An Open Architecture for End-to-End Document Analysis Benchmarking

Bart Lamiroy
LORIA
Nancy Université – INPL
Nancy, France
Bart.Lamiroy@loria.fr

Daniel Lopresti
Computer Science and Engineering
Lehigh University
Bethlehem, PA 18015, USA
lopresti@cse.lehigh.edu

***Abstract*—In this paper, we present a fully operational, scalable and open architecture allowing end-to-end document analysis benchmarking without needing to develop the whole pipeline. By decomposing the analysis process into coarse-grained tasks, and by building upon community provided state-of-the art algorithms, our architecture allows any combination of elementary document analysis algorithms, regardless their running system environment, programming language or data structures. Its flexible structure makes it straightforward to plug in new algorithms, compare them to other algorithms, and observe the effects on end-to-end tasks without need to install, compile or otherwise interact with any other software than one's own.**

***Keywords*-benchmark; web services; document analysis; performance evaluation;**

I. Introduction

The issue of benchmarking, and more generally speaking, comparing results in experimental science, is a recurring and fundamental discussion (*cf.* [1], [2], [3] for an incomplete list in the document analysis domain). To compare reported results with others in an irrefutably objective framework is essential in all scientific domains. Better still is to be able to reproduce reported experiments and compare the obtained results with those published. Moreover, it is not sufficient to only have access to the outcomes of experiments; in many cases sound scientific debate arises over conclusions and deductions taken from a set of experiments when contesting parties question the validity of the claims on grounds of incomplete parameter scopes or experimental conditions. This is not only part of normal scientific experimentation, it is the core essence of scientific activity.

In the domains of machine perception, and document analysis in particular, many initiatives and efforts have emerged over the years, to provide the research community with tools that would allow sound contradictory debate and objective comparison between published methods.

They can be placed into several categories:

- **annotated datasets**, allowing experiments on the same data, and thus partially share parameters [4], [5], [6];
- **execution frameworks and software**, which allow experiments to run on shared code bases, and therefore have a greater level of interaction and re-usability [7], [8], [9] (re-usability can be seen as some form of reproducibility);
- **data markup languages**, while sometimes much related to the previous point, they allow for more transparent data sharing and comparison of experimental results [10];
- **contests**, held at international conferences or hosted on a more continuous basis by scientific associations.

Despite these efforts[1], and despite the peer review processing, there still remains room for progress in the area of contradictory debate and objective comparison between published methods [11]. There are several factors that contribute to this, especially when it comes to reproducibility and comparison of results.

Shared datasets, for instance, while having significantly contributed to improving performance metrics [2], fail in other cases to contribute to a productive debate. After being used for some time, research teams inevitably, and often unconsciously, learn to adapt to their specificities and implicit context assumptions; their "ground-truths" (or pre-recorded interpretations) suppose a one-size-fits-all approach across all evaluations.

Execution frameworks and open-source software have the advantage of allowing users to share the same code base and experimental run-time conditions or input-output conventions. While this has a positive impact on reproducibility, their scope is sometimes limited, since users are coerced into software environments, programming languages or other operational constraints that may not be optimal in their specific context. Furthermore, these tools require significant efforts to maintain (open-source community development efforts require a critical mass to be self-sustaining) and may, over time, suffer from technological obsolescence, making them harder to maintain on newer platforms.

Contests are great tools for providing instant snapshots of the state of the art in the evaluation context that was specifically defined for the contest. Rarely do they offer the opportunity to "replay" the contest with other parameters,

[1]Owing to space constraints, it is only possible for us to cite a small fraction of the many existing efforts to contribute shared resources for the use of the research community.

or even to compare contestants of one year with those of previous editions ... or to imagine *what if* scenarios where the evaluation or execution context are shifted into a different setting. The result is that their outcomes are static, and hence cannot support a dynamic debate.

Creating the ideal conditions for productive, reproducible experimental science is not an obvious task, and is certainly not one that can be achieved by anything short of a whole community. It cannot be dictated, but rather comes from a common adoption of what are considered best practices. The previously enumerated examples are steps in that direction. We believe that they can be re-formulated into a new paradigm, providing an even higher level of scientific trustworthiness in and reproducibility of reported experimental results [11]. Because of space constraints, we are only addressing those parts of the technical implications of this paradigm that are concerned with execution and reproduction of algorithms. Other aspects, focusing on data or interpretation storage and retrieval, result certification or provenance recording have been described before [12], [13].

