Federal Register Document Image Database
NIST Special Database 25 - Volume 1

NISTIR 6245

Michael D. Garris, Stanley A. Janet, and William W. Klein

National Institute of Standards and Technology
Building 225, Room A216
Gaithersburg, MD 20899

ACKNOWLEDGEMENTS

We would like to acknowledge the Department of Defense who provided funding and resources in conjunction with NIST to support the development of this document image database.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

Federal Register Document Image Database
NIST Special Database 25 (Vol. 1)

Michael D. Garris (mgarris@nist.gov),

Stanley A. Janet, and William W. Klein

ABSTRACT

A new, fully-automated process has been developed at NIST to derive ground truth for document images. The method involves matching optical character recognition (OCR) results from a page with typesetting files for an entire book. Public domain software used to derive the ground truth is provided in the form of Perl scripts and C source code, and includes new, more efficient string alignment technology and a word-level scoring package. With this ground truthing technology, it is now feasible to produce much larger data sets, at much lower cost, than was ever possible with previous labor-intensive, manual data collection projects. Using this method, NIST has produced a new document image database for evaluating Document Analysis and Recognition technologies and Information Retrieval systems. The database produced with this method contains scanned images, SGML-tagged ground truth text, commercial OCR results, and image quality assessment results for pages published in the 1994 Federal Register. These data files are useful in a wide variety of experiments and research. There were roughly 250 issues, comprised of nearly 69,000 pages, published in the Federal Register in 1994. This volume of the database contains the pages of 20 books published in January of that year. In all, there are 4711 page images provided, with 4519 of them having corresponding ground truth. This volume is distributed on two ISO-9660 CD-ROMs. Future volumes may be released, depending on the level of interest.

Keywords: CD-ROM, database, document, ground truth, image, information retrieval, OCR, public domain software

1. INTRODUCTION

In 1997, the National Institute of Standards and Technology (NIST) in conjunction with the Department of Defense (DoD), began a project known as the Metadata Text Retrieval Conference (METTREC). The purpose of the project was to develop and evaluate the interface between Document Analysis and Recognition (DAR) technology with Information Retrieval (IR) technology. Specifically, the use of non-text items in a document (called metadata) was to be explored and its impact on retrieval performance evaluated. A need for technology development and evaluation in this area is much needed as the demand for large, automatically built, digital libraries continues to increase. Unfortunately, the project was discontinued due to a lack of interest and capability within the DAR and IR communities. This is discussed further in Reference [1].

In order to conduct DAR/IR integrated experiments and technology evaluations, a large database of prepared materials for training and testing is required. This paper documents the data that was collected in preparation for the first round of METTREC evaluations. Even though there is no longer a formal evaluation pending, the data produced for the project is valuable both to the DAR and IR research communities. To this end, the data preparation was completed and published on CD-ROM as a *NIST Special Database 25*.

This database fundamentally contains images of pages and their corresponding text from the 1994 *Federal Register* (FR). The United States Government Printing Office (GPO) prints the FR each work day to record the transactions of the government. The FR is the official daily publication for Rules, Proposed Rules, and Notices of Federal agencies and organizations, as well as Executive Orders and other Presidential Documents. Each issue is printed and bound into one or (on rare occasions) more books. Nearly 69,000 pages were published in the FR in 1994.

There are 20 daily FR issues (totaling 4711 pages) included in this volume of the database. These pages were taken from the FR books published for the entire month of January in 1994. Each image was scanned binary at 15.75 pixels per millimeter (ppm), equaling 400 pixels per inch (ppi). With each image is a file that contains the textual content of the page. The images and associated text can be used to evaluate alternative DAR technologies, and they can be used in IR experiments where retrieval performance can be compared between retrieving the pages of text provided with the database and retrieving the corresponding text produced by DAR. In this way, the impact of recognition errors on retrieval performance can be evaluated (a key goal of METTREC).

The text of each FR page in this database is formatted in Standard Generalized Markup Language (SGML) [2]. This representation is commonly used in IR experiments and has been used extensively in the TREC conferences [3]. Included in the ground truth text files is a set of tags that represent a small collection of logical structures contained in the FR books. These include such things as notices, tables, and footnotes. An SGML Document Type Definition (DTD) is provided that defines these structures. Recipients of the database may find it useful to augment these tags to suite there own needs, as this is a reasonable way to encode and utilize metadata.

The database is 1.27 gigabytes in size and contains much more than just images and their corresponding text. Optical character recognition (OCR) results generated from Xerox's ScanWorX API[1] are provided for each FR page in the database. These files are representative of the performance of commercially available OCR technology. In addition to the text recognized on each page, these files contain (among other things) rotation estimates of the page, word and character confidence values, and word bounding boxes. All of these are useful for a variety of experiments. The OCR application program used to generate this information is included in the distribution for use by those who have ScanWorX API.

The results of an image analysis process developed at Los Alamos Laboratory [4] are also provided for each image in the database. The quality of an image is automatically assessed based on several factors, and these factors are used to predict an OCR word error rate. The measured factors and the predicted OCR error rate are reported for each image in the database. A comprehensive discussion of all the various types of data provided with this distribution is provided in Section 3.2.

A document image database of this magnitude is every expensive to produce. Approximately 2.5 person-years were invested. Labor to scan in the entire 1994 FR was contracted to a scanning

[1] Specific hardware and software products identified in this paper were used in order to adequately support the development of the technology described in this document. In no case does such identification imply recommendation or endorsement by NIST, nor does it imply that the equipment identified is necessarily the best available for the purpose.

service bureau. Each of the resulting images had to be validated (at NIST) as being a complete image of reasonable dimensions (not blank or truncated) and verified to contain its corresponding page. This machine-assisted validation process was conducted at NIST and is documented in Reference [5].

While validating the images was necessary, the bulk of the labor was invested in creating an automated method for generating the ground truth text for each FR page image. Original GPO typesetting files, used to print each book of the 1994 FR, were obtained and are included in the database in their entirety. Portions of these files have been used in previous TREC evaluations [3]. The GPO files contain the text printed in each FR book along with binary typesetting codes. No page boundaries are provided in these files; therefore, a method for locating and extracting a specific FR page of text within the GPO files was developed. This method involves matching noisy OCR results from the corresponding page image to the GPO-provided data. This process is discussed in detail in Section 2.

Not only is the data produced by this method provided in this distribution, but a public domain software implementation of this process is also provided in the form of Perl scripts and C programs. This software precisely specifies how the ground truth text was derived, it enables the method to be applied to other document collections, and it makes it possible for the recipient to improve upon the method. With this new ground truthing technology, it is now feasible to produce much larger data sets, at much lower cost, than was ever possible with previous labor-intensive, manual data collection projects.

The organization of the software distribution and instructions for installing it are provided in Section 3.1. Manual pages for the various programs and utilities are included in the reference guide found in Appendix A.

At the heart of the ground truth production process is a robust and efficient string alignment technology. Alternative alignment methods were explored, including those based on the Levenstein distance [6] and an alignment algorithm developed by the University of Washington [7]. These algorithms are implemented using dynamic programming and in general execute very slowly and consume large amounts of memory. The University of Washington algorithm apparently attempts to find an *optimal* alignment in a way that requires significantly more time to compute as the discrepancies between the two strings increase. As a result, the use of these types of algorithms becomes prohibitive when analyzing the recognition performance on low quality documents. To compensate, NIST developed a new alignment technology based on a greedy implementation of a recursive maximum substring matching algorithm. This approach is designed to produce reasonable alignments between two different versions of a full page of text in reasonably fast and relatively constant time regardless of the level of discrepancy.

This public domain alignment technology is provided in the distribution and is bundled within a word-level OCR scoring package. Word-level scores are provided for each FR page image in the database and were computed by matching the ScanWorX results of each page image with the database's corresponding derived ground truth text.

Quality assessment and control was key to automatically generating ground truth text. This was extremely important because there was noise inherent throughout the process. The OCR text contains varied degrees of errors depending on the page's composition and image quality, and the GPO files presented certain challenges onto themselves. To account for this, two levels of quality assessment were imposed in the production of the ground truth text. First, a located

portion of GPO text (hypothesized to belong to a specific page image) was rejected if the location didn't fit trends derived from its neighboring pages. This was a binary accept/reject decision.

Once the body of a page's text was located in the GPO files, the page's starting and ending position had to be determined. The string alignment technology was applied and points where alignment sufficiently degraded (at both front and end) were determined to be the page boundaries. The accuracy of these selected page boundaries varies depending on the quality of the OCR and the condition of the text in the GPO files at or near the actual page boundaries. Due to the level of uncertainty involved in both the OCR and GPO texts, a method of estimating the amount of error in the derived ground truth text was developed. An error estimate is provided for each ground truth text file in the database. This is a conservative estimate that allows researchers to select files for experimentation based on their own criteria and tolerance. Both quality assessment steps are described further in Section 2.2.5.

Volume 1 of the Federal Register Document Image Database, *NIST Special Database 25*, is distributed on 2 CD-ROMs in ISO-9660[8] data format. (This format is widely supported on UNIX, VMS, DOS and Windows-based computers.) Information on how to order this database is given in Section 4.

2. CREATING THE DOCUMENT IMAGE DATABASE

This section provides an overview of how the FR document image database was created. The composition of the FR is first discussed, and then the process of generating the database is presented. Example FR page images are shown in Appendix C.

2.1 The *Federal Register*

This database contains pages scanned from the 1994 FR. The GPO prints the FR each work day to record the transactions of the government. Typically, the GPO publishes one book per day, and each book contains three major sections:

Prefix Pages: Labeled in the database as 'a' files, these pages consist of a hard cover page, a soft cover page, and content pages. Figure 22 illustrates a hard cover page that contains the date, the volume number, an address label area, and a postal class identification; it is printed on high grade paper to minimize or prevent damage incurred by postal processing and handling. (All other pages are printed on recycled newspaper-quality paper.) As illustrated in Figure 23, a soft cover page identifies the date, the volume number, and the page numbering sequence for the body pages contained within the book. A content page, illustrated in Figure 24, contains a page heading that includes an upper case Roman numeral page number. The first content page contains general information with regard to the FR and its usage. Each subsequent page identifies the contents of the specific book. There are typically less than 10 prefix pages printed in each FR issue.

Body Pages: Labeled in the database as 'b' files, these pages represent the bulk of each FR issue and provide a record of the meeting notices, proposals, and transactions of the United States government for the day. There are two types of body pages: *section* pages and *detail* pages. A section page, illustrated in Figure 25, is similar in appearance to a soft cover page and is used to divide the FR into distinct parts. A section page contains the name of the issuing agency, the

Code of Federal Record (CFR) title and part(s) affected, and a brief description of the section's subject; this type of page does not contain a page number field. A detail page, illustrated in Figure 26, elaborates Presidential and Executive Order(s), Rules and Regulations, Proposed Rules, and Sunshine Act Meeting Notices. Each page contains a page heading that includes an Arabic page number. Page numbers run consecutively across FR books. Typically, 200-300 pages are printed in each FR issue.

Appendix Pages: Labeled in the database as 'c' files, these pages provide reader aids which allow a reader of the FR to access information and to index specific information contained within previous issues. An appendix page contains a page heading that includes a lower case Roman numeral page number. Typically, there are less than 10 appendix pages printed with each FR issue. An example of an appendix page is illustrated in Figure 27.

With the exception of cover and section pages, each page of the FR is printed with a page heading that includes a text banner printed above two horizontal lines. The text banner contains information that identifies the document, the volume, the date, the topic, and a page number. Blank pages are not included in this database.

The FR for 1994 was chosen to be the source for this database because it is:

1. a complete set of documents within the public domain;
2. a large collection containing over 250 issues consisting of nearly 69,000 pages of information;
3. a structured document set;
4. a collection of pages containing significant variations in print and image quality;
5. a set of documents for which the text for the entire collection is stored within electronic files that can be used as ground truth for evaluating OCR and IR systems.

