A heuristic measure for detecting influence of lossy JP2 compression on Optical Character Recognition in the absence of ground truth

Sven Schlarb; Austrian National Library; Vienna, Austria; Clemens Neudecker, National Library of the Netherlands; The Hague, The Netherlands

Abstract

Cultural heritage institutions such as libraries, museums and archives have been carrying out large scale digitisation projects during the last decade, and the question how to store digital master images in a cost effective way made the JPEG 2000 standard (ISO/IEC 15444-1), especially the JP2 image file format (JPEG 2000 Part 1), popular in the library, museums, and archives community. Especially the lossy JP2 encoding of page image masters provides a good balance between file size reduction and preservation of the visible properties of a master image. Lossy JP2 encoding of digital images means that it is not possible to restore the original file at the bit level, even if there are no distinguishable differences to the human eye. But the absence of visual changes does not always imply that there is no influence on the computational processing of the images. In this context we present a heuristic measure that helps to detect undesired influence of lossy JP2 compression on the OCR result, and in the absence of ground truth.

Introduction

For large scale digitisation projects producing millions of book or newspaper images, lossy JPEG 2000 compression is a compelling option because it can lower the storage requirements significantly when compared to lossless JPEG 2000 compression.

According to an evaluation of archival image format alternatives carried out by the National Library of the Netherlands in 2008, the storage reduction to be expected can be approximately 50% for lossless and between 91% and 98% for lossy compression [1].

While some institutions, like the National Library of Norway, reported in 2007 that they opted for a lossless compression profile [2], others, like the Wellcome Digital Library commissioned a report which explicitly recommended the use of lossy compression for digital image masters [3].

Generally, format conversion implies a reorganisation of image data where different kinds of errors can occur due to hardware or software failures. For that reason, quality assurance is often part of image migration workflows in memory institutions. Particularly when lossy compression is applied to image master files, quality assurance becomes an essential activity since the lossy compression can lead to artefacts in the image that are not directly visible, but have an influence on the machine processing of the image.

One of the processes that are directly related to the quality of digital master images representing text, like scans of book or newspaper pages, is the transformation of a digital image into searchable electronic text by means optical character recognition (OCR) technology, and the question therefore is how the lossy JP2 encoding influences the text recognition accuracy.

The idea of applying a quality assurance process that compares original and target image of the format conversion process has been discussed in detail, especially with regard to the JPEG 2000 standard [4]. In addition, using OCR as a method in quality assurance is common practice and studying the influence of compression on the OCR success by comparing the OCR result of the original with the OCR result of the converted image has been mentioned as a measure to avoid undesired results, for example in relation to the JBIG2 compression:

> "The key for JBIG2 use in compressing documents that need to be archived or that have strict retention requirements is that the JBIG2 encoding be reliable and not lose any informational content. One way to measure informational loss effectively in an automated environment is through an OCR (optical character recognition) program which can verify that recognition rates are as high after JBIG2 compression as before [5]."

The process of evaluating OCR accuracy reliably requires Ground Truth data [6,7] which is defined as the close to 100% correct transcription of the text visible on a document page. The quality of the OCR result is then typically measured in accuracy levels as the percentage of correctly recognised characters and/or words among the total number of characters and/or words recognised. Ground Truth creation is usually done by service providers that manually re-key the text, which is an expensive service and can therefore only be done for small collections.

Assuming the situation that an archive decides to migrate TIFF image files to lossy JP2 compressed images and has no Ground Truth data available, the question is: Can a heuristic measure ibased on the deviation of the OCR result from the original image compared to the OCR result of the compressed image be used to detect undesired effects caused by the lossy JP2 compression? It is important to note that the deviation from the OCR result compared to the OCR result of the original TIFF image does not imply that the result is worse in terms of OCR accuracy. On the contrary, the comparison may even show better OCR performance on the compressed images, as it turned out in a study by Chapman et al [8].

The question is if this measure, which cannot be considered as an indicator of OCR accuracy, can still be used as a heuristic for detecting bad quality in digital images representing text? It is also important to bear in mind the high processing time of OCR that constrains using it as a bad quality detection mechanism, especially in mass processing of huge digital image collections.

