Empirical Evaluation of Semi-Automated XML Annotation of Text Documents with the GoldenGATE Editor[1]

Guido Sautter[1], Klemens Böhm[1], Frank Padberg[1], Walter Tichy[1]

[1] Department of Computer Science, Universität Karlsruhe (TH)
Am Fasanengarten 5, 76128 Karlsruhe, Germany
{sautter, boehm, padberg, tichy}@ira.uka.de

Abstract. Digitized scientific documents should be marked up according to domain-specific XML schemas, to make maximum use of their content. Such markup allows for advanced, semantics-based access to the document collection. Many NLP applications have been developed to support automated annotation. But NLP results often are not accurate enough; and manual corrections are indispensable. We therefore have developed the GoldenGATE editor, a tool that integrates NLP applications and assistance features for manual XML editing. Plain XML editors do not feature such a tight integration: Users have to create the markup manually or move the documents back and forth between the editor and (mostly command line) NLP tools. This paper features the first empirical evaluation of how users benefit from such a tight integration when creating semantically rich digital libraries. We have conducted experiments with humans who had to perform markup tasks on a document collection from a generic domain. The results show clearly that markup editing assistance in tight combination with NLP functionality significantly reduces the user effort in annotating documents.

1 Introduction

The digitization of printed literature currently makes significant progress. The Google Libraries Project, for instance, aims at creating digital representations of the entire printed inventory of libraries. Other initiatives specialize on the legacy literature of specific domains, such as medicine, engineering, or biology. While some projects only aim at creating digital versions of the text documents, domain-specific efforts often have more ambitious goals: To make maximum use of the content, text documents are annotated according to domain-specific XML schemas. The XML markup is necessary to access the document collection with techniques that are more sophisticated than keyword search and provide richer semantics. It enables fine-grained searching via XPath or XQuery, mining the document content, and linking the documents.

[1] Work partially supported by grant BIB47 of the DFG.

Manually creating markup for digitized documents is a cumbersome task. While the advances in NLP (Natural Language Processing) help automate the markup process, fully automated markup solely relying on NLP is not feasible, for several reasons: First, if markup quality is a hard requirement, the accuracy of 95-98% provided by up-to-date NLP applications [1] tends to be insufficient. Second, NLP accuracy decreases heavily if the data is noisy [2]. However, noise is common in raw OCR output. Third, if the markup process involves more than one NLP application, errors add up. When five NLP components arranged in serial order build on the output of each other, the overall estimated accuracy is $98\%^5 \approx 90\%$ at best. Only intermediate manual corrections can mitigate this effect. To date, no existing NLP toolkit allows manual editing of the documents. To achieve high markup quality, a user has to save the document after each NLP step and correct it in an XML editor, then apply the next NLP step, and so on. This back and forth incurs considerable effort. In addition, many NLP components do not produce XML, but other formats. Such output becomes editable only after expensive conversions. These problems call for tools that allow users to deploy NLP components and edit NLP output manually. To this end, we have developed the GoldenGATE editor [3]. It provides a slim API for the seamless integration of NLP components, such as automated taggers for locations or taxonomic names [4]. GoldenGATE offers useful features for editing markup that is the output of NLP, e.g., annotating all occurrences of a given phrase in one step. It also provides functions for cleaning up OCR artifacts and for restoring the structure of the original document.

To quantify the benefit of editing assistance and NLP integration, we conducted a *controlled experiment* [5, 6] in which participants were asked to annotate generic documents using GoldenGATE or XMLSpy, a standard pure XML editor. We measured the task completion times and performed a statistical analysis of the time differences between the editors. The experiment shows that a tight integration of editing assistance and NLP reduces the effort for marking up documents. This finding is of interest to a broader audience: In almost any application domain, large document collections need to be digitized and enhanced with semantic annotations.

Paper outline: Section 2 discusses XML editors and relevant NLP tools. Section 3 describes the features and design of GoldenGATE. Section 4 presents the setup and results of our experiment. Section 5 concludes.

2 Related Work

General-purpose text editors like UltraEdit [7] or Emacs provide little XML-specific support, e.g., for inserting tags. Specialized XML editors, like Oxygen [8] or XMLSpy [9], are tailored to handling XML data. They include document validation against DTDs and XML schemas, interpreters for the XPath and XQuery query languages and XSLT, etc. They also alleviate the creation of markup to some extent, but do not give way to any automation. They are not designed to integrate NLP applications either, since NLP has not been a usual part of XML data handling so far.

