Generic Tools for Data Analysis and Visualisation

Uta Priss

Edinburgh Napier University, School of Computing,
www.upriss.org.uk

Abstract. This position paper discusses challenges that need to be overcome in order to build generic tools for data analysis and visualisation. Its intention is to stimulate discussion among the CUBIST workshop participants, not to present results.

In the Star Trek television programs, data analysis is usually accomplished by speaking to the computer and asking it to analyse some problem. The computer then provides a succinct, coherent and relevant analysis of the data which allows the Star Trek crew to make their decisions. In reality, although humanity has made some progress with implementing Star Trek technology[1], achieving automated computerised data analysis is probably at least as difficult as implementing automated natural language processing because the computer would need to understand the problem within its full context. A more achievable but still difficult goal would be to build data analysis software that collaborates with human users during the data preprocessing and modelling stages and then automates the rest of the analysis. In recent years, great advances have been made with respect to the availability of large toolkits for data analytical methods, visualisation, storage and retrieval. Thus, although the general problem is difficult (or impossible), many of the building blocks for achieving somewhat more modest solutions are available, even using low cost or free, open source tools.

With respect to using Formal Concept Analysis (Ganter & Wille, 1999) as a tool for data analysis, drawing from our own experience in past projects (Priss & Old, 2010), the most labour-intensive part of using FCA is usually the preprocessing stage during which one builds the formal contexts from the raw data or during which one decides how to select smaller data sets if the original data is too large to be visualised in a single lattice. In our experience each new data set provides new challenges. One usually has to write scripts or use other computational means to preprocess the data. It is not always possible to reuse methods (or scripts) from one project directly for the next one. In some cases (Endres et al 2010), a custom application has to be purpose-built for the data. Ideally, there should be methods and tools which speed-up the data preprocessing stage. It would be nice if it was possible to apply FCA quickly to any new data set that one encounters in order to explore the data. So, with respect to FCA, the general problem of building a generic data analysis tool can be scaled down to the problem of building a generic data preprocessing tool which makes it easier to apply FCA to any

[1] For example, we finally have Star Trek's PADDs in the form of modern tablet computers, such as Apple's iPad, and there is a list of further "Star Trek Technologies that Actually Came True" available at http://electronics.howstuffworks.com/10-star-trek-technologies.htm.

given data set. Additionally, it would be desirable if FCA tools could be more easily integrated with existing tools for data preprocessing, mining, extraction, modelling and so on to allow for a combination of methods.

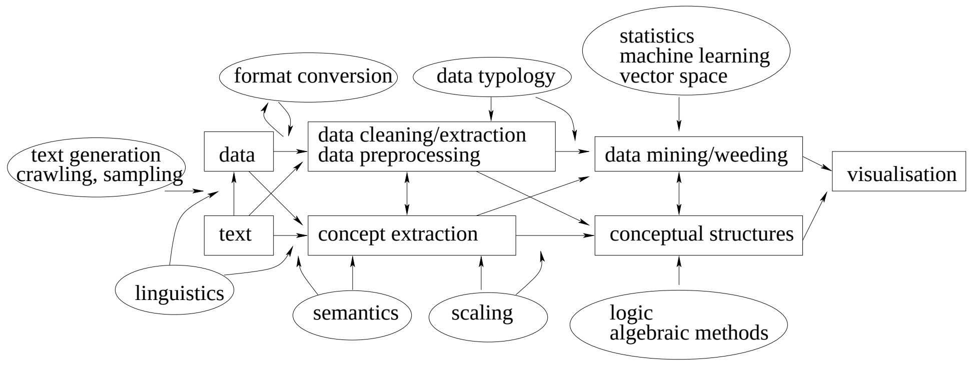

Fig. 1. Components of a generic tool for data analysis and visualisation

The purpose of this position paper is to provide an overview of the challenges that need to be overcome in order to build such generic tools for data analysis (both FCA-based ones and tools where FCA is just one component among many others). Figure 1 provides an overview of the steps, tasks and tools that we envisage as building blocks for generic data analysis tools. In the case of FCA, concept extraction, conceptual structures and visualisation are meant to refer to the corresponding FCA techniques. In the general case, conceptual structures could be represented using ontologies or other formal methods and visualisation could be any commonly used method (graph- or network-based, statistical plots or charts, 3-d visualisation, and so on). Many parts of Figure 1 are well established and supported by many existing tools (for example, tools for linguistic parsing and stemming, statistics, data mining and machine learning). In some cases, tools cover a variety of areas. For example, data mining tools tend to provide methods for data preprocessing and often have plugins for some linguistic processing. A tool such as NLTK[2] provides linguistic methods as well as some machine learning methods. A tool such as Sage[3] combines a wide range of mathematical methods (including statistics, network modelling and mathematical plotting) in a single toolbox.