2.2 Creating the Document Image Database

Figure 1 illustrates the steps taken to generate the images and ground truth text files included in this database. The original GPO typeset files were obtained through a data provider, and all the FR books issued in 1994 were obtained from the NIST Reference Library as the books were being removed from circulation. Having the books enabled us to get images of each page, and having the GPO files provided us with the text printed in each book. The goal was to create a process whereby OCR results from each page could be matched against the GPO files, and the text corresponding to a specific page be located and extracted and used as ground truth.

The bindings of the books were cut and nearly 69,000 pages were scanned by a service bureau at 15.75 ppm (400 ppi) binary. Upon scanning, each image was validated at NIST according to Reference [5]. Scanning and validating the images is discussed further in Section 2.2.1.

After the images were validated, they were processed by ScanWorX API[1], and OCR results were recorded. OCR text is provided with every page in the database (for which ScanWorX terminated successfully). The quality of this text varies significantly from page to page, depending on a page's composition and image quality. The impact of quality on OCR performance within the FR images is discussed in Reference [9]. The process of conducting OCR is discussed further in Section 2.2.2.

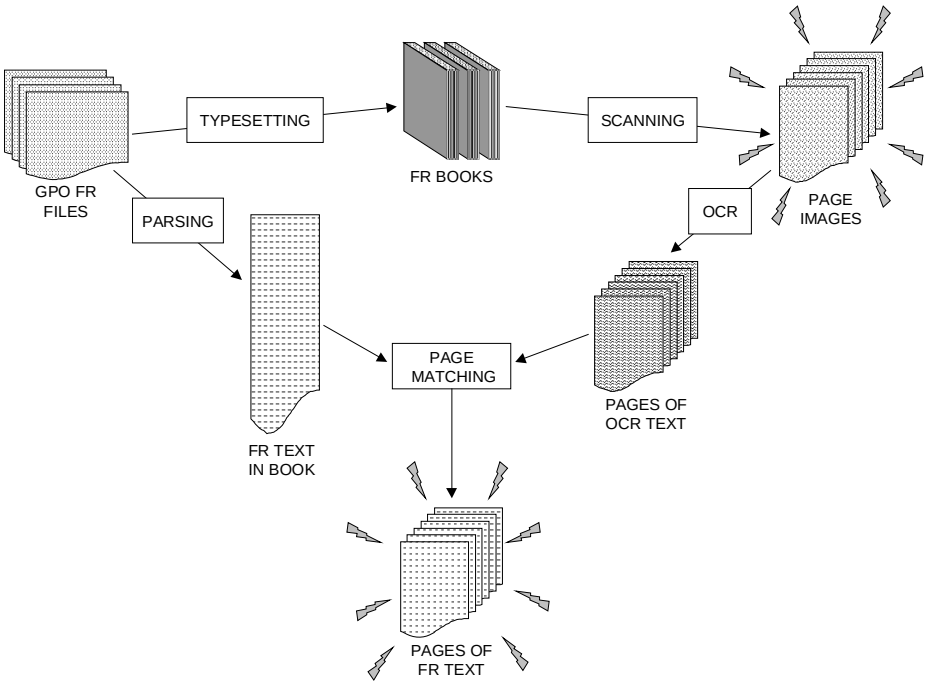

Figure 1. Overview of how the document image database was created.

On the left side of Figure 1, we have the GPO files that contain the true text printed in the books. While we have an entire book's text, we don't know what text was printed on which page; moreover, the GPO files are riddled with binary typesetting codes (a format named Microcomp). These codes represent physical markup for the corresponding book. A parser was written at NIST to strip the typesetting codes from the actual text. Then, based on the physical markup seen, logical structures were inferred and recorded. The parsing of the GPO files is discussed further in Section 2.2.3.

Multiple GPO files were used to typeset each FR book, and the ordering of the files and the order of the text within the files were not always clear, which compounded the challenge of locating the text associated with any specific page. As a result, the text parsed form each GPO file was simply concatenated together to represent the text for an entire FR book. This ordering was based on the alphabetical order of the GPO file names.

We now have in the middle of Figure 1 a book's text concatenated together in an arbitrary order on the left, and pages of OCR text on the right. A process was engineered at NIST to match a page's noisy OCR text against the large list of true text for the book and extract ground truth text for the page. This fully automated process of extracting ground truth is discussed further in Section 2.2.4.

This section has provided an overall description of how the FR document image database was created. Details of each step are provided in subsequent sections. Recipients of this database receive all the tools developed to prepare this data with the hope that they may prove useful in similar data collection efforts.

2.2.1 Scanning and Validating the *Federal Register*

As stated above, the scanning of the 1994 FR books was contracted to a service bureau. While the per-page scanning rate was very reasonable, significant additional cost was invested by NIST to verify the quality and content of the images. Pages of the FR were scanned by the service bureau using a high-speed Kodak 923 scanner[1] at a resolution of 15.75 ppm (400 ppi). The images were scanned as binary and stored in compressed Tagged Image File Format (TIFF[10]).

Each FR book had its binding cut, and its pages were placed in a hopper, feeding a high-speed paper transport. The pages were scanned with a sequential index assigned to each resulting image file. Upon scanning, the sequentially assigned index file names were cross-referenced automatically to ranges of physical pages in the book. This method of batch scanning is efficient, but prone to errors. For example, a significant number of blank pages were detected and their image files removed. In addition, a series of image quality checks was performed to detect truncated or otherwise corrupted bitmaps. Images that did not conform to the following criteria were rescanned using a Fujitsu 3096G scanner[1].

1. Resolution = 15.75 ppm
2. Compressed image file size >= 30 Kbytes
3. Width < 4000 pixels
4. 4200 pixels < Height <= 4900 pixels
5. Rotational skew < 5°

These tolerances were selected in order to minimize the amount of rescanning. Nonetheless, nearly 3% of the pages required rescanning. Apparently one shift operator scanned all his images at the wrong resolution, which accounted for 2% of the total pages rescanned.

Next to having the wrong image resolution, the most common error occurred when pages were scanned out of order, or two pages stuck together so that only one image was captured. OCR was used to catch out-of-order or missing pages. NIST utilized technology from its public domain OCR system [11] to create a machine print recognition application that located the page number on the top of FR pages. A subimage containing the page number was then segmented and classified. If the OCR result matched its hypothesized page number, then the image was determined to contain valid contents. If the OCR results did not match, then the image had to be reviewed by a human. Using this approach, 83% of the images were automatically verified without human adjudication [5].

Upon completion of this step, all FR page images were verified to be of a reasonable dimension, containing a properly encoded bitmap, and containing its proper page.

2.2.2 Running OCR on the FR Page Images

Once validated, each FR page image was processed using Xerox's ScanWorX API 3.1[1]. NIST developed a ScanWorX application program to process the FR pages, the source code for which is provided in this distribution. ScanWorX successfully completed execution on nearly all the FR images, providing results in XDOC format [12]. An XDOC file contains much more information than just the recognized text. Section 3.2.2.2 discusses the contents of the XDOC files in greater detail.

The majority of pages in a FR book (Figure 26 for example) contain a running header that identifies the book's volume, date, section, and page number. The GPO typesetting files do not contain page-level boundaries and information, so these headers are specified once at the beginning of major sections. As a result, the page images have headers whereas the parsed GPO text being matched against does not. To facilitate accurate matching, the OCR application locates the header and strips it off, so that its text is not included in the XDOC files. This also implies that header information is not provided in the database ground truth text files (a caveat introduced by the nature of the GPO files).

2.2.3 Parsing the GPO Typesetting Files

The files used by the GPO to print the FR are in Microcomp format. The GPO provided NIST with a collection of code tables. These binary codes are physical markup used by the typesetting system to choose fonts, format lines, construct tables, etc. The Microcomp documentation we received was incomplete, so significant effort was invested at NIST to reverse engineer a functional parser. Approximately one person year of labor was spent on this task. The resulting GPO parser (written in Perl) is provided in this distribution.

The GPO parser does more than strip the Microcomp codes from the text in the typesetting files. Additionally, it tracks the sequence of codes and infers logical context and structures from the physical markup. As a result, the parser is capable of producing SGML text tagged with sections, tables, footnotes, etc. Currently, the parser infers a rather limited number of logical structures defined in a provided DTD file (Section 3.2.1.2). If one desires, the parser may be augmented (with significant effort) to identify a broader set of logical entities.

Once all the GPO files for a FR book have been parsed, the resulting tagged text is concatenated together into one long list. At this point in the process, we have the textual content of an entire FR book in an uncertain order on one hand, and we have OCR text from each scanned page on the other. The next step is to map the OCR page text onto the GPO book text, extracting pages of text as ground truth.

2.2.4 Extracting Ground Truth by Matching OCR and GPO Text

The first sequence of steps involved in this stage of processing is illustrated in Figure 2. The concatenated text from the parsed GPO files (representing an entire book) is depicted on the left and the OCR page text is on the right. Both texts are treated as separate vectors of words. The book word vector is considerably longer than the page word vector. A FR book may contain 250,000 words whereas a page may contain 2000.

The immediate neighbors of each word are grouped into triples called trigrams. The locations of all the unique trigrams within the book's word vector are recorded. All the trigrams are computed from the OCR page word vector, and then matched against the unique trigrams from the book. Matching trigrams form a cluster, and the median of the cluster is used to anchor the page of OCR text to an area within the GPO text. The median of the cluster is called a "hit", and the area of GPO text matched is assumed to belong to the page from which the OCR was produced.

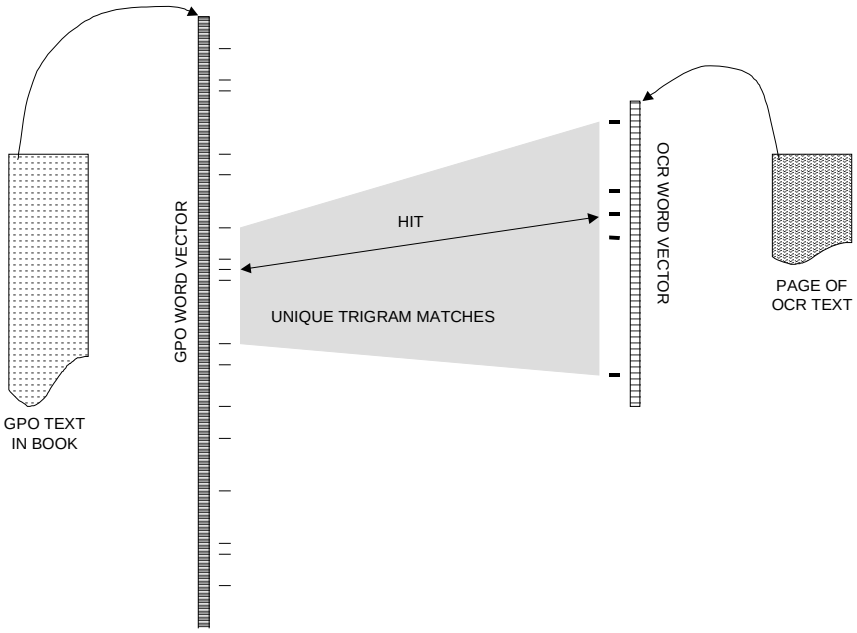

Figure 2. Matching the OCR page text to the GPO book text.

