implications on the actual work carried out. Considering the know-how of researchers and their organizational attachment to either Humanities or Computer Science departments, their research can either be more focused on just the creation and use of digital repositories, or on real program development in the Humanities as an area of applied Computer Science.

A practical consequence also in organizational terms of this way of looking at things would be to set up research groups in both scientific communities, Computer Science and Humanities. The degree of mutual understanding of research issues, technical feasibility and scientific relevance of research results will be much higher in the area of overlap between the Computational and Digital Humanities than with any intersection between Computer Science and the Humanities.

3.2 Design Principles for Transparent Software in Computational Humanities

Chris Biemann (TU Darmstadt, DE)

License © Creative Commons BY 3.0 Unported license
© Chris Biemann

Abstract. In this short statement, the importance of transparent software for humanities research is highlighted. Here, three dimensions of transparency are identified: First, software should be freely available so that results are reproducible. Second, software should be easy to use and hide complex underlying algorithmics from the user. Third, to avoid a black box situation where the software’s decisions are opaque to the user, the reasons for any of the automatically produced statements should be traceable back to the data they originated from. After elaborating on these principles in more detail, they are exemplified with a basic distant reading application.

Introduction. The newly emerging field of Computational Humanities (CH) is situated at the interface between humanities research and computer science. Research questions in CH are concerned with aspects of both fields: in Digital Humanities (DH) research, computational aspects either not considered relevant or are merely assigned a subordinated role, while in computer science, research on computational methods and algorithmic approaches is rather detached from their application domain – e.g. the field of Machine Learning produces
methods that learn from data, no matter what kind of data it is. In contrast to this, CH considers humanist’s questions and computational challenges both as first-class citizens, and focuses on their interplay. Whereas in both Computational and Digital Humanities, software solutions are needed that support the humanist – typically in accessing electronically available data in her respective field of study – CH research is also concerned with further automatizing the analysis using novel algorithmic approaches. As opposed to generic computer science approaches, however, algorithms in CH software are additionally required to be comprehensible by human(ist)s, in order to be open for scrutiny to allow for a depth of analysis that is satisfactory for the humanities. With respect to these prerequisites, a number of requirements on the software can be deduced. These will be subject of the following section, which discusses three dimensions of transparency that CH software should have in order to be a suitable tool for CH research. On a related topic, but written from the perspective of Computational Linguistics, see Pedersen (2008).

Transparency of Software for Computational Humanities. The term ‘transparency’ can be defined in organizational contexts as ‘the perceived quality of intentionally shared information from a sender’ (Schnackenberg & Tomlinson, 2014) and implies openness, communication and accountability. In this section, these facets of transparency are elaborated on and put forward as desired properties of software used in Computational Humanities research.

Open Source for Reproducibility. Whether hypotheses are merely empirically verified on data that has been mined by computational approaches, or hypotheses are generated from empirical observations in the first place: research in CH inherently includes empirical aspects, and rational deduction is complemented by a certain amount of experimentation. As in the experimental sciences, such as e.g. Physics, empirical investigations in CH must be reproducible to adhere to scientific standards. Just as it is considered bad science in the field of computational linguistics to rely on commercial search engines for data acquisition and statistics (Kilgarriff, 2006) because their inner workings are secret and they change over time, the CH researcher should not rely on commercial software with closed sources for the same reason. Rather, software in CH and other research contexts should be available open source in versioned public repositories, and the version of the software should be included in the description of the experimental setup. In this way, subsequent research is able to reproduce prior experiments of others and the inner workings of the software are fully transparent, at least for those that can understand computer programs. A further advantage of open source software over proprietary software, especially when distributed under a lenient license, is the possibility for subsequent research to combine several existing software into more advanced and more complex software without having to re-implement already existing methods.