Before we will address these questions in the context of the experiment presented here, we will first give a brief review on the emergence of the concept of collaborative experimentation, and describe the technical environment.

Collaborative Experimentation in Digitisation and Preservation

The concept and some technical cornerstones of collaborative experimentation in digitisation and preservation have been developed in the course of various EU funded projects, making use of several open source software resources.

The PLANETS Testbed Experiment

Conducting experiments in order to compare JPEG2000 migration alternatives, like Kakadu and Jasper, has already been presented at the Archiving 2010 conference in an experiment that used the PLANETS Testbed [9]. The focus of this experiment was to compare the encoding and decoding performance as well as to assess the reliability of the migration process.

The IMPACT Interoperability Framework

In the EU-funded research project: IMPACT (IMProving ACcess to Text, www.impact-project.eu), workflow development was introduced as a community-driven activity using an experimental workflow development and execution platform that was coupled to the IMPACT Interoperability Framework [10]. The main advantages of performing experiments as a community activity has been the ability to share, comment, rate, organise, maintain and manage workflows jointly: the community participants created their own workflows by combining ready-to-use software components requiring only basic technical knowledge, and alongside improved their understanding of the challenges and technical solutions. While the PLANETS Testbed, mentioned in the previous section, had a fixed six step experiment process in order to make experiment results comparable, in IMPACT one of the objectives was to provide maximum modularity and flexibility in designing the experiments, so that they can be tailored to very specific challenges, and conducted in a fully transparent, reproducible way.

The SCAPE Testbeds

In the SCAPE Project (SCAlable Preservation Environments, www.scape-project.eu), the results of the PLANETS and IMPACT projects are important foundations for furthering the concept of collaborative experimentation. A workflow represents in SCAPE the formalisation of a use case which is implemented by using a set of already existing or prototype development solutions. The SCAPE Project will not only develop a platform that enables the execution of these experimental workflows on a small test dataset, but also scalable solutions that can be used to process the huge amounts of data that cultural heritage and memory organisations are increasingly facing.

Technical Environment

Taverna

Taverna [11] is a scientific workflow language, and computational model designed to create software-based workflows of complex and data-driven processes. Taverna is known since 2004 as a tool that primarily bio-informaticians use to formally describe and enact the data pipelines of a computational experiment. In the meantime, its application domain has been extended and now also includes the digitisation and preservation domains. The Taverna Workbench can be used for interactively designing a workflow, and for starting and monitoring the execution of it. The latter is done by the Taverna engine, which directs data through the processing pipeline and has the ability to capture provenance traces.

myExperiment

The Taverna workflow design and execution workbench has been integrated with the community driven myExperiment platform (www.myexperiment.org), a virtual research environment for discovering and sharing scientific workflows [12]. Taverna and myExperiment are fully integrated, a user can launch Taverna workflows from within the myExperiment web site, and conversely, the myExperiment repository can be searched from within the Taverna workbench.

Hadoop

Hadoop [13] provides an implementation for MapReduce [14] which is a programming model for distributed processing of large datasets. In the above mentioned project SCAPE, the Hadoop framework is being used to enable the execution of workflows on a cluster of machines. An important question here is how components that make use of the MapReduce programming model can interact with the Taverna execution engine for workflows that run on very large data sets.

The experiment

The Taverna workbench has been used to develop an executable experiment that applies a processing pipeline to a sample set of image files. The main inputs of this workflow are the list of uncompressed input files (workflow input port "inFileAbsPaths" in Figure 1) and the list of JP2 compression parameter values (workflow input port "rates" in Figure 1: a simple string which is split into a list of compression parameter values). The two lists are combined using Taverna's cross product list handling type which means that each input image file from the one list is combined with each compression parameter value from the other list in order to compute the list of encoding/decoding tasks that will be executed. For example, a list of three image files $F = \{f_1, f_2, f_3\}$ and a list of two compression parameter values $C = \{c_1, c_2\}$ leads to a list of six JP2 encoding/decoding processes $F \times C = \{ \{f_1c_1, f_2c_1, f_3c_1\}, \{f_1c_2, f_2c_2, f_3c_2\} \}$ to be executed.

The remaining input ports are single input ports which configure the command for applying the Tesseract OCR [15] (workflow input port "tesscmd" in Figure 1) and indicate the Tesseract language module to be used (workflow input port "tess_langmod" in Figure 1).

