

Digital Weight Watching: Reconstruction of scanned documents

Tim Gielissen and Maarten Marx
ISLA, University of Amsterdam
Science Park 107 1098 XG Amsterdam, The Netherlands
maartenmarx@uva.nl

ABSTRACT

Scanned and OCR'd data leads to large file sizes if facsimile images are included. This makes storage of, and providing online access to large data sets costly. Manually analyzing such data is cumbersome because of long download and processing times. It may thus be advantageous to reconstruct the scanned documents as documents without scanned images which nevertheless closely resemble the original. We have done this reconstruction for a data set of Dutch parliamentary proceedings with positive results. 1.5% of the original storage space was needed, while the documents resembled the originals to a high degree. We describe the reconstruction process and evaluate the costs, the benefits and the quality.

Categories and Subject Descriptors

H.4.m [Information Systems]: Miscellaneous

Keywords

XML, Information Extraction, Scanned Documents

1. INTRODUCTION

This paper addresses the fact that scanned and OCR'd documents tend to be very large in file size because of the inclusion of facsimile¹ images. This makes them expensive to store, expensive to serve over the internet, and cumbersome to handle by users. We present a case-study in which we extracted the content and the structure from a large digitized corpus and reconstructed the documents from scratch “as new”. This yielded much smaller (less than 1% of the original size when stored in gzipped XML, 1.5% when stored as PDF) and far better readable documents which in addition were easier to browse because of added hyperlinked structure.

¹By a facsimile image, we mean a document that visually resembles the original.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AND '09, July 23-24, 2009, Barcelona, Spain

Copyright © 2009 ACM 978-1-60558-496-6 ...\$5.00

We present the data, describe our techniques, evaluate the results and generalize them to other cases.

The novelty of our work is located in the way we reconstruct the original files. The reconstruction is based on an almost (we only kept the original separation of the text into pages) purely semantical description of the original data. In addition we showed that all this is possible using only two well-described declarative programming languages: XSLT 2.0 and \LaTeX .

Outline. The rest of this introduction contains the actual problem we are addressing, the use case we present and related work. Section 2 describes the data; Section 3 the two transformation processes: structure extraction and document reconstruction. We evaluate the quality of these two transformations in Section 4, and conclude in Section 5.

Manual analysis of noisy data. Much, in particular qualitative, research involving digital documents is done “by hand”. This is especially true in the social sciences and in the humanities. Current large-scale digitization efforts of important collections make more data much more easily available, and offer new technologies, most importantly keyword search over the full (often OCR'd) text.

Keyword search makes it much easier to investigate unfamiliar documents or collections. Physically enormous collections are now available at the desktop. A common scientific method of enquiry is searching and browsing through texts and collections of texts. With a good search engine and fast internet access it is easy to go through a large number of documents in a short time, in order to get an overview of the data, or to find specific information.

This process works smoothly if documents are relatively small in size and can be handled by one simple and fast program. But the technical nature of the documents often prohibit fast workflows. As an example, it may very well take more time to open a PowerPoint presentation than to decide that it can be discarded as non-relevant. Unfortunately, digitization processes create large documents, even if the originals are relatively simple text documents. If the original layout and look-and-feel is important, or if mistakes caused by OCR are unwanted, scanned documents must be presented using facsimiles. If they have to be readable these

images get very large.

One of the most pressing problems arising from the large file sizes are long download times. For instance, with a fast (12 Mbit/second) internet connection it takes in the optimal case 10 seconds to download an average document (16 Mb) from our collection. In reality this speed is often up to 20 times slower. Such long download times prohibit natural workflows.

We believe that the facsimiles need to be available as the ultimate source but that in a search and browse interaction process with the data the alternative, much smaller, version based on the XML is preferable. Users get results faster, they get clean hyperlinked files, and they use much less bandwidth. Of course the document contains both OCR and layout errors, but users are able to cope with this. Once a user knows exactly which document she wants to consult, the large facsimile PDF can be downloaded.

Use case: Dutch parliamentary proceedings. The Netherlands have parliamentary proceedings since 1814. From 1995 these are available electronically as PDF files. The Dutch Royal Library together with the Dutch parliament have scanned and OCRed all proceedings from 1814 until 1995 and have so created the first complete copy of these proceedings. The collection consists of 2.5 million pages which physically span 150 meters. The digital copy needs approximately 30 Terra Byte storage space.