The OpenNLP [10] project encompasses a multitude of mostly open-source projects concerned with the development of NLP tools, which are heterogeneous

regarding purpose, programming platform, and quality. LingPipe [11] is a professional NLP library. Except for the rule-based tokenization, the analysis functions apply statistical models such as Hidden Markov Models [12]. While its functionality is powerful, LingPipe lacks a user interface: It has to be integrated in other programs to be accessible in ways other than the command line.

The NLP framework GATE [13] offers functionality comparable to OpenNLP, but allows for more complex applications and is capable of producing XML output. It includes Apache Lucene [14] for information retrieval and a GUI for visualization. It is relatively easy to extend with additional components. GATE is dedicated to NLP research and evaluation, rather than document markup and management: It provides functions for assessing markup results obtained with test corpora, but lacks any facility for manual correction of text or markup. Applications similar to GATE are WordFreak [15] and Knowtator [16].

3 The GoldenGATE Editor

In this section, we describe the GoldenGATE editor, which we have designed and built to support annotating text documents. This includes assistance for manual editing (Section 3.1) as well as the seamless integration of NLP components and functionality for correcting NLP output manually (Section 3.2). The development of GoldenGATE is part of a research effort which aims at creating a digital library of biosystematics literature by scanning and marking up the huge body of legacy articles from this domain.

3.1 The Document Editor

In GoldenGATE, a document is displayed and edited in its own document editor (Figure 1), which is a tab in the main window. The editor provides all the functionality required for manually editing both text and markup.

Controlled XML Syntax Generation: If the user has to handle the XML syntax manually – character-wise – this is unnecessarily cumbersome and gives way to syntax errors. Thus, the document editor only allows editing the tag content (i.e., the XML element name, and the names and values of attributes). It generates the syntax (e.g., the angle brackets) automatically and shields it from manual editing. It arranges the tags automatically to enforce wellformedness.

Markup Creation: To mark up a sequence of words with an XML tag, the user can simply select the words in the document and use the *Annotate* function in the context menu. The editor then prompts for the element name and does the rest automatically. To reduce the editing effort further, the context menu pro-vides the most recent element names for instant reuse. Changing the name of an element works similarly, the user does not need to modify start and end tag separately. The same is true for removing the markup around some text, or removing a marked up document fragment, i.e., both the tags and the text enclosed.

Global Markup Editing: Creating, modifying, and removing XML tags often applies to all elements of a certain type. The editor offers support for this, e.g., renaming all XML tags with a certain name, or removing them with or without the text enclosed. For instance, this is useful for removing the `font` tags from HTML-formatted OCR output. Only when marking up a piece of text with an XML tag, the functionality is slightly different: The user can choose to mark up all occurrences of the selected phrase throughout the document instead of just one. This facilitates marking up, say, all the mentions of a person or location in a document with just a single action.

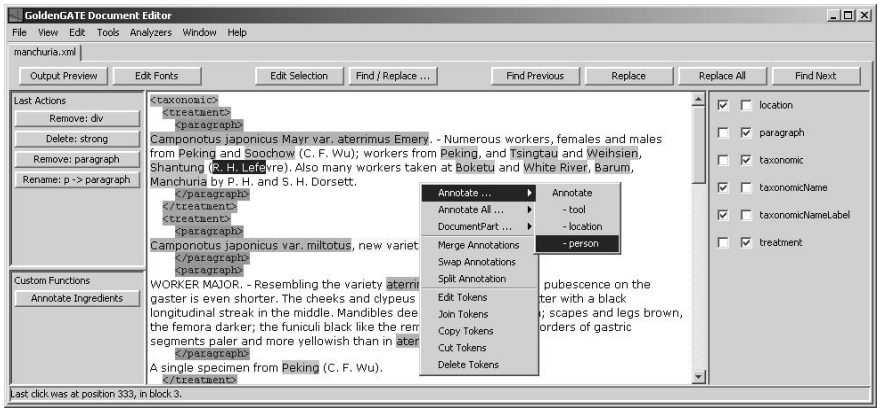

Fig. 1. The annotation editor.