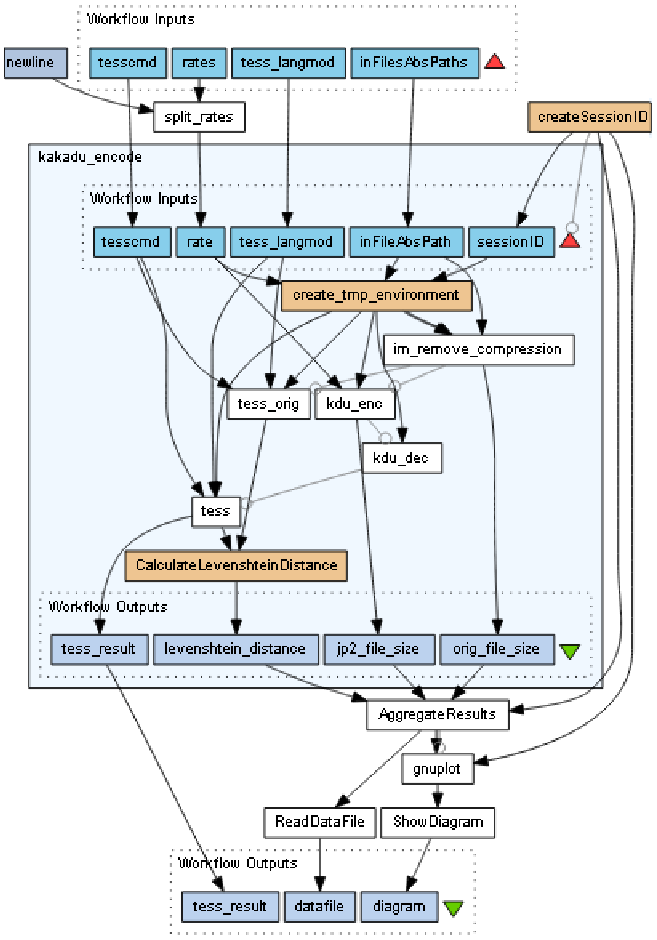

***Figure 1**. Diagram of the Taverna workflow used in the experiment; the workflow is available on myExperiment at http://www.myexperiment.org/workflows/2724*

Each JP2 encoding/decoding process f_ic_j encodes the input image using JP2 lossy compression and decodes the image back to the original image to which then OCR is applied. This OCR result is compared with the OCR result of the original image using the Apache commons-lang implementation of the Levenshtein Distance algorithm [16] which is a string metric indicating the number of edit operations needed to convert one text string into another.

For integrating external software components, Taverna's "tool service" is used which allows the inclusion of external command line applications within a workflow.

Regarding external dependencies, the workflow uses Kakadu [17] for encoding and decoding the uncompressed image files and Tesseract [15] as the OCR engine, but the workflow can easily be modified to use other (commercial) JP2 encoding/decoding or OCR alternatives by simply adapting the configuration of the corresponding tool service of the workflow ("kdu_enc", "kdu_dec" for JP2 encoding and decoding, and "tess_orig", "tess" for OCR in Figure 1). Additionally, ImageMagick [18] is used in the "im_remove_compression" tool service for practical reasons in order to remove any compression from the input image files (e.g. LZW TIFF). The "CalculateLevenshteinDistance" beanshell depends on the above mentioned Apache commons-lang library [16] that must be present in the "lib" directory within the Taverna home directory [19]. Gnuplot [20] is used for creating a diagram that visualises the results.

Results

The results from the runs of the Taverna workflow experiment are gathered and aggregated by the "AggregateResults" beanshell service [21] in Figure 1. The main output of the workflow is a text file ("datafile" output port in Figure 1) with the following results:

The overall file size reduction as the ratio of the total file size of the images sample after and before compression (fourth column of the output data file, "datafile" in Figure 1).

The mean Levenshtein distance of each compression parameter value over the image sample with the corresponding standard deviation (second and third column of the output data file respectively, "datafile" in Figure 1).

The data file is directly plotted by the gnuplot tool service available in the "diagram" output port in Figure 1. Additionally, the Tesseract results are displayed in the "tess_result" output for visual inspection.

