealing with dynamic data collections, including texts in natural language, and helps to query full-text data more effectively. There is a sample of unstructured and multivalued field descriptions in JSON format:

```
[ …
{       "Id": "03",
        "Path": "object/tag",
        "Caption": "Artwork Tags",
        "Group": "Common metadata",
        "Comment": "The set of tags",
        "DataType": "Text",
        "Unstruct": {       "Is": true,
                            "Language": "English",
                            "StopWords": [],
                            "Stemmer":      "Snowball"
        },
        "MV": {  "Is": true,
                 "MVType": "Set",
                 "MVRepresentation": "SameName",
                 "MVFormat": "tag" }
},
{       "Id": "04",
        "Path": "object/data/Comment",
        "Caption": "Artwork Comment",
        "Group": "Common metadata",
        "Comment": "",
        "DataType": "Text",
        "Unstruct": {       "Is": true,
                            "Language": "English",
                            "StopWords": [
                            "a", “an”, "the"      ],
                            "Stemmer": "Snowball"
        },
        "MV": {  "Is": false,
                 "MVType": "",
                 "MVRepresentation": "",
                 "MVFormat": ""     }
} ]
```

3.3. Scaling queries for snapshots

The system has “Scaling query language” for transforming snapshots into binary formal contexts. Scaling query is a phrase on formal language based on XML syntax. This language describes so-called “rules”. Main rule types are the following:

- *Simple rule* generates one attribute from atomic fields of a snapshot. This rule type has syntax very similar to SQL WHERE clause
- *Scaling rule* generates several attributes from atomic fields based on nominal or ordinal scale
- *Text mining rule* generates one attribute from unstructured text fields.
- *Multivalued rule* generates one or many attributes from multivalued field (arrays and sets)
- *Compound rule* merges rules of all types into a single rule. This rule uses standard logical operations and brackets to combine elements.

There are additional rule types: *Temporal rule* is used for manipulating date and time intervals and *Filter* is used for removing objects with their intents from final context.

In most cases, it is not necessary to write a query from scratch. One can select some entities in *Rules DB* (RDB) and automatically generates a query. It is possible because the RDB is aware of dependencies between rules. Each rule type has XML presentation, so every query (or full RDB) can be imported and exported as an XML-file.

FCART uses Lucene full text search engine[6] to index the content of unstructured text fields in snapshots. The resulting index is later used to validate very quickly whether the text mining or compound rule returns true or false. The following XML fragment is a sample of *text term* from *Text mining rule*. Here we can see the list of synonyms with parameter “proximity” (to find words are a within a specific distance away).

```
<term name="hey how are" id="t21934">
  <list>
    <searchterm proximity="3">hey how are</searchterm>
    <searchterm proximity="3">hey how r</searchterm>
  </list>
</term>
```

The following XML fragment is a sample of the scaling rule.

```
<scale name="Age" ScaleType="Order" DataType="Integer" Ends="Open" id="t34">
  <Offset1>8</Offset1>
  <Offset2>16</Offset2>
  <Offset3>35</Offset3>
  <Offset4>60</Offset4>
</scale>
```

The application of this rule to snapshot generates 5 binary attributes: 1) “Age<8”, 2) “8<=Age<16”, …, 5) “60<=Age”.

3.4. Working with formal concept lattice and other FCA artifacts

The concept lattice visualizer in FCART is a valuable interactive analytic tool. It can be used to browse the collection of objects with binary attributes given because of query to snapshot (with structured and text attributes). The user can select and deselect objects and attributes and the lattice diagram is modified accordingly. The user can click on a concept and drag it. The screen shows in a separate window names of objects in the extent and names of attributes in the intent of the concept. Names of objects and attributes are linked with initial snapshot records and fields. If the user clicks on the name of an object or an attribute, the content of the object or attribute description is shown in a separate window according to snapshot profile. There is also a *navigator* window to travel through children and parents of the concept.