This paper will focus of the question of reproducibility from an operational point of view, and especially in the light of document analysis problems. It should be noted that the basic ideas we are presenting are more general and not confined to the field of document analysis.

We begin by analyzing in Section II the technical features required for a generic framework for supporting reproducible experiments. In Section III, we lay out the technologies and architectures that address these requirements and present the tools we have developed to implement our system.

II. Requirements for Reproducible Scientific Experiments

The question on how to produce and report scientifically sound and valid experiments is not new [14], it is an ongoing debate in all disciplines, e.g. [15], [16].

The basics are:

1) reporting of clearly set goals and defined interpretation framework,
2) full access to all experimental data,
3) reporting of the experimental apparatus, setup and protocol, in such a way that it becomes fully reproducible,
4) all parameters defining the data (if applicable) and those related to the experimental process.

While these seem obvious, and are apparently met by the initiatives presented previously, the currently available resources fail to produce the effect of fully reproducible open experiment reporting. The reasons for this include:

Full disclosure and complete reporting: is often difficult to achieve for methods and algorithms, notably because of space constraints on publications. Even if the reporting is completely transparent, it still may be hard to reproduce complex algorithms and obtain identical behavior due to implementation choices, bugs, *etc.* Making source code available, or using shared development or execution platforms [7], [8], [9] is helpful, but practice shows that this only rarely yields comparative studies. The reason for that is that the platforms are very much technology dependent (choice of specific programming languages, operating systems, data structures or other constraining paradigms) that often require a time investment that discourages others from using it. They also may suffer from progressive obsolescence when not actively maintained. Releasing source code can also be problematic when private funding, IP or patents come into play.

Full access to all experimental data: should not really be an issue, given today's ubiquitous access to storage and bandwidth (although this does become an issue when the amount of data becomes too large [17]), but there are more subtle difficulties. The way benchmark datasets are currently conceived and made available is rather "monolithic" in the sense that they have usually been created for a specific experimental context, and that their intrinsic parameters (*e.g.* type of images, resolution, content, frequency ...) and associated interpretations are those that suit this context. In order to adapt to these implicit constraints, re-use of existing datasets often comes with recomposition, selection and filtering of the original data, blurring the exact boundaries of the effectively used data.

Exact description of all parameters: is difficult to provide, especially for data, since they often reflect a mix of arbitrary design decisions (my method does only take `.tif` images) and more subjective ones ("reasonably" good scan quality so that the OCR doesn't fail). Because of different experimental contexts, it is rarely the case that exhaustive experiments are reported over complete datasets, without the selection and filtering mentioned beforehand. This also sometimes holds for contests where training data is not always formally characterized, for instance.

The result is that most of the time, results are reported and published in good faith and that peer reviewers need to find the subtle compromise between investigating the veracity of the alleged claims and their experience-backed feeling of their plausibility. This also holds for readers who would want to build upon published results, either to compare their own methods to the published ones using the same experimental context, or to change the experimental context and setup to test the published approaches with other data.

In order to achieve the goals described in this section, any published result (data, algorithm and experimental setup parameters) should be transposable in any context and should allow any user, using any kind of environment to immediately reproduce the experiments in the exact same conditions as reported. Elsewhere we have addressed some of the issues in referencing datasets and interpretations [12], [13]. The following sections will focus on how to reproduce experimental setups and run algorithms in the exact same conditions as published.

III. Global Architecture

The solution presented here has been tested and integrated in the DAE platform at http://dae.cse.lehigh.edu/DAE. Its global architecture is represented in Fig. 1. It is noteworthy to mention that the platform itself is merely a validation of the concepts presented in this paper, and that the latter can be implemented outside of the platform without loss of the general goals announced in the previous section.

A. Document Analysis Algorithms as Web-Services

Rather than to require user to adopt a specific programming paradigm, dedicated libraries or integrated systems, we propose to build our solution on an open standard and to publish algorithms as web services [18]. This solution offers the following advantages:

Platform Independent Open Standard: web services are built upon a series of standards that are well documented and widespread. They can be accessed and used through a great variety programming languages and execution environments. While platform independent, they are completely interoperable, and services running in one environment/language can exchange data and collaborate seamlessly with services running elsewhere.