The proceedings are available online at

<http://www.statengeneraaldigitaal.nl>.

Each document is a combination of files: metadata in XML, a JPEG image for each page, the OCRed text in an XML wrapper, and an MPEG21-DIDL file describing the connections between all these files [1].

Related work. Within digital curation research XML is seen as an important data format for storing data for long periods of time. The National Archives of Australia intend to store all they have in XML and developed a software tool for this, XENA (<http://xena.sourceforge.net>). Another development in this direction is the UVC (Universal Virtual Computer) developed by IBM and the Dutch Royal Library [16]. The transformation of scanned and OCRed documents into XML is not the topic of this paper and has been described in [3]. For the use of XML as a format for governmental documents, see [15]. The extra step we take in this paper –automatically reconstructing the original document from the XML-version as a lightweight truly electronic file– is, up to our knowledge, not yet described in the literature. Here we focused on the 'look-and-feel' of the documents. An extensive body of research on sustainable digital preservation of properties exists under the name of *significant properties* [9].

2. DESCRIPTION OF THE DATA

Our experiment uses six years of proceedings of plenary meetings the Dutch Parliament, from 1980 to 1985. Each document contains the meeting notes of one day. Table 1 lists statistics on the sizes.

On average one document contains 51 thousand words, is 50 pages long and has a file size of 16.5 Megabyte. Because each document represents the meeting notes of one complete day, on an average day in Dutch parliament some 50 thousand words are officially spoken.

The largest document is 49 MB (151 thousand words) and the smallest is just 1 page, with 382 words. At the meeting of this one page document less than half of the Dutch MP's were present and then by law the meeting cannot start.

Figure 1 shows the facsimile of a typical page.



Figure 1: A typical page of the Dutch parliamentary proceedings. Page 2077 of the meeting of May 21, 1968. Available at http://resourceesgd.kb.nl/SGD/19671968/PDF/SGD_19671968_0000410.pdf (22MB).

In the next section we describe how we transform the scanned and OCRed PDF documents into XML. This leads to a great reduction in size. Whereas the original set of 843 documents was 13,62GB, the same set in XML takes only 295 MB, a reduction of 2 orders of magnitude. A further reduction to 88.3 MB can be obtained using gzip, yielding a total reduc-

year	Number of documents	Total file size	Number of words	Number of pages
1980	143	2,52 GB	7.943.904	7.709
1981	124	1,76 GB	5.613.432	5.405
1982	133	1,76 GB	5.552.685	5.544
1983	150	2,50 GB	7.714.903	7.612
1984	147	2,54 GB	7.870.318	7.750
1985	146	2,56 GB	8.251.097	8.081
Total	843	13,62GB	42.946.339	42.101

Table 1: Description of the test corpus.

tion of 99,4%. Note that gzipping the original PDF files has hardly any effect (less than 5% reduction in size). The reconstructed PDF files take 205 MB to store.

3. DESCRIPTION OF THE TRANSFORMATION

The transformation of a scanned and OCRred PDF document into a PDF document that closely resembles the original, but without the facsimile image, involves two steps. First, the structure that is implicit in the document is made explicit using a variety of text extraction techniques [3]. Next, the resulting XML document is transformed into a PDF document without a facsimile image using XSLT and L^AT_EX. In this section we describe both steps.

3.1 Making structure explicit: from PDF to XML

All documents contain structure. Often we can easily recognize titles, paragraphs and page numbers on the basis of their layout (e.g., position or size) and their content (e.g., a number) for example. This structure is often not explicit in digital form, especially not when the document is scanned and OCRred. If the structure of documents in a corpus is standardized to some degree, the structure can be made explicit in digital form automatically [2].

The Dutch parliamentary proceedings shows this kind of standardization of document structure. All elements of a proceeding, for example topics and speakers, are represented in their own distinct way. This enables the automatic recognition of these elements. We programmed the text and structure extraction as an Extract-Transfer-Load process [13] consisting of eight steps.