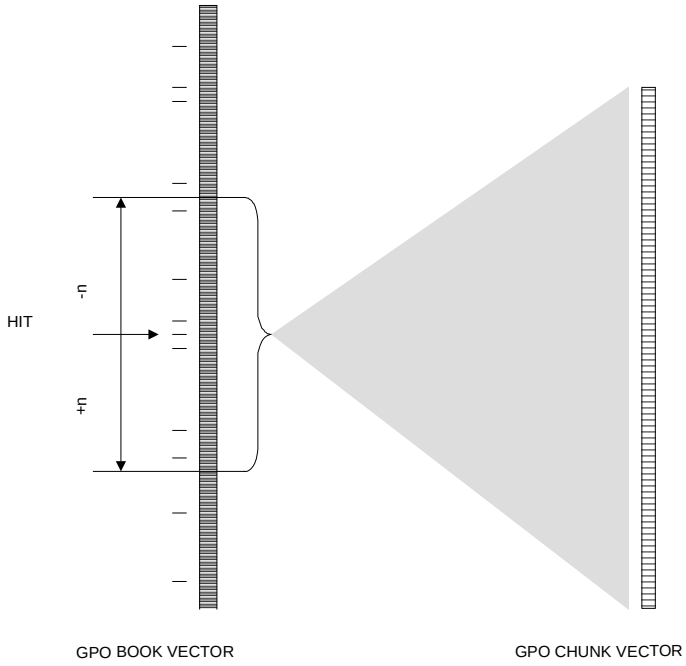

Figure 3. Extracting a chunk of GPO text.

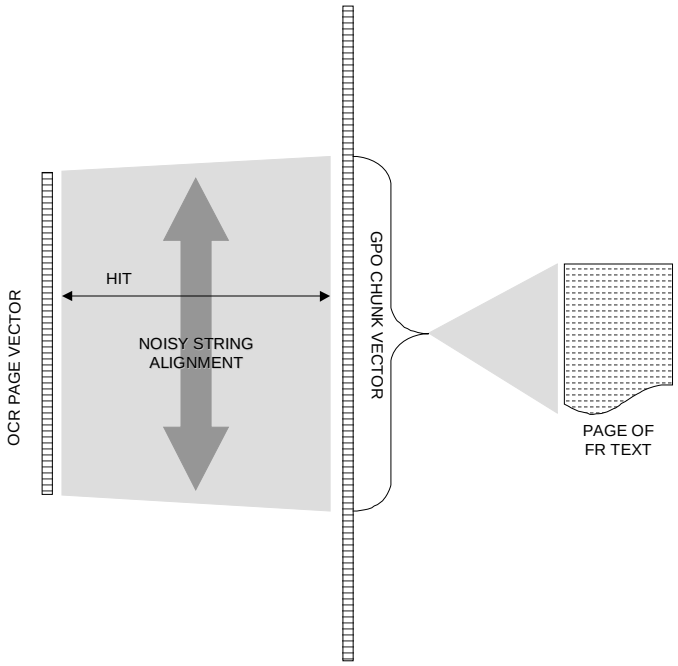

Figure 4. Trimming the chunk of GPO text.

Once the area of the page (or hit) has been located, the beginning and ending of the page must be determined from within the GPO text. String alignment is used to map the noisy OCR page text to the area surrounding the hit point within the GPO book vector. The FR pages vary in terms of the number of words on a page. Some pages with small fonts and dense text have more than 2000 words on a page. To localize the matching, a "chunk" of the GPO text is extracted, centered on the hit point. The radius of the chunk is conservatively set at (n=1500) words. Figure 3 illustrates extracting the chunk of GPO text from the book's word vector.

The chunk is assumed to contain the words for the desired page, but the chunk needs to be trimmed to correspond to the beginning and ending of the page. To narrow the chunk to the page's boundaries, string alignment technology developed at NIST was used to match the OCR page text to the GPO chunk. This is illustrated in Figure 4. String alignment was performed starting at the hit point in the chunk vector, working backward to the front of the chunk and forward to the end of the chunk. Working backward, the point at which the alignment sufficiently degraded was determined to be the beginning of the page, and working forward, the point at which the alignment sufficiently degraded was determined to be the end of the page. This string alignment technology is included in the distribution.

The detection of page boundaries is very difficult because the quality of the OCR text may be very poor at the beginning and end of the page and the OCR order of words may be different from the GPO typesetting order of words in these areas. Nonetheless, heuristics were developed based on the quality of alignment to hypothesize where page boundaries occur. The C program used to make these decisions is provided in the distribution. The reader is referred to the source code for documentation of these heuristics.

Based on the determined page boundaries, the GPO chunk of text was trimmed, and the remaining text was stored as the page's ground truth. An internal representation called a "word index file" was developed to support the steps described in this section. This representation (described in Section 3.2.1.3) enables SGML text files to be represented as word vectors while preserving the tagged context of the text. Using this representation, the trimmed GPO text chunk was reformatted into an SGML ground truth text file.

2.2.5 Uncertainty Issues and Quality Control

In the previous section, it was mentioned that not only does the performance of OCR impact the quality of the extracted ground truth, but that there were also order issues. The text contained in the GPO Microcomp files is in an order prescribed for the typesetting process, whereas the order derived by the OCR may be quite different. This difference occurs frequently in tables. The order that the cells are populated by the typesetter is typically row-oriented, whereas ScanWorX typically captures text in a column-oriented fashion. Differences in order also occur due to footnotes. OCR will locate footnotes typically at the bottom of a column of text, whereas there is no notion of bottom of page within the GPO files themselves. As a result, a footnote's text occurs sometime after the footnote's reference is encountered. But the footnote text may occur within a few sentences or a few paragraphs of its reference. To compound these ordering issues further, the OCR will at times incorrectly decolumnate a page, merging paragraphs across columns and vertically splitting paragraphs within a single column. This causes uncertainty in the OCR-derived order of the words on the page.

The good news is that the OCR order and GPO typesetting order generally follow the same reasonable reading order. Unfortunately, the order of the words at times may be quite different between the two sources. This, along with the potential of dismal OCR performance on low-quality images, introduces uncertainty in the derived ground truth text.

Quality control steps were developed to help reduce and manage this uncertainty. The process of clustering matched unique trigrams shown in Figure 2 may fail to produce a hit. This could happen, for example, if the OCR performance was so poor as to corrupt all the relevant trigrams on the page. This condition is easily detected. More difficult is detecting when an incorrect hit is generated. A hit may be bad due to 1) the ordering issues already discussed, 2) ambiguous word trigrams caused by OCR errors, or 3) text appearing on the FR page that is not explicitly contained in the GPO files.

The GPO Microcomp format uses external references to import data from external graphics files. These external references typically refer to items such as graphs, maps, mathematical formulae and some tables. When the OCR engine processes the contents of these pages, it recognizes text and symbols that are not present in the GPO parsed text for the book. This can throw the clustering of matching unique trigrams off, resulting in hits that do not point to their page's proper location.

An automated method of rejection was developed to detect incorrect hits. This process analyzes all the hits computed for all the pages in a book. Each hit references a location within the book's word vector as shown in Figure 4. FR pages on average contain 1000-2000 words each. Ideally, the hit values on contiguous pages will be evenly spaced at intervals of 1000-2000 words. The graph in Figure 5 plots all the hits calculated for each page in the FR book for January 3, 1994. Notice that the majority of points are members of linear segments. Those that do not follow the

linear trend of their neighbors are likely to be incorrect. These are automatically detected and flagged as bad, and no ground truth is extracted for these pages. It has been observed that the erratic points in the graph frequently correspond to pages comprised of large tables, which are prone to all the problem-causing issues described above.

The hit rejection algorithm is included in the distribution. There is a plot, similar to the one shown in Figure 5, provided for each FR book in the database. Inspection of these graphs gives the recipient of the database a visual representation of how well hits were detected within each FR book.

Ground truth files are provided for each accepted hit file. The same problem-causing issues of text order, OCR quality, and missing text also contributed directly to how accurately page boundaries could be determined. A method of rejecting ground truth was explored, but there was no way found that yielded reasonable results. Imposing such a mechanism would have required assumptions about the acceptable tolerances of experiments to be run on the data.

Rather than impose rigid and arbitrary procedures, a method was developed whereby the amount of error within a ground truth text file could be estimated. Giving a conservative estimate of error permits the researcher to determine what level of quality within the ground truth is required for a particular experiment. Once the ground truth text was extracted for a page, it was realigned with the OCR text using the NIST string alignment technology. A secondary step was invoked after initial alignment in order to match up blocks of text out of order between the ground truth (in typesetting order) and the OCR text.

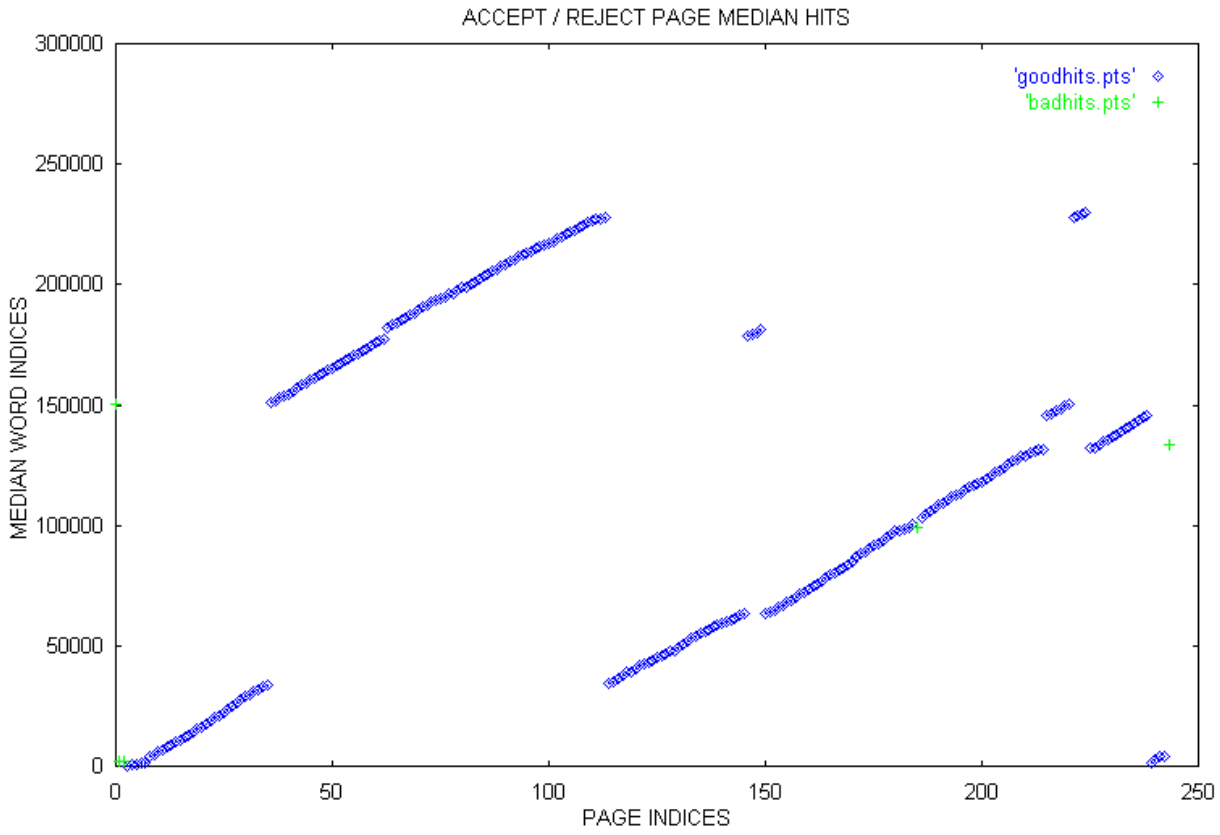

Figure 5. Plot of accepted/rejected hits for the FR book published January 3, 1994.