Advanced Search Functionality: In many situations, a user needs to access just certain XML elements, which are spread out over the document. With existing editors, this requires a search for each element. In order to simplify this form of element-specific access, the document editor allows displaying and editing elements with a certain name only. This is useful for, say, checking if the section titles in a document are in the correct case.

Flexible Document Display: If the number of tags becomes too large, the document will not be concise any more, and readability is reduced. Thus, the presentation of the document in the editor is flexible to provide the appropriate level of detail for the current editing activity. In the display control (see Figure 1, to the right of the document), a user may choose to highlight text enclosed by certain tags instead of displaying the tags, or not to show the markup at all.

OCR Cleanup: OCR output documents often contain artifacts that were recognized correctly, but do not belong to the text itself. They rather originate from the print layout, like page numbers and titles. Another problem are line breaks that do not mark the end of a paragraph, but also originate from the print layout. Related to the latter are hyphenated words. These artifacts may compromise NLP result quality severely, and removing them manually is cumbersome. Therefore, the document editor provides a function for resolving hyphenation and removing erroneous line breaks. In particular, resolving hyphenation automatically is far from trivial, since one has to pay attention not to destroy enumerations that use pre- or postfixes as abbreviations.

3.2 Integration of NLP Tools and NLP Correction

As mentioned in Section 2, powerful NLP tools exist. A major difficulty when implementing such a tool is that XML editors work on characters, while NLP components usually work at the word level: NLP regards text as a sequence of tokens, which are atomic units. This results in data models of different granularities, and the mapping between these models is complex. GoldenGATE hardly includes any hard coded NLP functionality, but lets users add arbitrary NLP functionality without difficulty. We have paid much attention to ensuring that the interface to the NLP components is slim.

To facilitate correction of NLP errors, the editor provides specific views on the NLP results. In particular, it can display a list of all XML elements of a certain name. The user can then review all these annotations without having to search them. He can choose which ones to keep, and which ones to remove. This facilitates finding parts of a text that have erroneously been marked as locations by a Named Entity Recognition component, for instance. On the other hand, if the component failed to recognize some location names, one can easily correct this using the function for annotating all occurrences of a phrase at once.

Another type of markup error is that two distinct entities have been marked as one, e.g., `<loc>Jamaica and Haiti</loc>`, or vice versa, e.g., `<loc>Trinidad</loc> and <loc>Tobago</loc>`. For correcting this type of error and similar ones at the structure level, e.g., paragraphs, the document editor includes functions for both splitting an XML element at a position between its tags and for merging XML elements of the same name. Within this step, attributes are copied or coalesced, respectively.

3.3 Further Functionality

The GoldenGATE editor natively provides basic NLP functionality like gazetteer **Lists** and **Regular Expression** patterns. Both can be applied for annotating a text document automatically. This is to overcome the need for integrating heavyweight external components for lightweight markup tasks.

All facilities for automated markup can be configured to be one-click accessible in the editor. This saves time when accessing the functions most important for the current task. Besides the ones named here, GoldenGATE pro-vides various further features, which we cannot describe here due to space limitations. New features integrate easily through a **resource manager** interface.

4 Controlled Experiment

To find out empirically whether GoldenGATE supports document mark-up tasks better than existing XML editors, we conducted a controlled experiment. In the experiment, participants were asked to annotate documents according to some XML

schema (see 4.5). The independent variable ("experimental condition") was the editor in use, either GoldenGATE configured with certain custom functions (see 4.5), or XMLSpy. The measured, dependent variable was the completion time for the mark-up task. All other variables which might affect the mark-up performance had to be controlled by experimental techniques.

4.1 Experimental Design

To achieve sufficient statistical power (see 4.6), we needed about 10 data points for *each* editor. We expected to attract no more than 15 volunteers for the experiment; experience shows, though, that not all volunteers actually show up. Hence, it was necessary to choose an experimental design in which every participant would contribute *two* data points, one for each editor.

In our experiment, every participant worked on two different tasks, using XMLSpy for one task and GoldenGATE for the other. We made sure that the tasks were equivalent (see 4.5). When exposing each participant to both experimental conditions (i.e., usage of both editors), there is a general risk that an observed effect is not caused by the variation of the independent variable alone, but also by the order in which the conditions were applied (sequencing effect). Two important sequencing effects might affect our experiment: An increased familiarity with the mark-up task, the structure of the documents, and the experimental environment after completing the first task might have a positive impact on the performance in the second task (learning effect). Being asked to use the "old" XMLSpy editor in the second task after using the "more comfortable" GoldenGATE editor in the first task might have a negative impact on motivation and performance (motivation effect).