First we extract the text from the PDF using the open source program `pdftohtml`² with the `-xml` option. This yields an XML file with for each line of text four coordinates which indicate the bounding box of that text. Multiple columns are detected and preserved. Some font and layout information is preserved but not all. The XML structure is simple and flat:

```

root  → (page)*
page  → (text)*
text  → (#PCDATA,b,i,span)*

```

The second step involves cleaning the output from `pdftohtml`. We make sure that the output is well-formed XML

²<http://pdftohtml.sourceforge.net/>

and we solve problems with diacritics. In this step, we also fix the most common OCR errors. We found one specific error quite often: the OCR inserts a space before the last letter of a word. For example, the token `wij` is OCRred as `wi j`. A regular expression designed to fix this problem matches 3.1% of all the lines in the text corpus (i.e., one line in one column in the original PDF as in Figure 1). A sample of 100 matches taken at random indicated that this regular expression fixes the problem in 93% of the cases and makes it worse in 7% of the cases (by incorrectly adding a single letter to the word in front of it).

The values indicating the position and size of the bounding box of the text are normalized in the third step as we found they can differ among different devices.

In the fourth step, we analyze the document’s layout. The margins, the number of columns and the header and footer are detected and marked. For this, we use the position of the text elements and the (deducted) position of the whitespace. Using this information, we sort the text of the body (i.e., not belonging to the header or the footer) in reading order in the fifth step.

During the sixth and seventh step, different markers are placed in the text to signal the start of different elements in the document. In the sixth step we place markers indicating the position of text elements in the document. We place markers on places where paragraphs begin (they are indented), where there is whitespace and when a new column starts. This information is used in the seventh step where markers are placed based on the content of the document. In the seventh step we use regular expressions to recognize standardized structure. The start of a statement for example, is represented as follows:

Mevrouw **Swenker** (VVD):

This adheres to the following structure: title, last name, party name within parenthesis, colon. This is then followed by the statement this person made. The start of a statement can only begin after a whiteline marker or the start of a new column. So in our XML, we convert this to:

```
<speechstart speaker='Swenker' party='VVD' .../>
```

with the ... containing additional information.

We now have an XML document that is flat and contains markers that mark the beginning of structural elements, but

not the end. In the last step we replace the markers by XML tags that enclose the entire element. This is done by performing a cascade of groupings starting with the elements which need to be most deeply nested: the paragraphs p. XSLT 2.0 has a very useful command for this task: `xsl:for-each-group`. This command, new in XSLT 2.0, replaces the so-called Muenchian method which was needed in version 1.0 of XSLT [8].

The result is an XML file with the same text as the original document but with explicit structure. The file is valid with respect to a rich Relax NG schema, constraining both the structure of the XML-tree as well as the values of many attributes. The structure can be used for many purposes, e.g. to analyse the structure of the debates [6].

3.2 Reconstruction of the originals: from XML to PDF

Because the structural elements of the documents are made explicit in the XML file, it is possible to reconstruct the original PDF with high resemblance. We use a combination of XSLT 2.0 and \LaTeX for this process. We created a XSLT stylesheet that transforms the XML file described in the previous subsection into a \LaTeX file. We briefly describe this transformation.

First, the stylesheet writes a general preamble that loads all necessary packages and specifies layout information for the document. Some values are copied from the XML file, like the value that indicates the number of columns.

After the preamble, the document itself begins. For every element specified in the Relax NG schema, we defined a template (for more information about the schema, see [3]). These templates are nested according to the structure of the proceeding. First we have a template for the topics, the highest level of the structure of the proceedings. The stylesheet writes the necessary layout information for the topic and then places the text from the XML file in the \LaTeX file. Next, it applies all the templates for the elements within the topic. This way, we cycle through all the elements in the deep structure of the XML and create the \LaTeX file step by step. If the stylesheet encounters a pagebreak, it redefines the page style (including header and footer) for the next page.

When the \LaTeX file is created, the next step is to create the PDF document with `pdflatex`. Creating the \LaTeX file with XSLT and compiling the PDF file takes about one to two seconds, depending on the size of the file.

An alternative approach for reconstructing the files is to use XSL-FO to produce a PDF document directly with XSLT. We used \LaTeX because it appeared to be easier to obtain fast results.