This means that this approach is not incompabible with conventions users may have to run their experiments in their preferred environment, while, at the same time, giving them access to software and experiments of others.

Formalized I/O: WSDL is part of the previously mentioned standard. It permits formalizing the input and output of services in a detailed way, and to provide service brokers to publish the list and location of available services. These are important features from a functional point of view, since they improve the level of re-usability and interaction with provided services. It makes it, for instance, possible to have more "semantic" information on the required inputs and provided outputs of an algorithm implementation and use zero-programming environments. These environments provide the opportunity to combine and connect available services into complex workflows, and "program" end-to-end pipelines (in our case document analysis pipelines) made out of multiple external individual programs (*cf.* Taverna, discussed in Section III-C).

No Code Re-Engineering: because of the open standards and the formalized I/O, there is no real need to re-engineer existing code to make it available as a web service nor is there any need to do so to adapt and re-use an existing service in one's personal environment.

Although the above statement is true, existing software doesn not turn into a web service by itself. There are multiple degrees of integration with the web service paradigm that can be envisaged and engineered. At the lowest level, providing the existing code does not require interactive user input and runs as a standalone command-line program, it is possible to use off-the-shelf wrappers that call the code as an external command (*e.g.* those provided in Section III-B).

Smooth Learning Curve (if any): given the facts enumerated above, the whole framework presented in this paper is totally non-disruptive with users' usual way of working and experimenting. They continue to develop software exactly the way it was done before, in the environment they are familiar with. Since the solution is built upon an open and widely adopted standard, all available tools and resources related to web services can be leveraged and integrated when needed. The learning curve for adoption is therefore smooth and directly related to the integration needs of the user. On the other hand, the user has immediate access to all algorithms published by others without additional cost.

B. Creating or Hosting Web Services

From a purely technical point of view, the only requirement is that a web service be associated with a network address, hosting an interface that complies with WSDL. For instance, the web services hosted by the DAE platform are available at http://dae.cse.lehigh.edu/DAE/services/soap.

It should be noted that there is no need for the web services to be universally accessible. The WSDL server does not need to be accessible from other parts of the Internet and may perfectly be running on one's personal machine, on a local non-routable network, if there are no other intended users of the service.

As already stated before, web service servers can be made as complex as needed for specific task and integration requirements, but that is beyond the scope of this paper. However, in order to achieve the claims of contributing to broad reproducible experimental research, it is necessary that there is a clearly perceivable benefit from using this framework. Complex technical constraints are a hurdle to attain this. We therefore provide interested users with a straightforward way to set up, test and use their personal applications as web services. Full instructions and an installation kit are provided at http://sourceforge.net/projects/daeplatform/. The kit mainly consists of 3 PHP files and require no more than 15 lines of code to be modified to operate, mainly consisting in naming the right I/O variables and constructing the command string corresponding to the application to be launched.

To be publicly registered and referenced by the DAE platform, the process is only slightly more complicated. A web service must comply with a number of supplementary conditions, the most important being that its input and output parameters be defined in accordance with the DAE data model. However, there is no specific obligation for public web services to be registered in this way to still be available to the broader community. Since the protocols are open and standardized, other service providing repositories can be created and interact with the ones presented here.

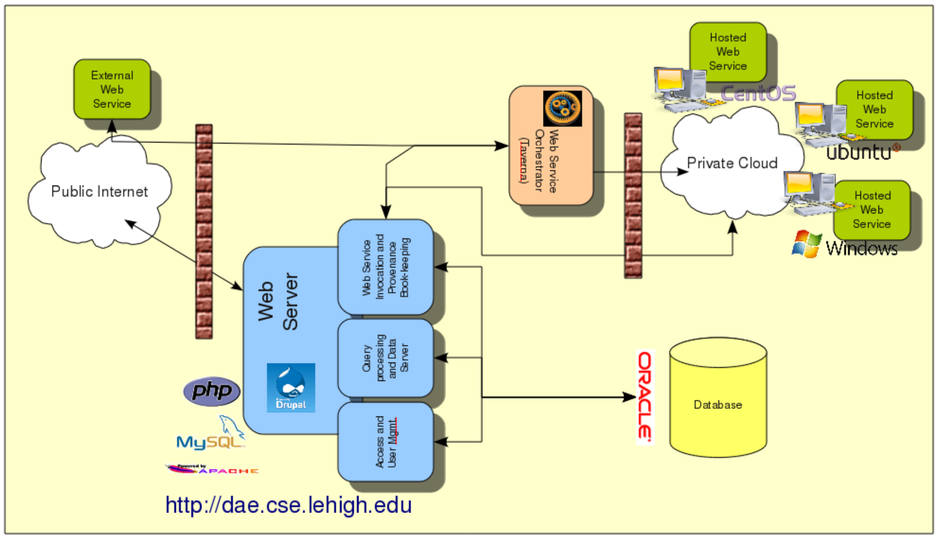
External Web Service
Public Internet
Web Server
Web Service Invocation and Provenance Book-keeping
Query processing and Data Server
Access and User Mgmt.
Web Service Orchestrator (Taverna)
Hosted Web Service
Hosted Web Service
Hosted Web Service
Private Cloud
Database
http://dae.cse.lehigh.edu