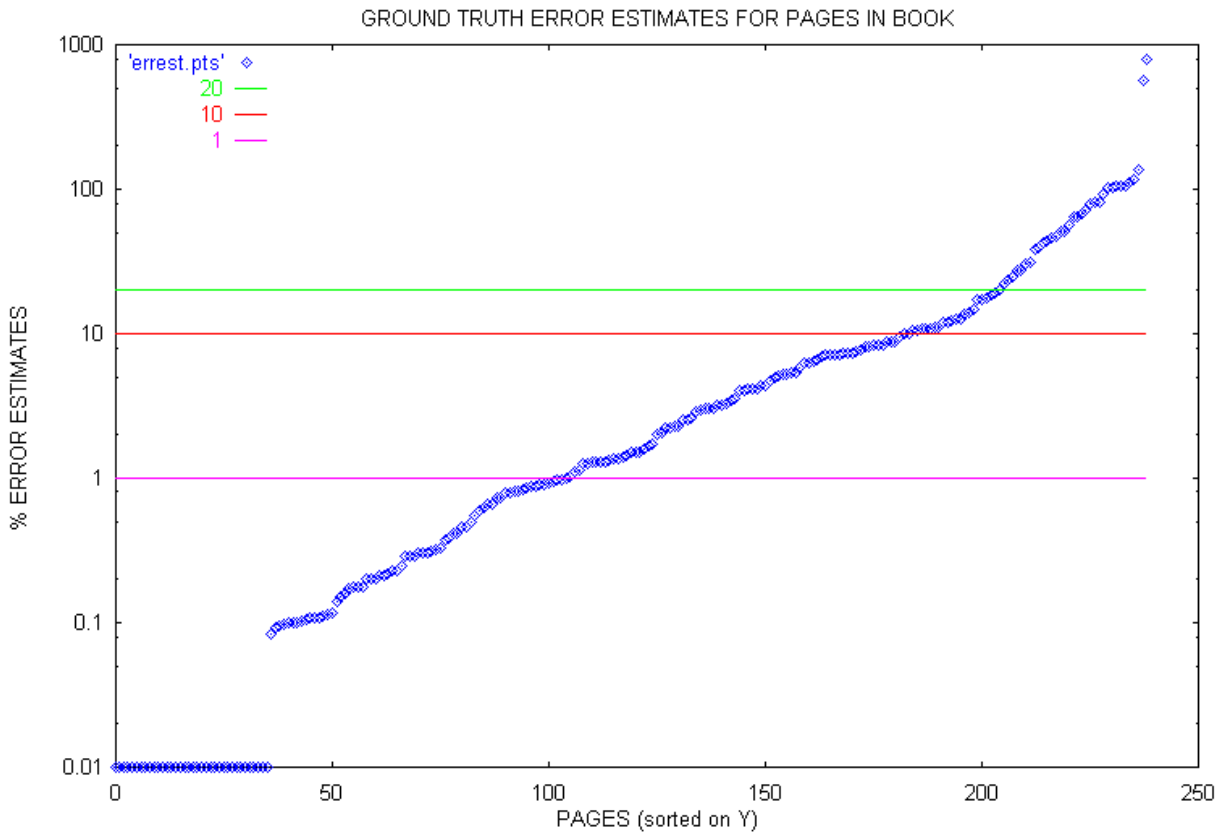

Figure 6. Plot of ground truth error estimates for FR pages on January 3, 1994.

After out-of-order blocks of text were matched, the remaining inserted and deleted words represent how much of the OCR text was not covered by the ground truth and how much of the ground truth was missing from the OCR text. The number of deleted and inserted words were used to compute a conservative error estimate as described in Section 3.2.2.9. An error estimate is provided with each ground truth text file in the database. These error estimates are plotted on a log scale in sorted y-order for each FR book in the database. An example of a plot for January 3, 1994 is shown in Figure 6. This graph provides the recipient of the database with a visual representation of the estimated quality of the ground truth for a given book.

3. DISTRIBUTION HIERARCHY AND CONTENT

This section describes the organization and contents of this database distribution. Figure 7 illustrates the top level directory structure in the distribution. The **src**, **include**, **bin**, and **lib** directories contain the source code and scripts used to produce the document image database as described in Section 2.2. The **man** directory contains on-line manual pages. A software reference guide is provided in Appendix A.

The **doc** directory contains PostScript formatted documentation (including this paper and other NIST reports referenced in this paper). All the files in **doc** have been GZIP compressed (see the compression discussion at the beginning of Section 3.2). The numbers used in these file names correspond to their reference number in this document. The file corresponding to this document is **db.ps**.

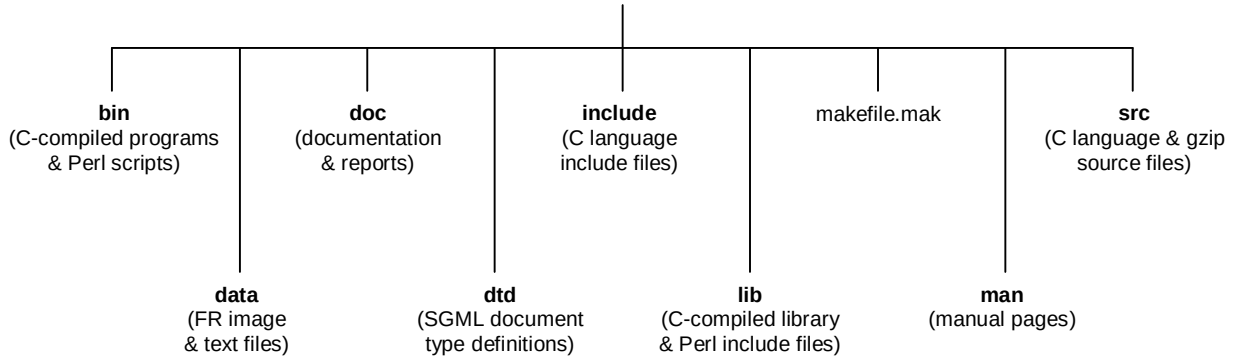

Figure 7. Top-level distribution hierarchy.

The **dtd** directory is a special directory containing two SGML Document Type Definition (DTD) files; one for the ground truth text files, and one for the SGML formatted OCR text files. Finally, the **data** directory holds all the database files for the distribution, including the FR page images and ground truth text.

3.1 Public Domain Software

All the software used to produce the data in this distribution has been provided. This is public domain code, and any portion of it may be used without restrictions because it was created with U.S. government funding.[2]

Three C programs have been provided. Their main routines are stored under the **src/bin** subdirectory, and they link subroutines archived in a C library, the source code for which is stored under the **src/lib** subdirectory. The C source code has been developed and tested to run on most common UNIX platforms, including LINUX, SunOS, Solaris, IRIX, AIX and HP-UX.[1] Compilation is controlled using the UNIX *make* facility through the use of a system of hierarchical makefile scripts. The makefile scripts are configured to use the GNU *gcc* compiler [13]. Porting the source code to other platforms and other compilers is the sole responsibility of the recipient.

To compile the C programs, you must mount the first of the two ISO-9660[8] CD-ROMs on your system. You may need the assistance of your system administrator. Once mounted, the directories **src**, **include**, **bin**, and **lib** must be copied completely to a read-writable disk partition. The directory into which the software is copied is referred to as the distribution's "installation directory" or *install_dir*. Once copied, all directories should be set to read, write, and executable and all files should be set to readable and writable (you may choose to keep the "**.c**" files read-only).

Variables are provided in the top-level makefile script (**makefile.mak**) that may be edited if you are using a different compiler than *gcc*. These variables allow you to define your specific compiler and the options it requires to have passed. It still may be necessary to edit the low-level makefile scripts to account for your specific compiler's options.

[2] This software was produced by NIST, an agency of the U.S. government, and by statute is not subject to copyright in the United States. Recipients of this software assume all responsibilities associated with its operation, modification, and maintenance.

The majority of utilities in this distribution are written in the portable scripting language Perl[14]. Some supporting Perl files are located under the top-level distribution **lib** directory, with all main scripts located under the top-level **bin** directory. One of these scripts, **catalog.pl**, automatically compiles C source code catalogs when the C library and programs are compiled. In order to access this script, the full path name to the top-level **bin** directory in *install_dir* must be added to your shell's environment path variable.

In order to execute these scripts, Perl must be installed on your computer [14]. The Perl scripts provided in this distribution are set up assuming that the Perl interpreter has been installed in **/usr/local/bin**. If Perl is not installed in this directory on your computer, then you will need to edit the top header line of each Perl script in the top-level **bin** directory. If, for example, the Perl interpreter is installed in **/apps/perl/bin**, then the header will need to be changed from:

```
#!/usr/local/bin/perl
```

to:

```
#!/apps/perl/bin/perl
```

To start compilation, it is recommended that you invoke the top-level makefile in your *install_dir*. You will likely need to edit the top-level makefile so that the variable PROJDIR is assigned to your *install_dir*. The makefile is set up by default with PROJDIR set to **/usr/local/fr94**.

Now you are ready for compilation. The first step is to direct the compiler to compute source code dependencies. If your compiler does not have this capability, then edit the low-level makefiles (in the program directories under **src/bin** and in the library **src/lib/db**), adding the general dependencies provided in the corresponding **depend.mak** files located in their same directories. Dependencies are created with *gcc* with the following command:

```
%  make -f makefile.mak depend
```

The next step is to compile and install the libraries and binaries. This is accomplished with the following command:

```
%  make -f makefile.mak install
```

Manual pages for all the C programs and Perl scripts are provided on-line in the **man** directory and are included in Appendix A.

3.2 Document Image Data

This section describes the organization and content of data files in the database. Each data file distributed under the top-level directory **data** has been compressed. Image files with extension **pct** are compressed using CCITT Group 4 [16]. All other files have been compressed using the public domain GNU zip (GZIP) utility [13]. The GZIP-1.2.4 source code distribution (requiring installation and compilation) is provide in the *tar* file **gzip.tar** under the top-level **src** directory.

Typically, files compressed using GZIP have an added extension of ".gz". This CD-ROM distribution is formatted in ISO-9660, requiring that all file names conform to the 8.3 naming convention. This means file names may be up to 8 characters long followed by a single extensions of up to 3 characters. This prohibits the use of a second extension on compressed data files. Therefore, all data files (other than image files) are GZIP compressed although their file names do not indicate it with an extra extension.

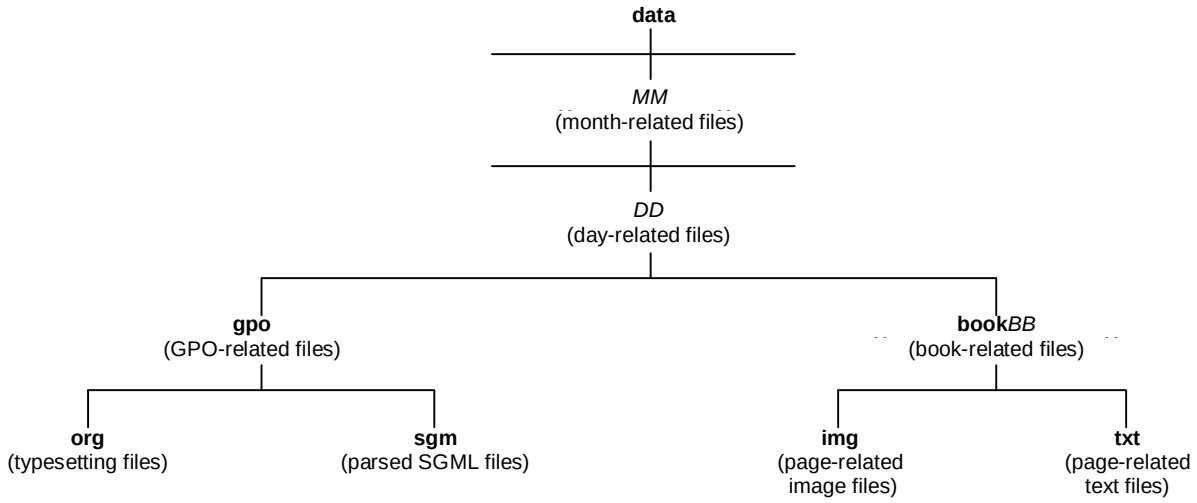

Figure 8. Data distribution hierarchy.