4. EVALUATION

We give a technical, an information-theoretic and an economic evaluation. The technical evaluation describes the reduction in file sizes obtained and the processing times needed for the reduction. In the economic evaluation we compare the efforts invested in creating the transformation

scripts with the benefits of smaller files and explicit markup information. In the information-theoretic evaluation we look at the quality of the transformations: we evaluate whether information was lost or distorted.

4.1 Benefits of the reconstruction

Size reduction. Our first goal was the reduce the file sizes. This was achieved with a reduction of 2 orders of magnitude. Table 2 lists the results.

	Size in MB	% of original
Original corpus	13.620	
Reconstructed PDF	205	1.5
gzipped XML	88	0.6

Table 2: Total file sizes of the test corpus described in Table 1.

Processing times. Transforming the whole corpus of 13,62 GB into reconstructed PDF's took 25 hours and 50 minutes, an average of 1.8 minutes per document. The table below shows the percentage of the total time that was spent on each of the elements of the entire pipeline, which were described in the section: "Description of the transformation".

pdftohtml	PDF-2-XML (without pdftohtml)	XML-2-PDF
40%	58%	2%

Table 3 shows the processing speeds of the two main steps of the transformation in pages per minute. (Recall that a typical document has about 50 pages.) These times were measured on an Apple MacBook with 2.4 GHz CPU running the OS X 10.5.6 operating system.

PDF-2-XML	XML-2-PDF
27,6 pages/minute	1503,6 pages/minute

Table 3: Processing speeds of the two main transformations.

Both transformations can be done offline and we need only store the reconstructed PDF on the webserver. The transformation from XML to PDF is fast enough to perform it at query time. Recall that an average document is 50 pages. Thus this can be transformed from a gzipped XML file into PDF in two seconds. This transformation can be further optimized as a streaming transformation [10].

Digital sustainability. We aimed to make our transformations in such a way that they can be performed again after minimally 100 years with relative ease. Thus we wanted a minimal dependency on specific software and hardware, and transparent and reproducible transformations [4]. Our goal was to have all steps of the transformation written

in a declarative language with a precisely defined software-independent semantics. XPath 2.0 and XSLT 2.0 meet these standards [7, 8].

For the conversion from PDF to XML we reached this goal except for the first step of the transformation and the final validity check. In the first step we used `pdftohtml -xml` (<http://pdftohtml.sourceforge.net>) to transform the PDF to XML. Unfortunately this program may produce non-well formed XML and it has problems with certain diacritics. We repaired these with a `perl` script and checked for XML well-formedness with `xmllint`. This was also used in the final step to check validity with respect to the schema specified in Relax NG.

The conversion from XML back to PDF is done by an XSLT 2.0 script which produces a \LaTeX source file.

Economics. We calculate what can be saved using the file size reduction based on the prizes for storage and bandwidth set by Amazon, <http://aws.amazon.com/s3/#pricing> at the time of writing. The fixed costs for storing the test collection are still very modest: \$ 30 per year. The variable costs are determined by the number of users and the number of documents they want to see: downloading at Amazon costs \$0.17 per GB. Table 4 lists the costs per year for 3 user models, and three ways of serving: the original collection (as it is now served at <http://www.statengeneraal.digitaa.nl>), serving the transformed PDF’s (as described in this paper), or serving the gzipped XML files only. The monetary savings are two orders of magnitude.

It is hard to quantify the amount of time needed to create the transformation scripts. This depends very much on the complexity and the regularity of the documents and the use of (commercial) ETL tools [13]. We gave our scripts and the Dutch proceedings DTD to third year Information Science students with rather restricted programming experience and asked them to transform German and Belgium proceedings into XML. Within two weeks they created reasonably robust scripts that extracted with high precision.

4.2 Quality of the reconstruction

Quality of the data: from PDF to XML. We now evaluate the transformation from the original PDF file to XML, which was described in 3.1. For each part in the proceedings that we wanted to extract and mark up by XML tags, we scored whether the start- and end-tags were placed correctly. We evaluated two complete days (50 pages). Table 5 shows the percentage of correctly marked elements for 7 typographical features.