Figure 2 shows the diagram for the first test sample of ten 8-bit greyscale TIFF image files of book pages using Tesseract OCR version 2. The left y-axis represents the Levenshtein distance and the right y-axis the overall file size reduction in percent. The x-axis represents the increasing lossy JP2 compression which corresponds to a decreasing value for Kakadu's parameter "rate". The upper line shows the overall file size reduction in percent and the lower line shows the mean Levenshtein distance with bars that indicate the standard deviation over the image sample.

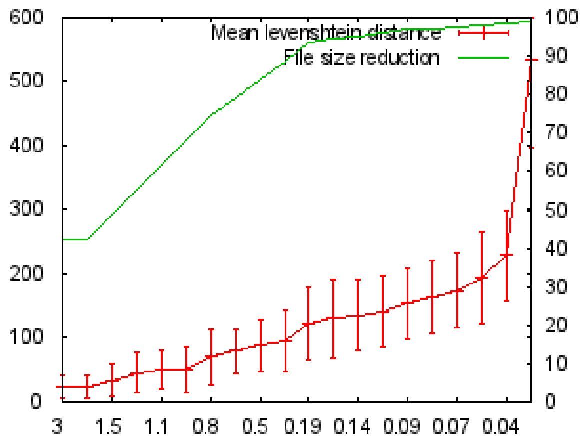

***Figure 2**. Overall file size reduction (upper line) compared to mean Levenshtein distance (lower line with bars for the standard deviation). Data sample: ten 8-bit greyscale TIFF images, Compression: Kakadu's "rate" parameter, OCR: Tesseract version 2.*

Obviously, the mean Levenshtein distance constantly increases with a decreasing "rate" parameter value of Kakadu. It is important to point out that this does not mean that the quality of OCR is getting worse. It only means that the difference compared to the OCR result of the original uncompressed image file is getting bigger. In this case, the insight that is obtained by this result is only the fact that using the parameter "rate" with Kakadu, lossy JP2 compression always has an effect on the OCR, and that the effect steadily increases with a decreasing parameter value using this data sample.

Based on the same experimental setting, Figure 3 shows the combined results of two workflow executions, one using Tesseract version 2 and one using Tesseract version 3.

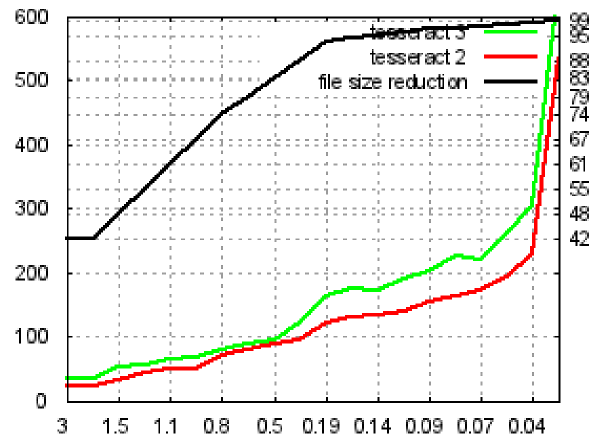

***Figure 3**. Overall file size reduction (upper line) compared to mean Levenshtein distance (lower line: Tesseract version 2, middle line: Tesseract version 3). Data sample: ten 8-bit greyscale TIFF images, Compression: Kakadu's "rate" parameter, OCR: Tesseract version 2 vs. Tesseract version 3.*

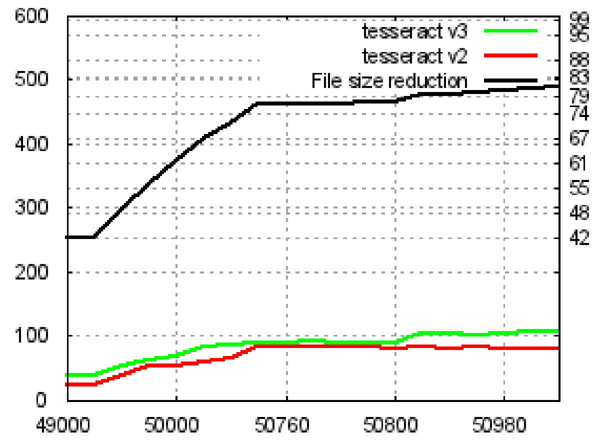

***Figure 4**. Overall file size reduction (upper line) compared to mean Levenshtein distance (lower line: Tesseract version 2, middle line: Tesseract version 3). Data sample: ten 8-bit greyscale TIFF images, Compression: Kakadu's "slope" parameter, OCR: Tesseract version 2 vs. Tesseract version 3.*