For example, the file **data/01/03/book01/txt/b0000100.xdc** is an XDOC file for page 100, published on January 3rd in the FR, and has been GZIP compressed. To access the file's contents, one can run the GZIP-provided utility *gzcat* with the command:

```
%  gzcat data/01/03/book01/txt/b0000100.xdc
```

The resulting decompressed data can be stored to a new file by redirecting the standard output of the previous command to a new file on a read-writable disk partition on your computer system.

Figure 8 illustrates the hierarchy in which the data files are distributed. Starting at the top level, the data is organized by month (***MM***). The files in this distribution are for the month of January (01). The data is sorted by day (***DD***) within each month. The FR is published each working day of the year, and there are 20 days worth of data provided for the month of January, 1994.

The overall size of the distribution is 1.27 gigabytes, requiring 2 CD-ROMs. The data files have been split by day over the two discs. The first CD-ROM contains the entire top-level hierarchy illustrated in Figure 7 and part of the data hierarchy illustrated in Figure 9, including day **01** through day **13** (excluding day **05**, which is included on the second disc due to space limitations). The second CD-ROM contains day **05** and the remaining data files for days **14** through **31**, with only a single top-level directory **data**. The first disc holds about 614 megabytes of data files, whereas the second disc holds 649 megabytes of data files. The data partitions are designed to be merged by simply copying the contents of their CD-ROM's top-level **data** directory into the same read-writable directory.

Within each day, the data files are divided into two parts. The first directory, **gpo**, contains all the FR files obtained from GPO and derivatives of these files. These files contain the true text for all the FR books published for a particular day. The original Microcomp typesetting files are stored in a subdirectory, **gpo/org**, with the subdirectory **gpo/sgm** containing corresponding parsed SGML files.

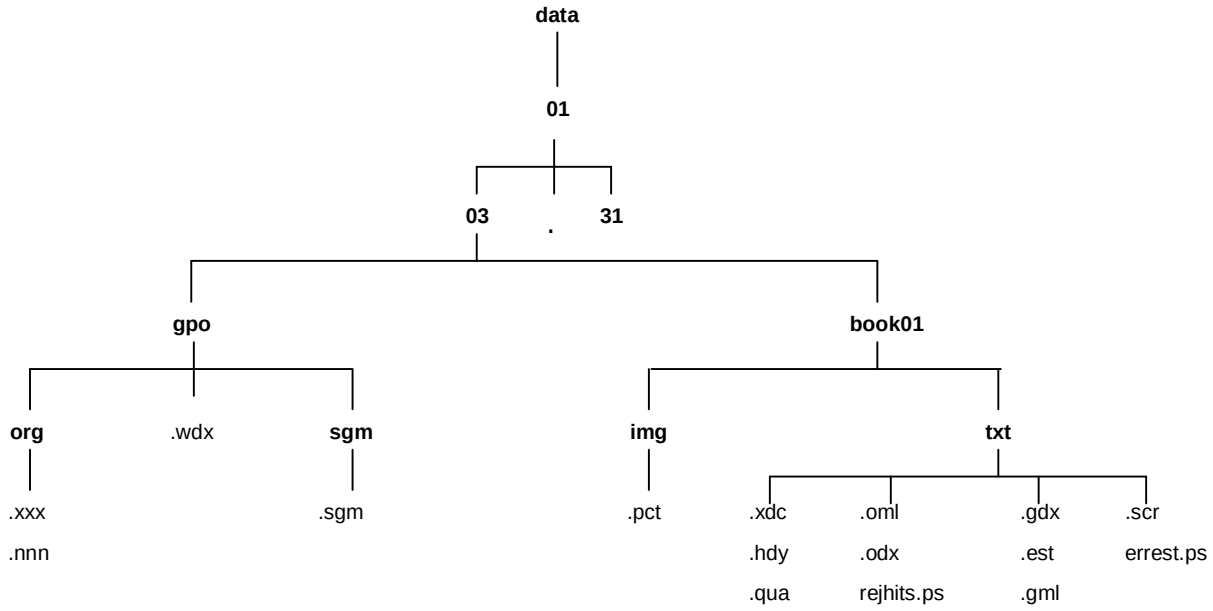

Figure 9. Location of data files.

EXT	ROOTNAME*	DESCRIPTION
xxx nnn	*TDDmmttt*	Original GPO Microcomp typesetting files
sgm	*TDDmmttt*	SGML files parsed from Microcomp files
wdx	*MMDD*	Master word index for the day
pct	*t9999999*	IHead page image
xdc	*t9999999*	Xerox XDOC file containing OCR results
hdy	*MMDD*	Vertical scanline at which page headers were detected and cut
qua	*MMDD*	Los Alamos image quality analysis and OCR error rate predictions
oml	*t9999999*	OCR SGML file parsed from corresponding XDOC file
odx	*t9999999*	OCR word index file parsed from corresponding OML file
ps	rejhits	Quality assurance plot of accepted and rejected trigram hits
gdx	*t9999999*	Ground truth word index file extracted from master word index
est	*MMDD*	Estimates of error in corresponding GDX files
gml	*t9999999*	Ground truth SGML file created from corresponding GDX file
scr	*t9999999*	Word-level score between corresponding GDX and ODX files
ps	errest	Quality assurance plot of sorted EST values for the entire book

* *TDDmmttt* — *T* - alphabetic file type; *DD* - numeric day; *mm* - alphabetic month; *ttt* - alphanumeric identifier
MMDD — *MM* - numeric month; *DD* - numeric day
t9999999 — *t* - alphabetic file type [a,b,c]; *9999999* - numeric page number

Table 1. Distribution data files.

The remaining directories, **book*BB***, contain files associated with each book published for a day. The **gpo** directory contains files for all the books, whereas the **book*BB*** directories contain files for each page in a book. There was only one FR book published for each day in January 1994, so each day (***DD***) directory contains only one directory (**book01**). (If a day were to have two books published, it would have two book directories present, **book01** and **book02**.)

The data files within a **book*BB*** directory are divided into two subdirectories. The first subdirectory, **book*BB*/img**, contains the page images for the book, whereas the subdirectory, **book*BB*/txt**, contains all the text-based files related to each page in the book (including the ground truth text files). All the text-based data files in this distribution are encoded using the ISO 8859-1[15] character set, which represent Western European characters. (Actually, many special characters are represented as SGML entities that map to the ISO 8859-1 character set. These entities are defined in the files located in **lib/sgml**.)

Table 1 lists all the various types of data files included in this database. File types are primarily identified by their extension, which are listed in the first column. The second column lists associated root file names followed by a short description. The location of these files in the data distribution hierarchy is illustrated in Figure 9.

A more detailed description of these files is provided below in the order prescribed in the table, starting first with the GPO-related files, and then continuing with the book-related page files. To help explain these files, examples have been taken from a specific FR page in this distribution. This page, shown in Figure 26, is body page 100 published on January 3, 1994.

3.2.1 GPO-Related Data Files

These files include GPO-provided Microcomp files, parsed SMGL files, and a master word index file. The GPO used the Microcomp files to typeset and print each FR book. The NIST SGML files contain FR text as well as a set of logical formatting tags. The master word index file provides cross-referencing for ground truth extraction.

3.2.1.1 GPO Microcomp Files

Each daily issue of the FR is represented by a set of Microcomp files. These files, used to typeset the FR, consist of ASCII text with binary encoded markup and typesetting instructions. The ordering within a Microcomp file does not always match the physical ordering of pages in a FR book. Additionally, a Microcomp file does not contain any information that denotes page boundaries on which the text is printed. Therefore, it is a difficult task to extract the exact content of a printed page from a Microcomp file. Nonetheless, an automated process (described in Section 2.2) was created to extract an hypothesized page-level ground truth.

All the GPO Microcomp files for a particular day are stored in the subdirectory **gpo/org** and have either the extension **xxx** or **nnn**. In our example, the text on FR page 100 is located in the Microcomp file **data/01/03/gpo/org/r03ja3.xxx**. A portion of this file is not illustrated here due to its binary encoding.

3.2.1.2 GPO SGML Files

A parser was developed at NIST that converts Microcomp files into text-based SGML files. The parser filters the physical markup codes and infers certain logical structures such as paragraphs of text, figures, tables, and footnotes. These logical structures are then represented by SGML

tags. The complete set of tags detected and stored is documented in the SGML DTD file **dtd/fr94gpo. dtd.** Table 2 lists those tags that are commonly detected. The middle column lists the total instances that a structure was detected in the FR for the entire month of January 1994. The last column provides a brief description of each tag. As can be seen, footnotes, dates, and tables occur quite frequently.

Figure 10 plots the frequency of each tag accumulated by day. The total in each category equals the value listed in Table 2. Each daily issue contributes to these categories in a relatively proportionate way. The majority of text is labeled with a generic tag, TEXT, whose total magnitude is off the chart.

TAG	TOTAL	DESCRIPTION
DOCNO	160	GPO SGML file base name
FN	2379	footnote body including identifier and text
FNID	2218	footnote identifier (ex. number used in reference)
FNTEXT	2378	footnote text
FR	2551	footnote reference (ex. superscripted numbers)
NOTICE	2063	major notice item submitted by agency
NOTICE-ACTION	1252	action field in notice
NOTICE-AGENCY	1218	agency field in notice
NOTICE-CONTACT	1277	contact information for notice
NOTICE-DATE	2431	date of notice
NOTICE-LOCATION	140	location of notice
NOTICE-MATTERS	48	matters field in notice
NOTICE-STATUS	80	status of notice
NOTICE-SUMMARY	1216	summary field in notice
NOTICE-TITLE	108	title of notice
TABLE	851	table
TEXT	226055	character sequence of unknown (generic) type

Table 2. Tags included in GPO SGML files.