We are pleased with these initial scores. The semantically important XML elements, topic and block, were all recognized. The topic description sometimes included too much text (22.2% of the topics), but they were correctly marked as topics. Most of our mistakes were caused by OCR errors which did not let our extraction rules fire. E.g. for recognizing the start of a speech element (and the title, name and party of the speaker) we use the pattern `[title][name][party][:]` as in `[De heer][Van der Spek`

Feature	Score	comments
Topics	77,8%	All recognized, but in 22.2% included too much text
Blocks	100%	
Speakers	88.7%	Caused by OCR errors
Paragraphs	93.5%	
Header	91.5%	Caused by OCR errors
Footer	92.5%	Caused by OCR errors
Stage directions	73.5%	

Table 5: Percentage correctly extracted structural elements.

`]](PSP)][:]`. But sometimes this string is wrongly tokenized by the OCR as `De heer Vander Spek (PSP) :.` OCR mistake repairing, using e.g. the TICL technique [14], will improve our scores considerably.

Quality of the look and feel: from XML to PDF. Our goal was to keep the look and feel of the structure of the documents. This was achieved with great success, see Figure 2 for an indication of the results.

We wanted to preserve the layout of each page as much as possible, but not be overly restrictive. For instance, old and new pages should contain exactly the same characters in the same order, but the words may be broken differently over the lines of texts. Table 6 contains the most important typographical features of these documents and an assessment of the quality of our reconstruction. We tested the similarity each time on ten randomly chosen documents. We score whether the element was visually indistinguishable throughout the complete document. Figure 3 contains examples of visual (in)distinguishability. In the top of the Figure we see two footers (on the left the original). These were scored as being “the same”. Below that we see twice the wording of a *motie* (the original is on the left). The header *Motie* between the two horizontal lines indicates an important structural feature of the text, which is not present in the reconstruction. For that reason these two are scored as different.

23 februari 1999 EK 20	23 februari 1999 EK 20
De voorzitter: De motie-Witteveen-Hevinga (26570, nr. 3) is in die zin gewijzigd, dat zij thans luidt:	De voorzitter: De motie-WitteveenHevinga (26570, nr.3)isindie zin gewijzigd, dat zij thans luidt: Motie
Motie	De Kamer,
De Kamer,	gehoord de beraadslaging,
gehoord de beraadslaging,	overwegende, dat er op provinciaal c.q.

Figure 3: Example of two features scored as the same (top) and two scored as different (bottom). At the left is the original, at the right the reconstructed document.

An error analysis indicated that almost all mistakes are due to OCR errors or to inconsistent layout in the original.

Daily use	Original Collection	Transformed Collection	Transformed Collection Client side XML-2-PDF
10 users 10 docs each	\$100	\$2	\$0.64
50 users 10 docs each	\$500	\$11	\$3
100 users 10 docs each	\$1000	\$21	\$6

Table 4: Costs per year in dollars (rounded) for downloading documents stored at Amazon (prices April 2009). Average document sizes are used.



Figure 2: Similarity of layout between original (right) and our copy (left).

Feature	Score
Header	9/10
Footer	7/10
Individual speeches	6/7
Stage directions	
- Members present	4/4
- Start of agenda topics	6/10

Table 6: Score of visual similarity on 10 typographical features. The score k/m indicates that on k of the m documents containing these elements, all elements in that document were visually the same as in the original.

4.3 Additional advantages

Having the data in XML creates numerous analysis possibilities which are impossible to do automatically from the original PDF files. [11] and [6] give a number of examples related to search, in particular focused retrieval and result aggregation. Here we look at several possibilities that become available when making a new PDF file with \LaTeX from XML input.

Hyperlinks The `hyperref` package will create a table of contents that PDF readers like XPDF and Acroread

use.

Table of contents The Dutch proceedings do not contain a table of contents. The table of contents acts as an agenda of the meeting and hence is very valuable. It can be created for free and accurately reflects the structure and order of the meeting.

Select from PDF Users can select text from the PDF file.

What you see is what you get The PDF files available at <http://statengeneraaldigitaal.nl> have a logic which is not obvious to average users. If opened in a PDF reader, one looks at the scanned images. But it is possible to search with Control F, and that yields highlighted hits. However, the search takes place in the OCRd text which may contain errors. Thus words which occur in the text may not be found due to OCR-errors. This can be confusing for users. Also, seeing the OCR errors gives users the opportunity to broaden their search terms and still retrieve what they look for.