It can be concluded that the Levenshtein distance is always higher when using Tesseract version 3 for all compression parameter values when using the fixed bit rate compression parameter "rate" of Kakadu. It can also be observed that when the "rate" parameter value falls below 0.5 the effect is getting inreasingly more significant for Tesseract version 3 compared to Tesseract version 2.

Figure 4 shows the same experimental setting with the only difference that Kakadu's "slope" parameter (variable-rate encoding) is used.

Using Kakadu's "slope" parameter, the effect of the lossy JP2 compression on the OCR result remains moderate for all "slope" parameter values. This is in line with the fact that some partners of Google book digitisation projects agreed on not using the fixed bit rate compression for lossy JP2 compression of book page images [16] but recommend a variable-rate encoding ("slope" parameter in Kakadu) instead:

"As a single method of encoding for mass digitization variable-rate encoding performs better than other methods to balance perceived image quality and file size [8]."

Figure 5 shows the results of the same experimental settings as for Figure 3 with the only difference of using another image sample (ten 24-bit colour images).

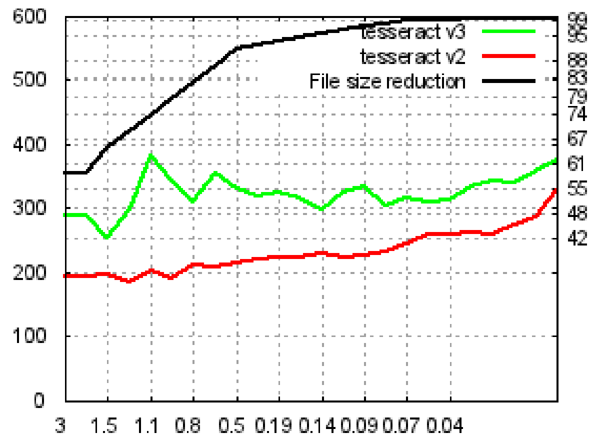

***Figure 5**. Overall file size reduction (upper line) compared to mean Levenshtein distance (lower line: Tesseract version 2, middle line: Tesseract version 3). Data sample: ten 24-bit colour TIFF images, Compression: Kakadu's "rate" parameter, OCR: Tesseract version 2 vs. Tesseract version 3.*

It can be observed that the deviation for this image sample is significantly higher compared to the first image sample for the lossy JP2 compression regardless of the compression parameter value used. Furthermore, the Tesseract 3 results have a local peak for Kakadu's rate parameter value 1.1. On the basis of this result, a Levenshtein distance threshold can be defined that helps to detect statistical outliers.

Closing

Coming back to the initial question, 'If a measure that cannot be considered as an indicator of OCR accuracy can still be used as a heuristic for detecting bad quality in digital images representing text?' We conclude that it can be used for detecting "outliers" that deserve further inspection. One possibility is to use the workflow presented here for applying lossy JP2 compression to a large image collection of book or newspaper pages so as to find out a sensible threshold for the Levenshtein distance that is determined by executing a variety of experiments, and then restricting the application of this workflow to a (representative) sample. Those images that exceed the defined threshold are used to build a set for manually assessing the appropriateness of the JP2 compression settings.

Possible further work would analyse the correlation of the heuristic measure with OCR accurracy measured against Ground Truth using a set of annotated image files.

Acknowledgment

Part of this work was supported by the European Union in the 7th Framework Program, IST, through the SCAPE project, Contract 270137.

References

[1] Gillesse, Robèrt, Judith Rog, and Astrid Verheusen: "Alternative File Formats for Storing Master Images of Digitisation Projects", In: Access, 2008, URL http://www.myopenarchive.org/documents/view/76, p 21.