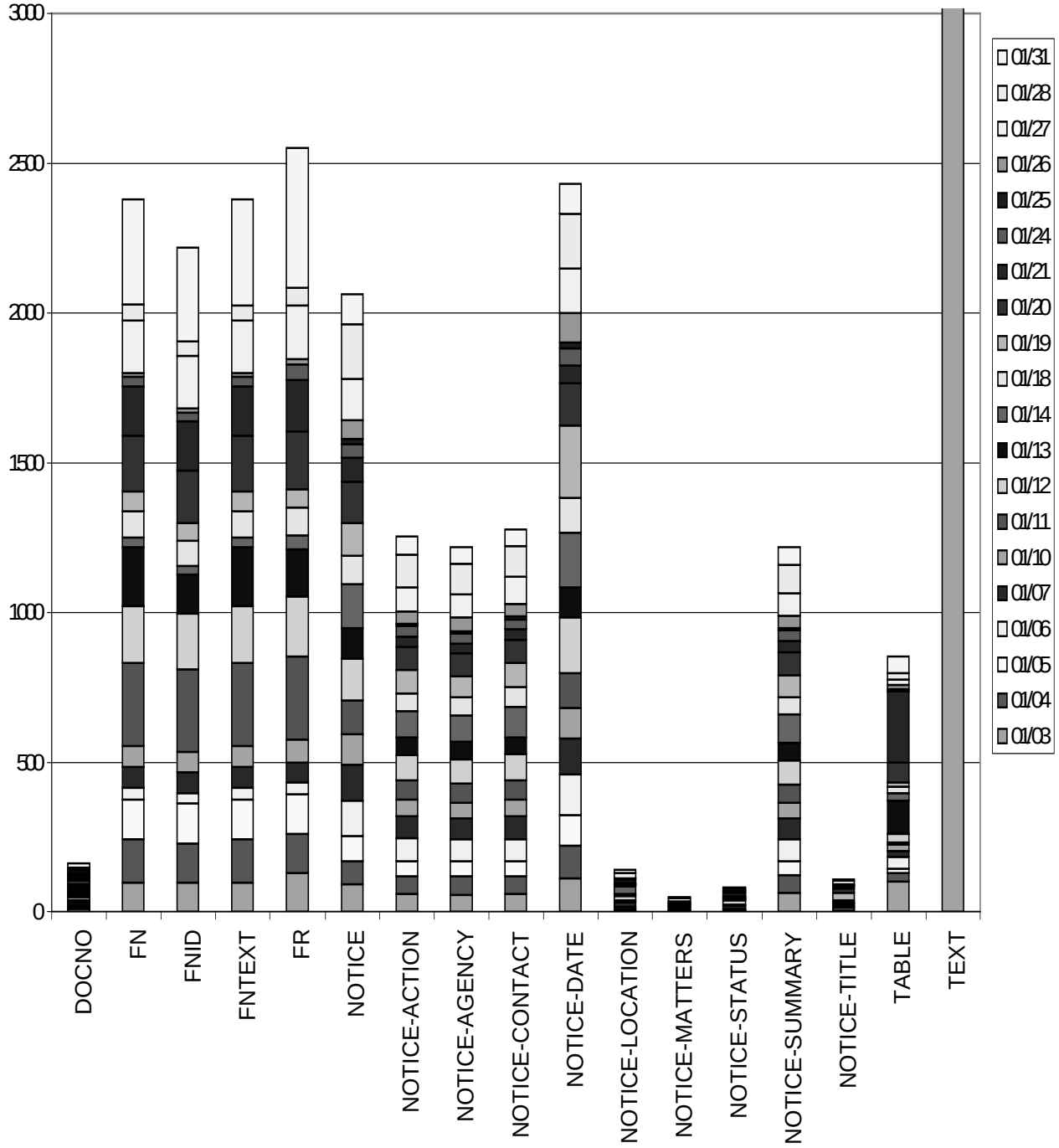

Figure 10. Frequency distribution of tags (accumulated by day) for January, 1994.

```
. . .
<TEXT>
Further information with reference to this meeting can be obtained from Ms. Yvonne Sabine,
Committee Management Officer, National Endowment for the Arts, Washington, DC 20506, or call
202/682&hyphen;5439.
</TEXT>
<NOTICE-DATE>
Dated: December 27, 1993.
</NOTICE-DATE>
<TEXT>
Yvonne M. Sabine,
</TEXT>
<TEXT>
Director, Office of Panel Operation, National Endowment for the Arts.
</TEXT>
<TEXT>
[FR Doc. 93&hyphen;32050 Filed 12&hyphen;30&hyphen;93; 8:45 am]
</TEXT>
<TEXT>
BILLING CODE 7537&hyphen;01&hyphen;M
</TEXT>
</NOTICE>
<NOTICE>
<TEXT>
NATIONAL LABOR RELATIONS BOARD
</TEXT>
<TEXT>
Appointments of Individuals to Serve as Members of Performance Review Boards
</TEXT>
<TEXT>
5 U.S.C. 4314(c)(4) requires that the appointments of individuals to serve as members of
performance review boards be published in the Federal Register. Therefore, in compliance
with this requirement, notice is hereby given that the individuals whose names and position
titles appear below have been appointed to serve as members of performance review boards in
the National Labor Relations Board for the rating year beginning October 1, 1992 and ending
September 30, 1993.
</TEXT>
. . .
```

Figure 11. Portion of a GPO SGML file.

All the GPO SGML files for a particular day are stored in the subdirectory **gpo/sgm** and have the extension **sgm**. There is one SGML file for each Microcomp file, and they share the same root file name. Figure 11 lists a portion of the GPO SGML for body page 100. This text corresponds mostly to the top-middle column on the page. Note that there are no newline characters contained on the end of the lines that wrap in the figure. This data is stored in the file **data/01/03/gpo/sgm/r03ja3.sgm**.

3.2.1.3 Master Word Index Files

A GPO master word index file contains, in typesetting order, a record for each word that was printed in an issue of the FR. The GPO SGML files are concatenated together based on the alphabetic order of their file names, and the concatenated text is converted to a word-based representation. This representation is critical to the ground truth extraction process described in Section 2.2.

```
 . . .
221597  39838   Dated:          r03ja3.sgm      [DOC:8] [NOTICE:80] [NOTICE-DATE:105]
221598  39839   December        r03ja3.sgm      [DOC:8] [NOTICE:80] [NOTICE-DATE:105]
221599  39840   27,             r03ja3.sgm      [DOC:8] [NOTICE:80] [NOTICE-DATE:105]
221600  39841   1993.           r03ja3.sgm      [DOC:8] [NOTICE:80] [NOTICE-DATE:105]
221601  39842   Yvonne          r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13745]
221602  39843   M.              r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13745]
221603  39844   Sabine,         r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13745]
221604  39845   Director,       r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221605  39846   Office          r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221606  39847   of              r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221607  39848   Panel           r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221608  39849   Operation,      r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221609  39850   National        r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221610  39851   Endowment       r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221611  39852   for             r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221612  39853   the             r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221613  39854   Arts.           r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13746]
221614  39855   [FR             r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13747]
221615  39856   Doc.            r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13747]
221616  39857   93&hyphen;32050 r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13747]
221617  39858   Filed           r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13747]
221618  39859   12&hyphen;30&hyphen;93; r03ja3.sgm [DOC:8] [NOTICE:80] [TEXT:13747]
221619  39860   8:45            r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13747]
221620  39861   am]             r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13747]
221621  39862   BILLING         r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13748]
221622  39863   CODE            r03ja3.sgm      [DOC:8] [NOTICE:80] [TEXT:13748]
221623  39864   7537&hyphen;01&hyphen;M r03ja3.sgm [DOC:8] [NOTICE:80] [TEXT:13748]
221624  39865   NATIONAL        r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13749]
221625  39866   LABOR           r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13749]
221626  39867   RELATIONS       r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13749]
221627  39868   BOARD           r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13749]
221628  39869   Appointments    r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221629  39870   of              r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221630  39871   Individuals     r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221631  39872   to              r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221632  39873   Serve           r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221633  39874   as              r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221634  39875   Members         r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221635  39876   of              r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221636  39877   Performance     r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221637  39878   Review          r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
221638  39879   Boards          r03ja3.sgm      [DOC:8] [NOTICE:81] [TEXT:13750]
 . . .
```

Figure 12. Portion of a master word index file.

A portion of the master word index for January 3, 1994 is listed in Figure 12. Each row in the index corresponds to a successive word from the GPO SGML files.

A row is comprised of the following:

1. global word index
2. local word index
3. word
4. SGML file
5. variable length list of all open logical document tags

The global word index is a unique index incremented for each word across all the GPO SGML files for a given day. The local word index is reset at the beginning of each GPO SGML file and incremented for each word in the file. The third field is the word string, and the fourth field is the SGML file from which the word was extracted. The remaining variable number of fields

store the SGML context of the word. In other words, all the SGML tags that are currently open when the word was encountered are recorded here.

There is one master word index file per day, and it is stored directly within the **gpo** subdirectory. A master word index file has the extension **wdx** and a root name formed from the concatenation of the numeric month and day strings (***MMDD***). For example the data listed in Figure 12 is stored in the file **data/01/03/gpo/0103.wdx**.

The words listed in this example correspond to the top-middle column of body page 100 (shown in Figure 26). The list starts with the line "Dated: December 27, 1993." and ends within the start of a new notice section for the National Labor Relations Board. Note the words on the date line are tagged with NOTICE-DATE followed by the number 105. This number is an instance of that particular tag type occurring in the current daily issue. The instance indices are zero-oriented, so that this date belongs to the 81st notice section in the issue. (Locate the preceding tag "NOTICE: 80" for the words on the date line.)

Now locate the sequence of words: "NATIONAL LABOR RELATIONS BOARD." This starts a new notice section which is represented by the change of the instance index on the NOTICE tag for the word "NATIONAL." The instance increments from 80 to 81 denoting the start of a new section.

This word index format allows a passage of SGML tagged text to be represented as a word list, preserving the logical context of every word. This format is the same used for the page-related OCR and ground truth word index files. (The ground truth word index files are actually extracted sequences from the master word index file.)

3.2.2 Book-Related Data Files

This section describes the data files contained within the distribution's **book*BB*** directories. All the daily issues of the FR in January 1994 were printed as single books, so all day (***DD***) directories contain a single directory (**book01**). Two classes of page-related files are contained within each **book*BB*** directory: page image files and page-related text files. The text files may be further distinguished as being either OCR results or derived ground truth files. The bulk of this database contains these page-related files.

Many different experiments may be performed on these files. For example, the images may be input to other OCR systems to evaluate and benchmark accuracy using the existing ground truth files and the provided scoring package. In this way, differences between OCR systems may be compared and understood. OCR results are also provided for every image file in the distribution and contain font detection, word segmentations, and character classifications. Thus, a large collection of commercial OCR results is provided that can be studied and modeled for various applications. Since the images vary in image quality, experiments can be setup to measure the effect of image quality on OCR errors and, subsequently, OCR errors on retrieval performance.

3.2.2.1 Page Image Files

Before scanning each FR book, its pages were cut from its bookbinding. This resulted in paper pages that were approximately 20 cm (8") by 28 cm (11") in size. Image scanning was performed using a high speed Kodak 923 scanner[1]. For the month of January 1994, 4771 pages were scanned. The pages were scanned binary at 15.75 ppm (400 ppi) and converted to 2-dimensionally compressed IHead image files (described in Appendix B).

IHead is a public domain, image file interchange format that has been designed with an extensive set of attributes and can be used to represent both binary and gray level images. It has been implemented across heterogeneous computer architectures and environments. Software is provided in the CD-ROM distribution that allows applications to read, decompress, and write IHead images.

All FR page image files for a given book are stored under the subdirectory **book01/img** with extension **pct**. They have a root file name format *t9999999*, where *t* represents either '**a**', '**b**', or '**c**', corresponding to prefix pages, body pages, or appendix pages respectively. The characters *9999999* correspond to the page number within the FR book. The majority of page-related files use this same root file name format. For example, the file containing the image for body page 100 (shown in Figure 26) is stored as **data/01/03/book01/img/b0000100.pct**.