Figure 1. Overview of the DAE platform architecture

C. *Using and Accessing Web Services*

While it remains perfectly possible to write web service aware programs that would interact through WSDL interfaces, there is no real need to program anything to use the document web services mentioned in the previous section. The only thing that is needed is a web service invocation tool like Taverna [19].

Fig. 2 gives an example of what this looks like. Taverna allows the user to create input and output flows, and redirect them to and from web services, in a graphical click-and-play environment. The workflows can be saved into lightweight XML description files and shared with others. The tool also provides simple and straightforward operators like loops, parallelization, *etc.*

IV. Conclusion

In this paper we have advocated the use of web services to improve overall reproducibility of published document analysis results. In this section we will review the advantages this offers and how it can contribute to more generic and widespread benchmarking and evaluation protocols.

A. *Availability, Access and Archiving*

Providing software as a web service removes three major hurdles found with simply making source code available, without adding disadvantages, since all source code can still be provided (and should be promoted as good practice).

First, difficulties related to cross-platform issues disappear. Moreover, since the development overhead of making one's own code available as a web service is close to zero (and is most certainly far less than distributing and maintaining cross-platform code) there is a clear incentive to contribute programs through web services. Second, cases where intellectual property or commercial constraints have previously hindered sharing, can now be integrated for research and benchmarking, since all resources can remain under the control of those who provide the service. Finally, it becomes technically possible to keep operational and functional archives of legacy programs through virtualization techniques, since older code, running on obsolete operating systems or in discontinued environments can still be made available by having them run in virtual machines. The DAE platform, for instance, provides access to VirtualBox-hosted [20] environments.

There is one catch to the approach presented in this paper, however: the dependency on network and computing resources. Other approaches, notably those consisting in distributing source code only, leave it to those who want to reproduce the results to prepare and execute the experiments, and therefore dispose of the required resources. In our case, it is the provider of the service who also provides execution resources. Partially centralized or distributed community resources may alleviate this.

B. *Reproducibility, Benchmarking and Evaluation*

The great gain that is achieved through this framework is full reproducibility for results, since the complete operational conditions of experiments are available, without bias related to implementation choices or parameter tuning. Complete experimental pipelines can be formalized and distributed with Taverna (or other, equivalent resources) and executed anywhere, without constraints. These pipelines (or workflows) can be re-appropriated by others and enhanced, challenged or transposed to other contexts, and through publication, contribute to sound and productive contradictory discussion of results and methods in general.