[2] National Library of Norway. "Digitization of Books in the National Library: Methodology and Lessons Learned.", 2007, Accessed at http://www.nb.no/content/download/2326/18198/version/1/file/digitizing-books_sep07.pdf, p 6.

[3] Buckley, Robert; Tanner, Simon: "JPEG 2000 as a Preservation and Access Format for the Wellcome Trust Digital Library", London, 2009, Accessed at http://library.wellcome.ac.uk/assets/wtx056572.pdf.

[4] Robert Sharpe, Christy Henshaw. (2010) "Managing "Visually lossless" compression with JPEG2000. In: IS&T Archiving 2010: Preservation Strategies and Imaging Technologies for Cultural Heritage Institutions and Memory Organisations, 1-4 June 2010, Den Haag, The Netherlands, p.107-112.

[5] http://jbig2.com/jb2com_technical_advantages.html

[6] Tanner, Simon et al, "Measuring Mass Text Digitization Quality and Usefulness. Lessons Learned from Assessing the OCR Accuracy of the British Library's 19th Century Online Newspaper Archive.", In: D-Lib Magazine 15, 2009, Accessed at http://www.dlib.org/dlib/july09/munoz/07munoz.html

[7] Van Beusekom, Joost et al.: "Automated OCR Ground Truth Generation", In: 8th IAPR International Workshop on Document Analysis Systems, 2008, pp. 111-117.

[8] Chapman, Steven et al, 2007. Page Image Compression for Mass Digitization, inArchiving 2007, Final Program and Proceedings. Accessed at http://preserve.harvard.edu/massdig/hul_study/IST_PageImageCompression_preprint.pdf

[9] Sven Schlarb, Edith Michaeler, Max Kaiser, Andrew Lindley, Brian Aitken, Seamus Ross, Andrew N. Jackson: „A case study on Performing a Complex File-format Migration Experiment using the Planets Testbed", in: Archiving 2010. Preservation Strategies and Imaging Technologies for Cultural Heritage Institutions and Memory Organisations. Final Program and Proceedings, Springfield 2010.

[10] Clemens Neudecker, Sven Schlarb, Zeki Mustafa Dogan, Paolo Missier, Shoaib Sufi, Alan Williams, and Katy Wolstencroft. 2011, "An experimental workflow development platform for historical document digitisation and analysis.", in: Proceedings of the 2011 Workshop on Historical Document Imaging and Processing (HIP '11), 2011, pp. 161-168.

[11] D. Hull, K. Wolstencroft, R. Stevens, C.A. Goble, M.R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services.," Nucleic Acids Research, vol. 34, 2006, pp. 729–732.

[12] C. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, and D. De Roure, "myExperiment: a repository and social network for the sharing of bioinformatics workflows," Nucleic Acids Research, 2010.

[13] Andrzej Bialecki, Mike Cafarella, Doug Cutting, Owen O'Malley, "Hadoop: A Framework for Running Applications on Large Clusters Built of Commodity Hardware", 2005, http://hadoop.apache.org/.

[14] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters", 2008, pp. 107-113.

[15] http://code.google.com/p/tesseract-ocr

[16] http://commons.apache.org/lang/api-2.4/org/apache/commons/lang/StringUtils.html, see method getLevenshteinDistance

[17] Kakadu version 6.3.1 command line application based on the JPEG 2000 Developers' Toolkit. See http://www.kakadusoftware.com

[18] http://www.imagemagick.org

[19] http://www.taverna.org.uk/documentation/faq/customising-taverna/taverna-home-directory/

[20] http://www.gnuplot.info

[21] http://www.mygrid.org.uk/dev/wiki/display/taverna/Beanshell

Author Biography

Sven Schlarb received his PhD in Humanities Computer Science from the University of Cologne. He joined the Austrian National Library in 2008, participated in the EU funded projects PLANETS and IMPACT, and is now participating in the SCAPE project. Before, he worked as a web software developer in Cologne, and as software engineer (C++/Java) and SAP support consultant at SAP in Madrid.

Clemens Neudecker holds a M.A. in Philosophy, Computer Science and Political Science from the University of Munich. Since 2010 he works as a Technical Project Manager for the European projects team in the Innovation and Development department of the National Library of the Netherlands. Before, he worked at the Bavarian State Library where he was involved in numerous digitisation projects.