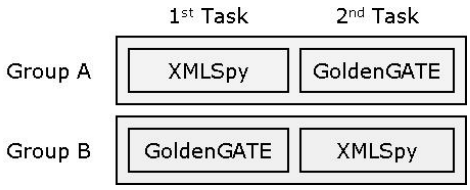

Fig. 2. Counterbalanced experimental design.

To make sure that conclusions about performance advantages of the GoldenGATE editor are valid, selecting a proper design which controls for sequencing effects is mandatory. We applied a *counterbalanced* design [18]: half of the participants used XMLSpy for the first task and GoldenGATE for the second one (Group A); the other participants used the editors in the opposite order (Group B), see Figure 2. To even out differences in individual abilities of participants we *randomized* the assignment of participants to groups.

4.2 Pilot Study

A pilot study serves the purpose of validating the experimental material and environment. A pilot study also helps to estimate the size of the effect to be observed in the experiment, a number which is needed to determine the number of data points required for the experiment (see 4.6). In February, we conducted a pilot study with about half a dozen student volunteers as participants. We used excerpts from biosystematics documents (3, 4) in the tasks for the pilot study. A half-day tutorial was offered the day before the pilot study. It covered the features of XMLSpy and GoldenGATE, but also the structure of biosystematics documents. The emphasis in the tutorial was on hands-on work with the editors.

In the experimental tasks, some participants used XMLSpy to mark up their document, others used GoldenGATE. The pilot study revealed several problems with setup and material. Despite the training, the participants showed a lack of proficiency using the more advanced features of Golden-GATE. They did not have sufficient domain knowledge regarding the structure and contents of the biosystematics articles; hence, they needed considerable time to recognize the relevant parts of the documents. Finally, the experimental tasks were too long, and participants became tired before the tasks were finished. As a consequence, we adjusted the tutorial contents and the material for the main experiment.

4.3 Tutorial

In March, we offered an extended tutorial on one day and carried out the experiment on the next day. Given the insights from the pilot study, we included more and longer practical exercises covering the features of the GoldenGATE editor in the tutorial. We still covered XMLSpy to make sure that the participants had the same degree of familiarity with both editors.

For the tutorial and experimental tasks, we let go of the biosystematics documents and used documents from generic domains which are immediately understood by everyone, such as sports news and recipes for Italian dishes. We held a "competition" at the end of the tutorial where the participants had to mark up a document as quickly as possible using GoldenGATE. The rationale was to see how individuals use the tools when working under pressure. The competition showed that the participants were sufficiently familiar with GoldenGATE.

4.4 Participants

12 graduate students in computer science volunteered for the tutorial and experiment. We had 2 no-shows who attended the tutorial, but did not show up for the experiment, and 1 dropout who gave up after having worked on the first task for more than 2.5 hours. Therefore, we had a total of 9 participants in the experiment. The majority of these students were in their 7th semester; the others were more senior, up to their 13th

semester. All of them had taken a graduate level database class this semester, which also covered XML.

At the beginning of the tutorial, we handed out a pre-test questionnaire that asked for the students' knowledge of XML, their practical experience with editing XML documents using an XML editor, and their practical experience with correcting errors in digitized documents by hand. Except for two students who had used XMLSpy on and off in the past, the pre-test did not reveal any capabilities of the participants relevant for the experiment.

4.5 Tasks

For the experimental tasks, we used sets of recipes as documents, mainly pasta dishes. We made sure that the two documents had about the same length (12 pages), number of recipes (20), and difficulty. The participants easily understood the structure and contents of the recipes. This was important as we wanted to measure the speed advantages resulting from the features of GoldenGATE and not the time needed to understand the problem domain or document content.

The descriptions of the two experimental tasks were identical, except for the name of the document and editor to use. The participants were asked to add suitable tags to structure the document in recipes, preparation sections, and preparation steps. They had to mark up recipe titles, ingredient lists, individual ingredients, and cooking tools. In addition, the participants had to correct errors which are typical left-overs from a previous OCR phase, including extra page titles, incorrect line and page breaks, and misspelled words. This last requirement is particularly tedious when using XMLSpy, hence it was relaxed during the experiment for the XMLSpy users.