The user can customize settings of lattice browsing in various ways. He can specify whether nodes corresponding to concepts show numbers of all (or only new) objects and all (or only new) attributes in extent and intent respectively, or names of all (or only new) objects and all (or only new) attributes. Separate settings can be specified for the selected concept, concepts in the order filter, and the remainder of the lattice. The visual appearance can be changed: zooming, colouring, and other tools are available.

Right clicking on the name of an attribute user can choose several options: he can build a sublattice containing only objects with selected attribute; build a sublattice containing only objects without selected attribute; or find the highest concept with selected attribute. Right clicking on the name of object allows the same actions.

Additional menu allows building several types of sublattices from current concept lattice. The multiple-document interface allows us to inspect several artifacts, so a sublattice will be opened in a new window.

4. Concept stability analysis

We introduce the new addition to FCA workflow as a subsystem to work with formal concept indices (See Fig. 3). Indices discussed below are implemented as scripts of type “Concept numerical property”.

4.1. Stability index

The stability index describes the proportion of subsets of objects of a given concept whose closure is equal to the intent of this concept[7,8]. In other words, it is meant to capture how much concept intent depends on particular objects of the extent: should some objects be removed from the concept extent, would the concept intent remain the same? In an extensional formulation, the index specifies how much a concept extent depends on intent attributes. We thus distinguish between intensional and extensional stability indices σ_i and σ_e:

$$\sigma_i(A,B) = \frac{\left|\{C \in A | C' = B\}\right|}{2^{|A|}}, \quad \sigma_e(A,B) = \frac{\left|\{D \in A | D' = A\}\right|}{2^{|B|}} \tag{2}$$

The intent of a concept with a high intensional stability index would be likely to exist even if we ignore several of its objects: it does not disappear if the intent of some of its objects is modified, e.g., if these objects lose some of the properties of this concept. Put differently, concepts relying on noisy objects and, therefore, not typical of a realistic category, are more likely to be unstable. Similarly, extensional stability helps isolating concepts that appear because of noisy attributes.