Intuitive Interfaces and Hiding Complexity. Just as in communication between humans, communication, i.e. human-computer interaction, happens when a CH researcher uses CH software. And just as successful fact-oriented communication between humans just provides enough detail to communicate the intended amount of information, supportive software should be intuitive to operate and hide unnecessary complex aspects from the user. For this, design principles of graphical user interfaces should be adhered to, and e.g. developed according to the visual analytics process (Keim et al., 2010). Abstracting from complexity, however should not be confused with obfuscation – while it is necessary for the acceptance of the software and its methods that algorithmic results are easy to obtain without necessarily understanding the algorithmic details, it is still crucial that the implementation of such details are transparent (cf. Section 2.1) and the algorithmic decisions are backed up by access to the data that leads to these decisions (cf. Section 2.3). Only in this way, the CH researcher
can build trust in her algorithmic methodology and develop an intuition about its utility and potential. A result of a successful CH research is always twofold: an algorithmic method and/or a mode of its application that allows to easily analyze data from the humanities, and a result in humanities research obtained with the help of such method.

Accountability and Provenance. The most precise automatic result will still be subject to doubts and disbelief by human experts, as long as no explanation is provided how the automatic method arrived at such result. As mentioned in the previous section, in order for a method to be trusted, it needs to provide the possibility to drill down into the details of its decision-making process, to be fully accountable and to provide a fully transparent reason why the method arrived at a particular result, which is in software development known as data provenance (cf. W3C.org, 2005; Simmhan et al., 2005). In the context of CH, data provenance means not only to store and use algorithmic derivations of the input data (such as e.g. the number of times a certain term appears in texts of a certain time span), but also the sources from which these derivations were derived from (i.e., pointers to the positions in the documents where the term appeared) and a way to access them via the user interface. Data provenance enables the researcher to judge the software’s decisions and to accept or discard algorithmically found evidence.

CoocViewer – a Distant Reading Tool. In this section, we discuss CoocViewer (Rauscher et al., 2013), a simple tool for distant reading, along the three facets of transparency as outlined above. CoocViewer is an Open Source tool that allows browsing of statistically extracted networks of terms (cf. Quasthoff et al., 2006) extracted from corpora in the format of significant concordances. Figure 2 shows significant concordances for the term ‘bread’. The complexity of the computation of such concordances and details of the concordance are abstracted; the user only notices the most significantly co-occurring terms, for example ‘butter’ located two positions to the right of ‘bread’. To investigate this connection, the user can click on the link and drill down into all 20 references that lead to the link, as shown on the right side of the figure: CoocViewer provides full data provenance by showing – on demand – detailed information about single word frequencies and the references, including document titles and page numbers.

While not being a very complex example, CoocViewer adheres to the three design principles of transparency for CH software. Additionally, it enables the import and export of data in various formats for improved usability. During its development, several measures of significance, which determine the related terms shown as most significant concordances, have been examined to investigate computational aspects of distant reading. The tool was productively used in quantitative literary analysis of crime novels, see (Rauscher, 2014).
Conclusion. This short statement laid out design principles for the transparency of software for the computational humanities. Three important facets of transparency were identified that are desirable for software in the field of Computational Humanities: open source codebases for reproducibility, intuitive interfaces for effective communication between user and software, and data provenance for accountability and to build trust in algorithmic methods. These facets were exemplified on CoocViewer, a distant reading tool that adheres to these principles. Creating software to answer research questions in humanities research and computational research alike is one of the main aspects of the field of Computational Humanities. Adhering to the design principles of transparency, as discussed in this statement, enables a firm basis for reproducible research, the exchange of techniques and components, and the credibility of results through data provenance. Thus, not only the source data should be available freely to other researchers, but also the software that allows us to produce scientific results in the field of computational humanities.

References

3.3 On Covering the Gap between Computation and Humanities
Alexander Mehler and Andy Lücking (Goethe-Universität Frankfurt am Main, DE)

Since digital or computational humanities (CH) has started its triumph in the humanities’ research landscape, it is advisable to have a closer look at its methodological and epistemological range. To this end, we look at CH from the point of view of preprocessing, machine learning, and the general philosophy of science and experimental methodology. From this perspectives, a number of gaps between CH on the one hand and classical humanities on the other hand can be identified. These gaps open up when considering: (i) the status of preprocessing in CH, its logical work-flow and the evaluation of its results compared to the needs and terminological munition of the humanities. Most importantly, corpus preprocessing often comes before hypothesis formation and respective model selection has been carried out, turning the logically as well as methodologically required workflow upside down. (ii) The predominant role of functional explanations in CH applications vs. the predominant role of intentional explanations with regard to the humanities. While so far computational