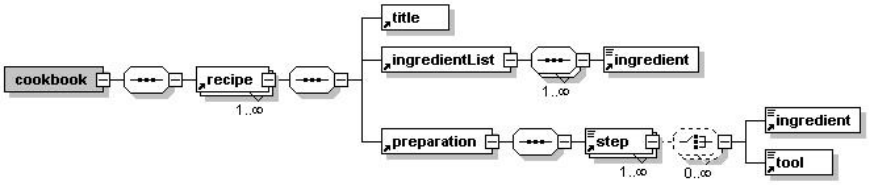

Fig. 3. XML schema for the experiment.

XMLSpy users were given a schema (Figure 3) to help them with the markup. The only NLP functionality that was part of the GoldenGATE configuration used in the experiment was an ingredient tagger and a function for normalizing paragraphs. Thus, if GoldenGATE is superior in this current setup already, we can expect it to be better in ‘real’ settings with more NLP functionality as well. Other GoldenGATE features that we expected to be particularly useful for the experimental task are the list of most recently used annotations and the function for global annotations.

4.6 Sample Size

When planning the experiment, we performed a *power analysis* [17] to estimate the number of data points required to achieve statistically meaningful results. We first chose a significance level of 5 per cent and a desired power [17] of 80 per cent. Then we estimated the effect size for a t-test by considering the expected overlap of the completion time distributions for XMLSpy and GoldenGATE. Based on the data from the pilot study, we expected a large performance advantage of GoldenGATE; hence, we assumed a small overlap of the time distributions of just 10 per cent. This expected overlap maps to an effect size [17] of 1.3 for the t-test. Given a significance level of 5 per cent and an effect size of 1.3, a power of 80 per cent maps to a requirement of 8.3 data points in each experimental condition (i.e., for each editor) for a one-sided t-test [17]. Similarly, the desired power maps to a requirement of 9.6 data points for each editor for a one-sided Wilcoxon test. As a result, we needed to collect between 8 and 10 data points for each editor in the experiment.

4.7 Document Quality

When measuring task completion times as the dependent variable, it is important to make sure that the output of the experimental tasks has a uniform (and minimum) quality; otherwise, short completion times might simply correlate with low or even unacceptable output quality. We defined thresholds for the correctness of the final document: We required 100% correct structural mark-up (recipe, title, ingredientList, preparation, step) and 85% correctness of the semantic markup (ingredients, tools).

We installed a test server in the local intranet which compares the quality of uploaded documents against "gold documents." As testing had very low overhead, we encouraged the participants to upload their intermediate documents to the test server for acceptance testing at will during the experiment. Participants were finished with their task only after having passed the full acceptance test. This required meeting all thresholds and implied having worked on all parts of the task successfully.

4.8 Results

We have 9 valid data points for each editor. For all but one participant, the task completion time when using GoldenGATE was *significantly smaller* than the task completion time when using XMLSpy (Figure 4). The mean of the XMLSpy task completion times is 107 minutes; the mean for GoldenGATE is 77 minutes. The average relative speed-up was 25 per cent with GoldenGATE.

The performance advantage of GoldenGATE over XMLSpy is statistically significant at the 2 per cent level, with a p-value < 0.013 for the paired t-test and a p-value < 0.004 for the paired Wilcoxon test. Our experiment provides strong empirical evidence that the GoldenGATE editor supports document mark-up tasks better than a standard XML editor, such as XMLSpy.

On average, the time needed to complete the first task (102 minutes) was longer than for the second task (82 minutes). Obviously, there was a *learning effect* between the two tasks. This effect does *not* invalidate our findings, though, because the learning effect applied uniformly to both editors: For XMLSpy, the mean task completion time decreased from 117 (Group A) to 97 (Group B) minutes between the two tasks; for GoldenGATE, it decreased from 84 (Group B) to 72 (Group A) minutes. (These differences were visible, but not statistically significant, with p-values larger than 0.11 and 0.19, respectively). Note that this analysis would not have been possible without a counterbalanced design.

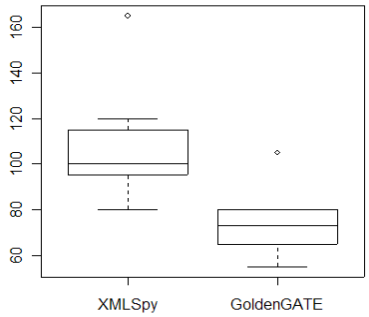

Fig. 4. Boxplot of the completion time distributions.