But we believe that there are also some aspects of Figure 1 that are still missing and pose challenges, in particular:

Text generation, crawling, sampling: In modern applications text is not just extracted from existing databases and document collections but also generated on demand from web-based or other continually updated sources. This area is probably most difficult to automate because it heavily depends on the structures of the textual data. Semi-

[2] http://www.nltk.org/
[3] http://www.sagemath.org/

structured text (such as XML, linked data) is easier to process than unstructured text. It is easier to look for specific patterns than to discover previously unknown structures.

Text/data: In this paper, "data" denotes text that is slightly more structured (using mark-up, database or spreadsheet tables, linked data, etc) whereas text can be in any format or medium. The main challenges with respect to the data itself are the amount (for example, processing all of Wikipedia would require giga- or terabytes of space) and internationalisation. For example, although Unicode is an international coding standard, in our experience, it can still pose problems because not all software supports it perfectly. Unexpected effects can occur. For example, printing a mixture of characters from languages that write right to left and those that write left to right can confuse printed output. Some major languages (such as Chinese) are often written in non-Unicode encodings.

Format conversion: Many tools for format conversion exist[4] but not all formats tend to be supported and errors may be introduced in the conversion process. For example, even though Weka is a very popular data mining toolkit[5] and data is commonly stored in spreadsheet or csv formats, importing such formats into Weka's internal format can introduce errors because, for example, leading zeros in string data are automatically deleted.

Data cleaning: Data cleaning is often discussed in a database context, probably because databases provide rules for consistency and integrity checks. In a more general context, some form of conceptual modelling is required in order to determine what constitutes an error.

Data typology, scaling: The idea for data typology or scaling is that once a datatype or conceptual type is established it should predict the kind of analyses that are suitable for the data. Commercial tools often provide "Wizards" that help users with modelling decisions, but this is less supported in free tools or tools that have more general functionality.

Data weeding: We see a difference between mining and weeding (Priss & Old, 2011) in that mining explores all of the data simultaneously whereas weeding allows for a careful (concept-guided) selection of subsets of the data.

Selection of programming language: A promising programming language for toolkits of mathematical, mining and machine learning software is currently Python. Although Python is a scripting language, more complex algorithms and treatment of large data sources can be accomplished by writing relevant routines in C or C++ which are accessed by Python scripts. Unfortunately, because Python does not have a standard graphical component, different visualisation software requires different additional graphical software which can make tools difficult to install. The main software for graphical, GUI applications is probably Java. Scripting is easier with Python than Java. Both Python and Java are cross-platform but Python is probably more suited for Unix than PCs.

[4] For FCA these are tools such as FcaStone (http://fcastone.sourceforge.net/), FcaBedrock for reading csv files (http://sourceforge.net/projects/fcabedrock/) and ToscanaJ for database connections (http://tockit.sourceforge.net/).

[5] http://www.cs.waikato.ac.nz/ml/weka/

Cross-disciplinary approaches: Toolkits often combine software at a syntactic level but not necessarily in a semantically consistent manner. For example, an analysis of using FCA and Sage (Priss, 2010) demonstrates that although Sage allows to combine a variety of tools with FCA software, coding is still required to model the data appropriately for each tool. Also different tools in a single toolkit can be of varying quality. Approaches from other disciplines are sometimes missing.

Testing, validation, evaluation: Methods and standards need to be established that allow for comparison and evaluation of data analysis methods across a variety of disciplines. In particular, it would be interesting to discuss the methods and techniques provided by FCA that are not already available through more traditional methods. Evaluation and testing must involve both the domain experts and the tool builders.

Usability and learning curve: Tools must be usable, well documented and fairly easy to learn in order to attract sufficient users.

In summary, modern toolkits (such as Sage and NLTK) are a good starting point for building generic data analysis and visualisation tools. But there are numerous challenges that need to be overcome in order to truly integrate a large variety of methods in a manner that renders them widely applicable. So far FCA software does not appear to be integrated into any of the existing toolkits, but integration of FCA and Sage, for example, can be accomplished (Priss, 2010).

References

1. Endres, Dominik M.; Foldiak, Peter; Priss, Uta (2010). *An Application of Formal Concept Analysis to Semantic Neural Decoding.* Annals of Mathematics and Artificial Intelligence, 57, 3, Springer-Verlag, 2010, p. 233-248.
2. Priss, Uta; Old, L. John (2010). *Concept Neighbourhoods in Lexical Databases.* In: Kwuida; Sertkaya (eds.), Proceedings of the 8th International Conference on Formal Concept Analysis, ICFCA'10, Springer Verlag, LNCS 5986, p. 283-295.
3. Priss, Uta (2010). *Combining FCA Software and Sage.* In: Kryszkiewicz; Obiedkov (eds.), Proceedings of the 7th International Conference on Concept Lattices and Their Applications (CLA'10), 2010, p. 302-312
4. Priss, Uta; Old, L. John (2011). *Data Weeding Techniques Applied to Roget's Thesaurus.* In: Knowledge Processing in Practice. Springer Verlag, LNAI 6581, p. 150-163.
5. Ganter, Bernhard, & Wille, Rudolf (1999). *Formal Concept Analysis. Mathematical Foundations.* Berlin-Heidelberg-New York: Springer.