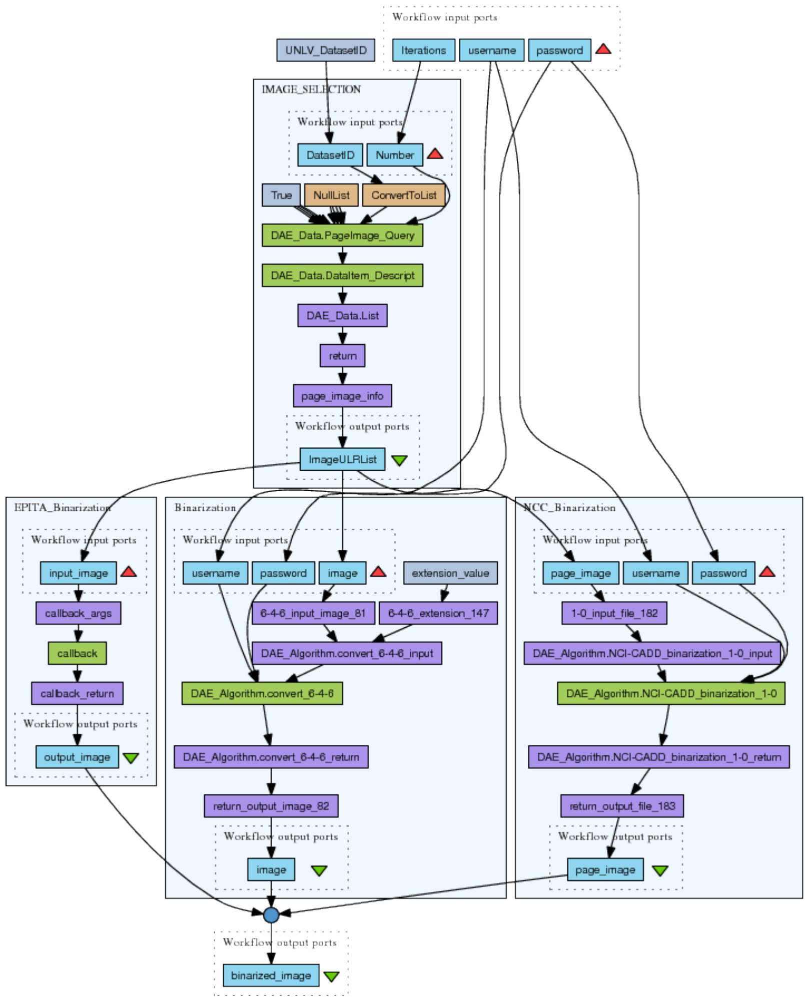

Figure 2. Taverna workflow example, automatically taking random images from a DAE dataset, running different binarization algorithms.

By doing so, they become a cornerstone to the establishment of open, sharable benchmarking procedures in a transparent and reproducible way. Furthermore, it makes it much easier to conceive and distribute complex end-to-end workflows as evaluation procedures for individual algorithms. For example, studying both the influence of individual OCR or layout analysis programs with respect to higher level interpretation tasks, like named entity detection, can be formalized and published as a workflow of generic web service components. It then becomes possible to replace individual algorithm contributions (*e.g.* a new OCR algorithm) without affecting the general experimental setup, and thus maintaining the objectivity of resulting performance measurements, without major re-engineering or software integration efforts.

C. *Future Work*

Currently ongoing work is related to the deployment and effective use of the framework for IAPR ICDAR and GREC contests [21]. This will provide, on the one hand, stress tests on the DAE infrastructure hosting some of the web services. The data generated will give useful information for further use and development. It will also provide user feedback on the usability and the efficiency of the approach. One of the interesting expected by-products of an effective use of the paradigm is that in a first stage, contributers are expected to simply provide a web service executing their software (either self-hosted or hosted on the DAE platform). In a second stage, when trying to interconnect various web services there will be need for specific contributions like format converters. It is expected that these converters, however, can be published as web services as well, thus contributing to a virtuous cycle of increased usability and usefulness and eventually users.

Other further extensions include a thorough evaluation of the impact on performance (throughput and bandwidth as well as overhead caused by the web service orchestrator) compared to direct execution. It should be emphasized that this framework is designed for research flexibility, and is probably not the optimal solution for high performance production environments. Also, there is an open question whether the standard synchronous WSDL approach is actually the best solution, or if in some cases, REpresentational State Transfer (REST) Web services would not be more appropriate.

A complete tutorial comprising examples and source code downloads can be found at http://tinyurl.com/DAE-Web-Service-Tutorial.

Acknowledgements

The authors wish to thank Lehigh students A. Borden and Q. Long for their technical contributions and research on Taverna. Bart Lamiroy was a visiting scientist at Lehigh University in 2010-2011. This work was conducted at the Computer Science and Engineering Department at Lehigh University and was funded through DARPA IPTO via Raytheon BBN Technologies.

References

[1] T. Kanungo, H. S. Baird, and R. M. Haralick, "Special issue on "performance evaluation: Theory, practice, and impact"," *IJDAR*, vol. 4, no. 3, p. 139, 2002.