When comparing XMLSpy with GoldenGATE for the first task only, the performance difference is significant at the 6 per cent level; similarly, when comparing the editors for the second task only, the difference is significant at the 2 per cent level. Hence, the performance advantage of GoldenGATE over XMLSpy is independent of the order in which the editors were used.

In one exceptional case, the participant was slightly faster using XMLSpy than using GoldenGATE. As a possible explanation, this participant stated in the pre-test questionnaire that he had used XMLSpy on and off prior to the tutorial. In addition, he used GoldenGATE in the first task; hence, the learning effect between the two tasks is likely to have aggravated the observed effect.

From the means and variances of the completion time distributions, we compute [17] an observed effect size of 1.43. Given a significance level of 5 per cent, the experiment has a post-hoc power of 89 per cent for the t-test and of 77 per cent for the Wilcoxon test. Thus, even with fewer data points than originally planned our experiment had a satisfactory power.

5 Conclusions

Integrating assisted manual XML editing and automated markup via NLP applications is promising to efficiently create semantically rich digital libraries. We have implemented this integration in the GoldenGATE editor. In this paper, we have reported on a thorough empirical assessment of this approach. Our study provides

strong evidence that a user can perform markup tasks much faster when he can conveniently use NLP functions and does not have to pay attention to the XML syntax, as he would have to in a conventional XML editor. GoldenGATE shows a strong performance because of its easy-to-integrate task-specific NLP functions and its sophisticated assistance for manual XML editing. Users do not need to worry about the wellformedness of the markup because the editor enforces it with each editing step.

In our controlled experiment, the performance advantage of GoldenGATE configured with moderate NLP-functionality was 25% over XMLSpy. When fully customized for a class of documents, NLP can automate the annotation task to a large degree. Thus, we expect a much larger performance advantage of GoldenGATE in domains where the documents have a richer semantic structure, for instance, in biosystematics legacy literature.

The current version of GoldenGATE is available for download at http://idaho.ipd.uka.de/GoldenGATE/.

References

1. A. Mikheev, M. Moens, C. Grover, *Named Entity Recognition without Gazetteers*, in Proceedings of EACL, Bergen, Norway, 1999
2. D. Miller, S. Boisen, R. Schwartz, R. Stone, R. Weischedel, *Named Entity Extraction from Noisy Input: Speech and OCR*, in Proceedings of 6th Conference on Applied NLP, Seattle, WA, USA, 2000
3. G. Sautter, D. Agosti, K. Böhm, *Semi-automated XML Markup of Biosystematics Legacy Literature with the GoldenGATE Editor*, in Proceedings of PSB, Weilea, HI, USA, 2007
4. G. Sautter, D. Agosti, K. Böhm, *A Combining Approach to Find All Taxon Names (FAT) in Legacy Biosystematics Literature*, Biodiversity Informatics Journal, 3-2006
5. W. Tichy, *Hints for Reviewing Empirical Work in Software Engineering*, Journal of Empirical Softw. Eng. 5 (2000) 309-312
6. M. Müller, F. Padberg, *An Empirical Study about the Feelgood Factor in Pair Programming*, Int. Symp. on Softw. Metr. 10 (2004) 151–158
7. IDM Computer Solutions Inc., www.ultraedit.com
8. <oxygen/>, www.oxygenxml.com
9. Altova GmbH, www.altova.com
10. The OpenNLP project, www.opennlp.org
11. LingPipe, www.alias-i.com/lingpipe
12. L. Rabiner, B. Juang *An Introduction to Hidden Markov Models*, in IEEE ASSP Magazine, Jan 1986, Volume 3, Issue 1, pp 4-16
13. GATE, General Architecture for Text Engineering, gate.ac.uk
14. Apache Lucene, lucene.apache.org/java/docs
15. WordFreak, http://wordfreak.sourceforge.net/
16. Knowtator, http://bionlp.sourceforge.net/Knowtator/index.shtml
17. J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed., Erlbaum, Hillsdale, NJ, USA, 1988, ISBN 0-8058-0283-5
18. L. Christensen, *Experimental Methodology*, 10th ed., Pearson, Boston, MA, USA 2007, ISBN 0-205-48473-5