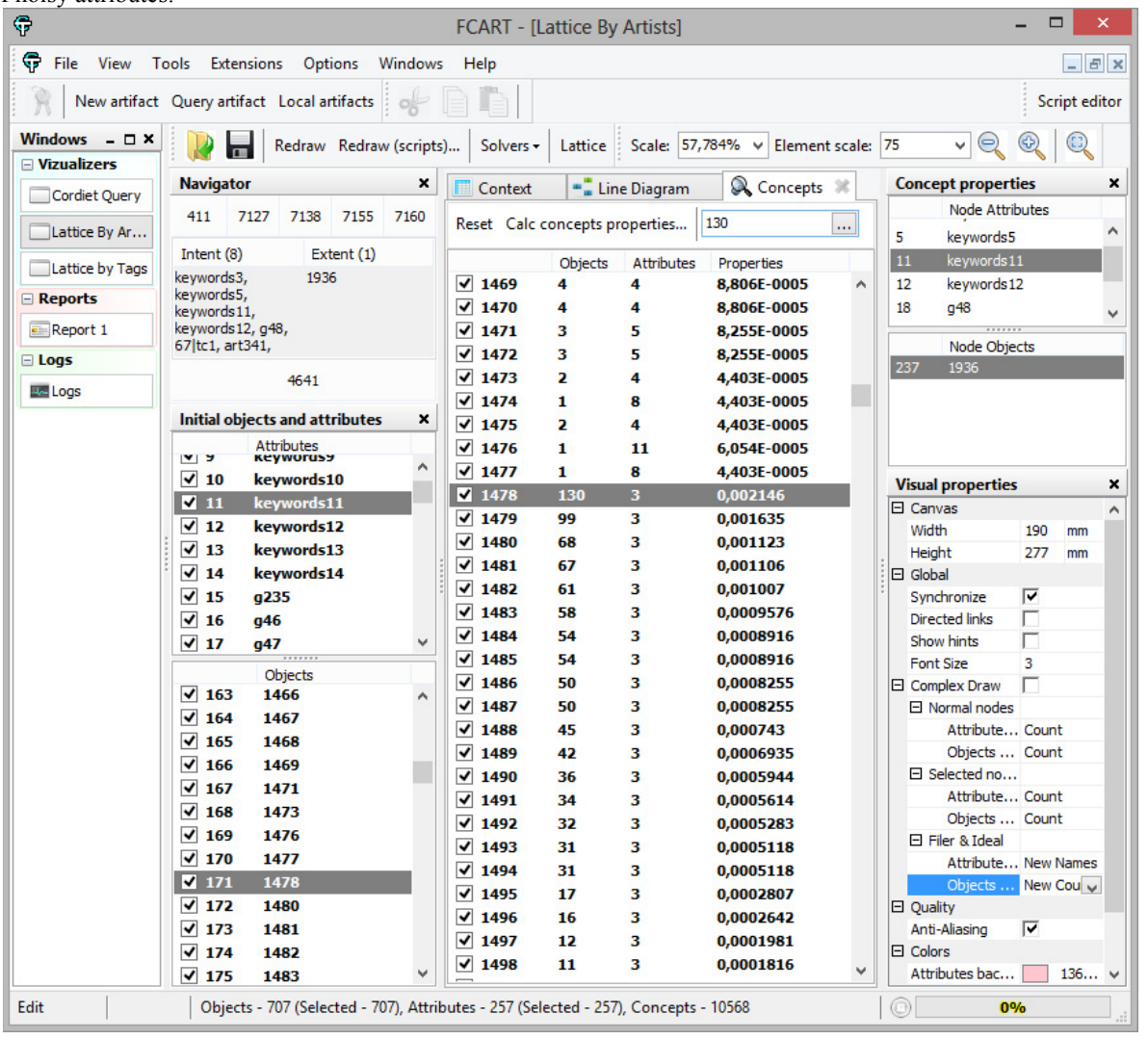

Fig. 3. Stability Index as a property of concept in the list of all concepts

4.2. Concept probability

A concept that covers fewer objects is normally less intensionally stable than a concept covering more objects. Still, these rather specific concepts can correspond to interesting associations that should not be ignored in the analysis. To give such specific concepts a chance of surviving the stability test, we need to normalize the stability

index. To this end, we can use the notion of concept probability. The idea is that if a concept has low probability, but is still observed in the data, it may reflect an interesting dependency and should be taken into account.
The notion of concept probability essentially follows a relatively similar line of reasoning. For $m \in M$, denote by p_m the probability of an arbitrary object having the attribute m. For $B \subset M$, define p_B, the probability of an arbitrary object having all attributes from B, by

$$p_B = \prod_{m \in B} p_m \tag{3}$$

thus assuming the mutual independence of attributes. By denoting $n = |G|$, we obtain the following formula for the probability of B being closed:

$$p\left(B = B''\right) = \sum_{k=0}^{n} p\left(\left|B'\right| = k, B = B''\right) = \sum_{k=0}^{n}\left[\binom{n}{k} p_B^k \left(1 - p_B\right)^{n-k} \prod_{m \notin B}\left(1 - p_m^k\right)\right] \tag{4}$$

To see the reasoning behind the formula, note that, for |B'| = k and B = B'', we need that

1. There are k objects that have all attributes from B;
2. Each of the other n – k objects does not have at least one attribute from B;
3. No attribute outside B belongs to all the k objects.

Again, one can regard this as "intensional probability" of a concept and define "extensional probability" dually (as the probability of an object subset being closed).