Wikification and hyperlinking Names of persons who speak may be hyperlinked to pages with their biographical information [12]. References to other parliamentary documents may be hyperlinked to these documents.

Back of the book index Using machine learning and keyword extraction techniques [5] useful index terms can be extracted and indicated in the running text. Creation of a back of the book index is then automatic using L^AT_EX's `makeindex` command.

5. CONCLUSION

We have shown that reconstructing PDF documents from scanned and OCR'd data is feasible and leads to size-reductions of two orders of magnitude. The quality of the reconstructed PDF files is very good, and in several aspects better than the original. The most important gain of this exercise is the reduction in download time from unacceptably slow to instantaneous. For instance, with a fast (12 Mbit/second) internet connection it takes in the optimal case 10 seconds to download an average document (16 Mb) from our collection. In reality this speed is often up to 20 times slower. But the achieved size reduction to just over 1% makes downloading immediate, even in unfavourable cases.

Our sample of 6 years seems to be representative of the last 80 years of Dutch data, in terms of layout complexity and noisiness of the OCR data³. Preliminary investigations abroad (Belgium, Germany, Ireland) shows that our findings generalize to other parliamentary proceedings corpora. Interesting directions of future research are to exploit the rich structural semantics of these documents in ways described in Section 4.3.

6. REFERENCES

- [1] Koninklijke Bibliotheek. Staten-generaal digitaal. <http://www.statengeneraaldigitaal.nl/backgrounds.html>, 2009.
- [2] AnHai Doan, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Managing information extraction: state of the art and research directions. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 799–800, New York, NY, USA, 2006. ACM.
- [3] T. Gielissen and M. Marx. Exemelification of parliamentary debates. In *Proceedings of the 9th Dutch-Belgian Information Retrieval Workshop (DIR 2009), Twente, The Netherlands*, pages 19–25, 2009.
- [4] H. M. Gladney and R. A. Lorie. Trustworthy 100-year digital objects: durable encoding for when it's too late to ask. *ACM Trans. Inf. Syst.*, 23(3):299–324, 2005.
- [5] Anette Hulth, Jussi Karlgren, Anna Jonsson, Henrik Boström, and Lars Asker. Automatic keyword extraction using domain knowledge. In *Proceedings CICLing 2001*, pages 472–482. Springer, 2001.
- [6] R. Kaptein, J. Kamps, and M. Marx. Who said what to whom? Capturing the structure of debates. In *Proceedings SIGIR*, 2009.
- [7] M. Kay. *XPath 2.0 Programmer's Reference*. Wrox, 2004.
- [8] M. Kay. *XSLT 2.0 3rd edition Programmer's Reference*. Wrox, 2004.
- [9] G. Knight and M. Pennock. Data without meaning: Establishing the significant properties of digital research. In *iPRES 2008 Conference Proceedings*, 2008.
- [10] Bertram Ludäscher, Pratik Mukhopadhyay, and Yannis Papakonstantinou. A transducer-based xml query processor. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 227–238. VLDB Endowment, 2002.
- [11] M. Marx. Long, often quite boring, notes of meetings. In *Proceedings ESAIR 2009 : Exploiting Semantic Annotations in Information Retrieval*, 2009.
- [12] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242, New York, NY, USA, 2007. ACM.
- [13] E. Rahm and H.H Do. Data cleaning: Problems and current approaches. *IEEE Techn. Bulletin on Data Engineering*, 23(4), 2000.
- [14] M. Reynaert. Non-interactive OCR post-correction for giga-scale digitization projects. In *Proceedings. CICLing (Computational Linguistics and Intelligent Text Processing, 9th International Conference)*, pages 617–630, 2008.
- [15] A. Salminen. Building digital government by XML. In *Proceedings of the Thirty-Eighth Hawaii International Conference on System Sciences*. IEEE Computer Society, 2005.
- [16] J. R. Van Der Hoeven, R. J. Van Diessen, and K. Van Der Meer. Development of a universal virtual computer (uvc) for long-term preservation of digital objects. *J. Inf. Sci.*, 31(3):196–208, 2005.

³We had access to the data from 1960–1995 and from 1930 and 1931.