3.2.2.2 OCR XDOC Files

OCR results recorded in the database were generated using Xerox's ScanWorX API 3.1[1]. Each page of OCR results is stored in a text-based XDOC format [12]. A portion of an XDOC file is shown in Figure 13. Although this format is text-based, it is not easily deciphered by a human. Some of the types of information contained in an XDOC file include:

1. global estimation of image skew
2. word bounding boxes
3. font identification
4. character classifications and associated confidences
5. word confidences
6. line formatting information

```
. . .
[s;5;777;0;707;p;8][w;806][v;33;42;44][b;773;684;932;709][c;8][q;858]N[q;937]A[q;928]T[q;891]
I[q;939]O[q;930]N[q;941]A[q;806]L[h;936;11][w;733][v;33][b;942;685;1049;709][q;733]L[q;959]A[
q;868]B[q;941]O[q;872]R[h;1053;11][w;742][v;33][b;1060;685;1236;710][q;869]R[q;868]E[q;742]L[
q;742]A[q;936]T[q;871]I[q;941]O[q;869]N[q;939]S[y;1759;520;707;1;S]
[s;5;777;1;741;p;8][w;310][v;33][b;774;718;885;742][q;422]B[q;363]O[q;310]A[q;366]R[q;374]D[y
;1759;871;741;1;H]
[s;5;777;0;802;p;7][w;800][v;33][b;772;779;979;808][c;7][q;930]A[q;854]p[q;930]p[q;891]o[q;87
3]i[q;892]n[q;931]t[q;918]m[q;891]e[q;901]n[q;936]t[q;800].[h;984;11][w;764][v;33][b;989;779;
1017;803][q;860]o[q;764]f[h;1022;11][w;396][v;33][b;1027;780;1186;805][q;896]I[q;851]n[q;915]
d[q;875]i[q;923]v[q;882]i[q;924]d[q;700]u[q;697]a[q;396]l[q;843]s[h;1191;10][w;865][v;33][b;1
196;782;1223;805][q;865]t[q;911]o[h;1228;11][w;785][v;33][b;1234;780;1316;805][q;937]S[q;922]
e[q;785]r[q;785]v[q;924]e[y;1759;438;802;1;S]
[s;5;777;0;835;p;7][w;843][v;33][b;772;818;805;836][q;882]a[q;843]s[h;810;11][w;1][v;32][b;81
6;812;949;839][q;445]M[q;407]e[q;724]m[E[q;399]s[h;954;11][w;861][v;33][b;960;813;987;836][q;
907]o[q;861]f[h;992;11][w;766][v;33][b;998;813;1184;838][q;935]P[q;934]e[q;766]r[q;766]f[q;89
8]o[q;809]r[q;806]m[q;920]a[q;840]n[q;889]c[q;931]e[h;1189;12][w;855][v;33][b;1196;814;1299;8
38][q;916]R[q;924]e[q;917]v[q;855]i[q;924]e[q;880]w[y;1759;455;835;1;S]
[s;5;777;1;869;p;6][w;310][v;33][b;773;846;877;870][c;6][q;310]B[q;388]o[q;407]a[q;310]r[q;38
8]d[q;451]s[y;1759;877;869;1;H]
. . .
```

Figure 13. Portion of an XDOC file.

This distribution includes a public domain XDOC parser written at NIST in Perl. The parser can be used to filter the different types of information from XDOC files. For example, it can separate the recognized text from non-text items and store the results as tagged SGML.

An XDOC file is provided for every image in the database (with a few exceptions caused by ScanWorX failures). All XDOC files for a given FR book are stored in a subdirectory **book01/txt** with extension **xdc**.). For example the data listed in Figure 13 is from body page 100 and is stored in the file **data/01/03/book01/txt/b0000100.xdc**. The results listed in the figure are for a few lines from the top-middle column of the page, starting with the line "NATIONAL LABOR RELATIONS BOARD."

3.2.2.3 Header Cut Files

With the exception of cover and section pages, each page of the FR is printed with a page heading that includes a text banner printed above two horizontal lines. The text banner line contains information that identifies the volume, date, topic, and page number. An example of a FR page header is seen in Figure 26. The GPO typesetting files do not contain page-level boundaries and information, so these headers are specified once at the beginning of major sections. As a result, the page images have headers whereas the parsed GPO text being matched against does not. To facilitate accurate matching, the OCR application locates the header and strips it off, so that its text is not included in the XDOC files.

The OCR application program locates the bottom of the heading's two horizontal lines and erases the header from the page image prior to recognition. The distance (in terms of pixels or scan lines) from the top of the image to the detected bottom of the header is recorded. If no header is detected, a value of "-1" is recorded.

The provided utilities detect the header location for each XDOC file in the database. Each of these locations is stored to a separate file. These small files have been concatenated together into a single compressed file for distribution. All the detected header locations for the pages of a given FR book are stored in a subdirectory **book01/txt** in a single file named ***MMDD*.hdy**. For example, the detected header location for body page 100 is recorded in the file **data/01/03/book01/txt/0103.hdy** as:

```
b0000100  380
```

In this case, the bottom of the header was detected 380 pixels from the top of the image and removed.

3.2.2.4 Image Quality Assessment Files

Software developed at Los Alamos Laboratories was used to assess the image quality of each FR page image in the database [4]. This software is designed to predict an OCR word error rate based on a set of computed image quality descriptors. These descriptors include measurement of:

1. *Small Speckle Factor*
   measures fine background speckle in the document image.
2. *Large Speckle Factor*
   measures larger chunks of background speckle.

3. *White Speckle Factor*
   measures the small white enclosed areas in characters. These shrink in size as characters bloat.
4. *New White Speckle Factor*
   same as above, but possibly more meaningful to the eye.
5. *Touching Character Factor*
   measures how much neighboring characters touch each other.
6. *Broken Character Factor*
   measures how broken the text characters are. This factor is also affected by small chunks of noise that may lie on a line of text.
7. *Font Size Factor*
   indicates the size of the font, normalized from one (small) to zero (large).

These image quality descriptors may be used as a selection criteria for designing experiments by obtaining different image sample profiles based on different levels of image quality. The results recorded for each FR page image contain a vector of 8 floating point numbers, representing in order the 7 image quality descriptors listed above, followed by a predicted OCR word error rate. The Los Alamos software was trained to predict the error of the Caere OmniPagePro recognition engine[1]. (The image quality assessment utility is the only software used to produce elements of the database that is not included in this distribution.)

The image quality results for each page of a given FR book are stored together in a single file named ***MMDD*.qua** in a subdirectory **book01/txt**. Each line in an image quality assessment file contains the results for a specific page. A line begins with the page's root file name followed by the computed image quality descriptors and a predicted OCR word error rate. For example, the results for body page 100 are recorded in the file **data/01/03/book01/txt/0103.qua** as:

```
b0000100 0.0311094 0.00691304 0.139588 0.21434 0.0826667 0.257034 0.433333 0.334115
```

In this case, the predicted word error rate is 33.4%.

3.2.2.5 OCR SGML Files

Results from parsing each XDOC file are stored within an SGML file that contains the recognized words and includes line break information. The files contain noisy text generated by the commercial OCR process and can be used in IR experiments. The OCR system reports results one isolated line of text at a time without any additional logical context. As a result, the OCR text is simply tagged at the line level. A DTD for these OCR SGML files is provided in **dtd/fr94ocr.dtd**.

An OCR SGML file is provided for every XDOC file in the database. All OCR SGML files for a given FR book are stored in a subdirectory **book01/txt** with extension **oml**. An example of a portion of one these files is listed in Figure 14. This figure lists the OCR results corresponding mostly to the top-middle column of body page 100. This data is stored as the file **data/01/03/book01/txt/b0000100.oml**.

Notice that the OCR system merged the first three lines of the right-most column with the first three lines of the middle column. This OCR text can be compared to the passage of GPO SGML text listed in Figure 11. Notice that the OCR SGML is tagged line by line, whereas the GPO SGML is tagged paragraph by paragraph.

```
. . .
<LINE>
Further information with reference to
</LINE>
<LINE>
this meeting can be obtained from Ms.
</LINE>
<LINE>
Yvonne Sabine, Conunittee Managemen:t NUCLEAR REGULATORY
</LINE>
<LINE>
Officer, National Endbwment for the COMMISSION
</LINE>
<LINE>
Arts, Was~iington, DC 20506, or call [Docket No. 5~261]
</LINE>
<LINE>
202/682&mdash;5439.
</LINE>
<LINE>
Dated: December 27, 1993.
</LINE>
<LINE>
Yvonne M Sabine,
</LINE>
<LINE>
Director, Office of Panel Operation, National
</LINE>
<LINE>
Endowment for the Arts.
</LINE>
<LINE>
[FR Doc. 93&mdash;32050 Filed 12&mdash;3(~93; 8:45 am]
</LINE>
<LINE>
BILUNG COO! 7337~1~
</LINE>
<LINE>
NATIONAL LABOR RELATIONS
</LINE>
<LINE>
BOARD
</LINE>
<LINE>
Appointment. of Individuals to Serve
</LINE>
<LINE>
as Mem~s of Performance Review
</LINE>
<LINE>
Boards
</LINE>
. . .
```

Figure 14. Portion of an OCR SGML file.

3.2.2.6 OCR Word Index Files

Each OCR SGML file is formatted into a word index file (the same format used to construct the master word index file in Section 3.2.1.3). This representation facilitates matching a page's OCR results to a book's text parsed from the GPO typesetting files. It also allows for the cross-referencing of words while maintaining each word's logical context. An OCR word index file contains an entry for each recognized word from a page image in an OCR-derived order. This order does not always match the GPO's typesetting order as seen in Figure 14.

```
. . .
365     365     Dated:          b0000100.oml    [DOC:0] [LINE:69]
366     366     December        b0000100.oml    [DOC:0] [LINE:69]
367     367     27,             b0000100.oml    [DOC:0] [LINE:69]
368     368     1993.           b0000100.oml    [DOC:0] [LINE:69]
369     369     Yvonne          b0000100.oml    [DOC:0] [LINE:70]
370     370     M               b0000100.oml    [DOC:0] [LINE:70]
371     371     Sabine,         b0000100.oml    [DOC:0] [LINE:70]
372     372     Director,       b0000100.oml    [DOC:0] [LINE:71]
373     373     Office          b0000100.oml    [DOC:0] [LINE:71]
374     374     of              b0000100.oml    [DOC:0] [LINE:71]
375     375     Panel           b0000100.oml    [DOC:0] [LINE:71]
376     376     Operation,      b0000100.oml    [DOC:0] [LINE:71]
377     377     National        b0000100.oml    [DOC:0] [LINE:71]
378     378     Endowment       b0000100.oml    [DOC:0] [LINE:72]
379     379     for             b0000100.oml    [DOC:0] [LINE:72]
380     380     the             b0000100.oml    [DOC:0] [LINE:72]
381     381     Arts.           b0000100.oml    [DOC:0] [LINE:72]
382     382     [FR             b0000100.oml    [DOC:0] [LINE:73]
383     383     Doc.            b0000100.oml    [DOC:0] [LINE:73]
384     384     93&mdash;32050  b0000100.oml    [DOC:0] [LINE:73]
385     385     Filed           b0000100.oml    [DOC:0] [LINE:73]
386     386     12&mdash;3{~93; b0000100.oml    [DOC:0] [LINE:73]
387     387     8:45            b0000100.oml    [DOC:0] [LINE:73]
388     388     am]             b0000100.oml    [DOC:0] [LINE:73]
389     389     BILUNG          b0000100.oml    [DOC:0] [LINE:74]
390     390     COO!            b0000100.oml    [DOC:0] [LINE:74]
391     391     7337~1~         b0000100.oml    [DOC:0] [LINE:74]
392     392     NATIONAL        b0000100.oml    [DOC:0] [LINE:75]
393     393     LABOR           b0000100.oml    [DOC:0] [LINE:75]
394     394     RELATIONS       b0000100.oml    [DOC:0] [LINE:75]
395     395     BOARD           b0000100.oml    [DOC:0] [LINE:76]
396     396     Appointment.    b0000100.oml    [DOC:0] [LINE:77]
397     397     of              b0000100.oml    [DOC:0] [LINE:77]
398     398     Individuals     b0000100.oml    [DOC:0] [LINE:77]
399     399     to              b0000100.oml    [DOC:0] [LINE:77]
400     400     Serve           b0000100.oml    [DOC:0] [LINE:77]
401     401     as              b0000100.oml    [DOC:0] [LINE:78]
402     402     Mem~s           b0000100.oml    [DOC:0] [LINE:78]
403     403     of              b0000100.oml    [DOC:0] [LINE:78]
404     404     Performance     b0000100.oml    [DOC:0] [LINE:78]
405     405     Review          b0000100.oml    [DOC:0] [LINE:78]
406     406     Boards          b0000100.oml    [DOC:0] [LINE:79]
. . .
```

Figure 15. Portion of an OCR word index file.