[2] G. Nagy, "Document systems analysis: Testing, testing, testing," in *DAS 2010, Proceedings of the Ninth IAPR International Workshop on Document Analysis Systems*, D. Doerman, V. Govindaraju, D. Lopresti, and P. Natarajan, Eds., 2010, p. 1.

[3] E. Valveny, P. Dosch, A. C. Winstanley, Y. Zhou, S. Yang, L. Yan, W. Liu, D. Elliman, M. Delalandre, É. Trupin, S. Adam, and J.-M. Ogier, "A general framework for the evaluation of symbol recognition methods," *IJDAR*, vol. 9, no. 1, pp. 59–74, 2007.

[4] I. T. Phillips, S. Chen, and R. M. Haralick, "CD-ROM document database standard," in *Document image analysis*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1995, pp. 198–203.

[5] G. Thoma, "Automating the production of bibliographic records for MEDLINE," National Library of Medicine, Bethesda, MD USA, Tech. Rep., September 2001.

[6] G. Agam, S. Argamon, O. Frieder, D. Grossman, and D. Lewis, *The Complex Document Image Processing (CDIP) test collection*, Illinois Institute of Technology, 2006.

[7] T. M. Breuel, "The ocropus open source ocr system," in *DRR*, ser. SPIE Proceedings, B. A. Yanikoglu and K. Berkner, Eds., vol. 6815. SPIE, 2008, p. 68150.

[8] J. Rendek, G. Masini, P. Dosch, and K. Tombre, "The search for genericity in graphics recognition applications: Design issues of the qgar software system," in *Document Analysis Systems*, ser. Lecture Notes in Computer Science, S. Marinai and A. Dengel, Eds., vol. 3163. Springer, 2004, pp. 366–377.

[9] Stefan Jaeger, Guangyu Zhu, David Doermann, Kevin Chen, and Summit Sampat, "DOCLIB: a Software Library for Document Processing," in *International Conference on Document Recognition and Retrieval XIII*. San Jose, CA, 2006, pp. 1–9.

[10] David Doermann, Elena Zotkina, and Huiping Li, "GEDI – A Groundtruthing Environment for Document Images," *Ninth IAPR International Workshop on Document Analysis Systems (DAS 2010)*, 2010.

[11] D. Lopresti and B. Lamiroy, "Document Analysis Research in the Year 2021," in *Twenty-fourth International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2011)*, ser. LNCS, Syracuse University. Syracuse, NY, United States: Springer, Jul. 2011.

[12] B. Lamiroy and D. Lopresti, "A platform for storing, visualizing, and interpreting collections of noisy documents," in *Fourth Workshop on Analytics for Noisy Unstructured Text Data - AND'10*, ser. ACM International Conference Proceeding Series, IAPR. Toronto Canada: ACM, October 2010.

[13] B. Lamiroy, D. Lopresti, H. Korth, and H. Jeff, "How carefully designed open resource sharing can help and expand document analysis research," in *Document Recognition and Retrieval XVIII*, ser. SPIE Proceedings, G. Agam and C. Viard-Gaudin, Eds., vol. 7874. San Francisco, CA USA: SPIE, January 2011.

[14] K. R. Popper, *The Logic of Scientific Discovery*, reprint ed. Routledge, October 1992, original edition, 1934 "Logik der Forschung".

[15] B. M. E. Moret and H. D. Shapiro, "Algorithms and experiments: The new (and old) methodology," *Journal of Universal Computer Science*, vol. 7, no. 5, pp. 434–446, 2001.

[16] M. Schwab, M. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science and Engg.*, vol. 2, pp. 61–67, November 2000.

[17] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and TRECVid," in *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*. New York, NY, USA: ACM Press, 2006, pp. 321–330.

[18] D. Booth and C. K. Liu, "Web services description language (WSDL) version 2.0 part 0: Primer," W3C, Candidate Recommendation, Mar. 2006.

[19] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1067–1100, August 2006.

[20] J. Watson, "Virtualbox: bits and bytes masquerading as machines," *Linux Journal*, vol. 2008, February 2008.

[21] B. Lamiroy, D. Lopresti, and T. Sun, "Document Analysis Algorithm Contributions in End-to-End Applications," in *11th International Conference on Document Analysis and Recognition - ICDAR 2011*. Beijing, China: International Association for Pattern Recognition, Sep. 2011.