4.3. Separation

The separation index[9] is meant to describe how well a concept sorts out the objects it covers from other objects and, jointly, how well it sorts out the attributes it covers from other attributes of the context. Put differently, it indicates the significance of the difference between the objects covered by a given concept from other objects and, at the same time, between its attributes and other attributes. To do so, the separation index s is defined as the ratio between the area covered in the context by a concept *(A; B)* and the total area covered by its objects and attributes:

$$s\left(A, B\right) = \frac{\left|A\right|\left|B\right|}{\sum_{g \in A}\left|g'\right| + \sum_{m \in B}\left|m'\right| - \left|A\right|\left|B\right|} \tag{5}$$

4.4. Identical denotations

Another possible type of filtering is related to finding out and combining with each other group of objects which refer to the same object of the real world. Such objects and their descriptions called identical denotations.

After building the set of formal concepts it is necessary to distinguish formal concepts having an extent which includes only identical denotations.

Building indices we have to consider all main features of these concepts. At first, index must increase if number of attributes which are different for objects in a concept is decreasing and all other things being equal. To account for this feature one can use the following index:

$$I_1(A, B) = \frac{|A||B|}{\sum_{g \in A}\left|\{g\}'\right|} \tag{6}$$

The second feature that must have an index – it must increase when number of common attributes is increasing and all other things being equal. As a result one can construct the following index:

$$I_2(A,B) = \sum_{m \in B} \frac{|A|}{\left|\{m\}'\right|} \tag{7}$$

It is easy to see that the appearance of a new attribute in the intent of a formal concept (other things being equal) augments the index value. Here, the more objects in the context share this attribute the less the changes of the index value are.

The final index DII^{10} is a combination of these two indices. Absolute values of the coefficients have an influence only to the value of a threshold and filtering quality depends on the coefficients correlation in the index formula. So we can specialize indices family without loss of the optimum and take 1 as a value for the one of the coefficients:

$$DII_+ = I_1 + kI_2, \quad DII_* = I_1 * I_2^{\,k} \tag{8}$$

Conclusion

Formal Concept Analysis Research Toolbox is a powerful software system being in active development. The release of the version 1.0 of FCART is planned for July 2014. In this article, we considered in details some new additions to the toolset of the system. New functions on each stage of FCA workflow (from querying external data sources to calculating formal concept indices) are suitable and improve the overall performance of the analysts.

Acknowledgements

The work of the authors on the project “Mathematical Models, Algorithms, and Software Tools for Intelligent Analysis of Structural and Textual Data” was supported by the Basic Research Program of the National Research University Higher School of Economics.

References

1. 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.
2. Poelmans J., Elzinga P, Neznanov A.A., Viaene S., Kuznetsov S.O., Ignatov D., Dedene G. Concept Relation Discovery and Innovation Enabling Technology (CORDIET) // CEUR Workshop proceedings Vol. 757, Concept Discovery in Unstructured Data, 2011.
3. Ganter B., Wille R. Formal Concept Analysis: Mathematical Foundations, Berlin, Springer, 1999.
4. Brewer E. Towards robust distributed systems: Principles of Distributed Computing, Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, 2000.
5. MongoDB (http://www.mongodb.org)
6. Apache Lucene (http://lucene.apache.org)
7. Kuznetsov S.O. On stability of a formal concept. Annals of Mathematics and Artificial Intelligence, Vol. 49, 2007. pp. 101-115.
8. Kuznetsov S.O., Obiedkov S.A., and Roth C. Reducing the Representation Complexity of Lattice-Based Taxonomies. Proc. 15th International Conference on Conceptual Structures (ICCS 2007), Lecture Notes in Artificial Intelligence (Springer), Vol. 4604, pp. 241-254, 2007.
9. Klimushkin M.A., Obiedkov S.A., and Roth C. Approaches to the Selection of Relevant Concepts in the Case of Noisy Data, Proc. 8th Int. Conf. on FormalConcept Analysis, ICFCA_2010, 2010, pp. 255-266.
10. Ilvovsky D.A., Klimushkin M.A. FCA-based Search for Duplicate Objects in Ontologies, Proceedings of the Workshop Formal Concept Analysis Meets Information Retrieval, Vol. 977. CEUR Workshop Proceeding, 2013.