An OCR word index file is provided for every OCR SGML file in the database. All OCR word index files for a given FR book are stored in a subdirectory **book01/txt** with extension **odx**. Figure 15 lists a portion of the OCR word index file for body page 100. This data is stored in the file **data/01/03/book01/txt/b0000100.odx**.

3.2.2.7 Rejected Hits Plot

As part of the ground truth extraction process described in Section 2.2.4, a page's OCR text is matched against the GPO text for an entire FR book, pinpointing the page's location within the GPO text. This location is referred to as a *hit*. Due to OCR errors and anomalies in the GPO typesetting text, a computed page hit is not always accurate. An automated method of rejecting bad hits was developed, and the results after imposing this rejection criteria are plotted. An example of one of these plots is shown in Figure 5.

There is one rejected hits plot computed and stored for each FR book processed, and each point in the plot represents a page whose hit was either accepted or rejected. Each of these plot files

contain a PostScript encoded graph. There is one **rejhits.ps** file provided in each **book01/txt** subdirectory in the database. The plot in Figure 5 is stored in the file **data/01/03/book01/txt/rejhits.ps**.

3.2.2.8 Ground Truth Word Index Files

A ground truth word index file contains all words extracted from the GPO master word index determined to appear on a specific FR page image. A ground truth word index file is provided for every OCR XDOC file in the database (excluding those pages with rejected hits).

All ground truth word index files for a given FR book are stored in a subdirectory **book01/txt** with extension **gdx**. Figure 16 lists the beginning and ending of the ground truth file derived for body page 100. This data is stored in the file **data/01/03/book01/txt/b0000100.gdx**. Notice that in this example the ground truth text was accurately trimmed to the beginning and ending of the page.

```
221238  39479   &hyphen;Chairman's r03ja3.sgm [DOC:8] [NOTICE:79] [TEXT:13729]
221239  39480   Report          r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13729]
221240  39481   &hyphen;Committee r03ja3.sgm  [DOC:8] [NOTICE:79] [TEXT:13730]
221241  39482   Reports         r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13730]
221242  39483   &hyphen;Wrap-up/Annual r03ja3.sgm [DOC:8] [NOTICE:79] [TEXT:13731]
221243  39484   Schedule        r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13731]
221244  39485   of              r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13731]
221245  39486   Committee       r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13731]
221246  39487   Meetings.       r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13731]
221247  39488   It              r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13732]
221248  39489   is              r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13732]
221249  39490   imperative      r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13732]
221250  39491   that            r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13732]
221251  39492   the             r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13732]
221252  39493   meeting         r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13732]
221253  39494   be              r03ja3.sgm     [DOC:8] [NOTICE:79] [TEXT:13732]
. . .
222177  40418   full            r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222178  40419   power           r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222179  40420   operation       r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222180  40421   after           r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222181  40422   a               r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222182  40423   refueling       r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222183  40424   outage          r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222184  40425   when            r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222185  40426   low             r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222186  40427   power           r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222187  40428   physics         r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222188  40429   testing         r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222189  40430   revealed        r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222190  40431   an              r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222191  40432   improper        r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
222192  40433   configuration   r03ja3.sgm     [DOC:8] [NOTICE:82] [TEXT:13779]
```

Figure 16. Portion of a ground truth word index file.

3.2.2.9 Ground Truth Error Estimate Files

Locating page boundaries when matching a page's OCR text to the GPO typesetting text is a difficult, inexact process. Page boundaries may be obscured by a number of factors including quality of the OCR text and word order discrepancies between the OCR and GPO texts. To help manage this uncertainty, an error estimate is computed for each ground truth word index file

produced. This is a conservative estimate computed by aligning a page's OCR text with its corresponding ground truth text file. Blocks of out-of-order text are accounted for, and the remaining number of deleted and inserted words are used to represent how poorly the derived ground truth matches the page.

The error is estimated using the following formula:

$$error = \frac{(ins + del)}{nwords} \quad (1)$$

where *ins* is the number of inserted words after alignment and reordering, *del* is the number of deleted words after alignment and reordering, and *nwords* is the number of words recognized on the page by OCR. This error estimate loosely represents the reliability of the associated derived ground truth. Using these estimates, experiments can be designed by selecting pages that have an "acceptable" level of potential error.

The provided utilities compute an error estimate for every ground truth word index file in the database. Each of these error estimates is stored to a separate file. These small files have been concatenated together into a single compressed file for distribution. All the ground truth error estimates for the pages of a given FR book are stored in a subdirectory **book01/txt** in a single file named ***MMDD*.est**. For example, the error estimate for body page 100 is recorded in the file **data/01/03/book01/txt/0103.est** as:

```
b0000100 0.013699
```

In this case, the amount of word errors contained in the derived ground truth is estimated to be 1.4%, which is quite low.

3.2.2.10 Ground Truth SGML Files

The extracted ground truth word index files contain the hypothesized words on a page and their logical context. Given the context, the word index files are converted into an SGML format where tags (listed in Table 2) are provided to identify paragraphs of text, figures, tables, footnotes, and some hierarchical labeling of sections within the FR. The full set of tags is defined in the DTD file **dtd/fr94gpo.dtd**.

Ground truth SGML files can be used as baseline text for indexing and retrieval experiments. One of these files is provided for each ground truth word index file in the database. All ground truth SGML files for a given FR book are stored in a subdirectory **book01/txt** with extension **gml**. Figure 17 lists the derived SGML ground truth for the beginning and ending of body page 100 (shown in Figure 26). This data is stored in the file **data/01/03/book01/txt/b0000100.gml**.

3.2.2.11 OCR Scoring Files

The string alignment technology used in ground truth extraction has been bundled into a word-level scoring package which is also included in this distribution. This package is designed to take an OCR word index file and reconcile it against a corresponding ground truth word index file, accumulating various word-level statistics.

```
<DOC>
<DOCNO>
b0000100.gml
</DOCNO>
<NOTICE>
<TEXT>
&hyphen;Chairman's Report
</TEXT>
<TEXT>
&hyphen;Committee Reports
</TEXT>
<TEXT>
&hyphen;Wrap-up/Annual Schedule of Committee Meetings.
</TEXT>
<TEXT>
It is imperative that the meeting be held on this date to accommodate the scheduling
priorities of the key participants.
</TEXT>
<NOTICE-DATE>
Dated: December 28, 1993.
</NOTICE-DATE>
<TEXT>
Timothy M. Sullivan,
</TEXT>
<TEXT>
Advisory Committee Management Officer.
</TEXT>
<TEXT>
[FR Doc. 93&hyphen;32068 Filed 12&hyphen;30&hyphen;93; 8:45 am]
</TEXT>
<TEXT>
BILLING CODE 7510&hyphen;01&hyphen;M
</TEXT>
</NOTICE>
<NOTICE>
<TEXT>
NATIONAL FOUNDATION ON THE ARTS AND THE HUMANITIES
</TEXT>

. . .

<TEXT>
The proposed exemption would allow temporary relief from the requirements of 10 CFR part 50,
Appendix E, Section IV.F.2., for the licensee at HBR to annually exercise its emergency plan.
By letter dated November 21, 1993, as supplemented November 22, 1993, the licensee requested
an exemption from the requirements of 10 CFR part 50, Appendix E, to conduct an annual
exercise of the HBR Radiological Emergency Plan in 1993. The licensee had planned to conduct a
full-participation exercise involving the State of South Carolina and local response
organizations on November 30, 1993. The licensee requested that an exemption be granted for
the conduct of the onsite portion of the exercise because the licensee would not have
sufficient staff to conduct a meaningful exercise of the HBR Emergency Plan due to resource
constraints caused by an unscheduled outage to investigate and address core design issues.
This proposed delay will prevent HBR from meeting the requirement to conduct an annual
exercise of the HBR Emergency Plan. However, the licensee proposed that the offsite portion of
the exercise involving the State of South Carolina and local governmental authorities be
conducted as scheduled on November 30, 1993. The licensee requested a delay of the onsite
portion of the 1993 annual exercise from November 30, 1993, to the week of March 21, 1994. The
request to move the exercise date was originated by the licensee because they would be unable
to participate on November 30, 1993, as a result of their involvement at that time in
addressing safety issues identified during startup after refueling outage 15. The reactor was
ascending to full power operation after a refueling outage when low power physics testing
revealed an improper configuration
</TEXT>
</NOTICE>
</DOC>
```

Figure 17. Portion of a ground truth SGML file.

```
Thu Aug 27 09:36:03 1998

/gig19/imrec/fr94/dist/bin/wrdscore
    /gig19/imrec/fr94/dist/data/01/03/book01/wrk/b0000100.odx
    /gig19/imrec/fr94/dist/data/01/03/book01/wrk/b0000100.gdx

HYP File: /gig19/imrec/fr94/dist/data/01/03/book01/wrk/b0000100.odx
REF File: /gig19/imrec/fr94/dist/data/01/03/book01/wrk/b0000100.gdx

WORD SCORING STATISTICS:

# Reference Words  = 955
# Hypothesis Words = 949
# Correct Words    = 852
# Wrong Words      = 97
# Deleted Words    = 6
# Inserted Words   = 7

WORD ERROR RATE =  11.43%
```

Figure 18. Example word scoring report.

Scoring results are recorded for each FR page in an OCR scoring file. An example of a scoring report is shown in Figure 18. The scoring package accumulates errors after applying string alignment alone. No accounting for out-of-order blocks of text is currently done; however, such algorithms are provided in the distribution and could be incorporated. The word error rate is computed as:

$$word\_error = \frac{(wrn + del + ins)}{cor + wrn + del + ins} \qquad (2)$$

where *wrn* is the number of wrong words, *del* is the number of deleted words, *ins* is the number of inserted words, and *cor* is the number of correct words.

The scoring files can be used to assess and compare OCR accuracy. One of these files is provided for each pair of OCR and ground truth word index files in the database. All word scoring files for a given FR book are stored in a subdirectory **book01/txt** with extension **scr**. The example shown is for body page 100 and is stored in the file **data/01/03/book01/txt/b0000100.scr**.

3.2.2.12 Ground Truth Error Estimate Plot

A PostScript encoded plot file of all the ground truth error estimates is provided for each FR book in the database. An example of one of these plots is shown in Figure 6. All the error estimates for a book are sorted in increasing order and then plotted on a log scale. This gives the recipient a snapshot of the quality of the ground truth for a given FR book.

There is one **errest.ps** file provided in each **book01/txt** subdirectory in the database. The plot in Figure 6 is stored in the file **data/01/03/book01/txt/errest.ps**.

4. SUMMARY

NIST has produced a document image database designed to evaluate both DAR and IR technologies. The database is comprised of scanned images and ground truthed pages taken from the 1994 *Federal Register* for the month of January. In total, there are 4711 full page

images, comprising 20 FR books. An automated truthing process generated 4519 ground truth files (96% of the total number of images.) If there is sufficient interest, future volumes of other FR months may be released.

The ground truth files were created by matching noisy OCR from each page to typesetting files containing the text for an entire book. A page's location in the typesetting text was located, its boundaries detected, and its corresponding text extracted. This process is very difficult due to OCR errors and caveats of the typesetting files. To help manage uncertainty, an error estimate is provided with each derived ground truth file that conservatively represents the percentage of words missing and/or inserted in each ground truth file.

Figure 19 plots the total distribution of ground truth error estimates for this database (the entire month of January). The pie chart is divided into ranges of estimate values, starting with the largest wedge on the right and working clockwise: [0%-1%], [1%-5%], [5%-10%], and so on. The percentage of files represented by each wedge is provided in parentheses below each range label. For example, 53% of the ground truth files have an error estimate below 1%, whereas 17% of the ground truth files range between 1% and 5%. The number displayed on top of each wedge is the total number of pages counted in each range.

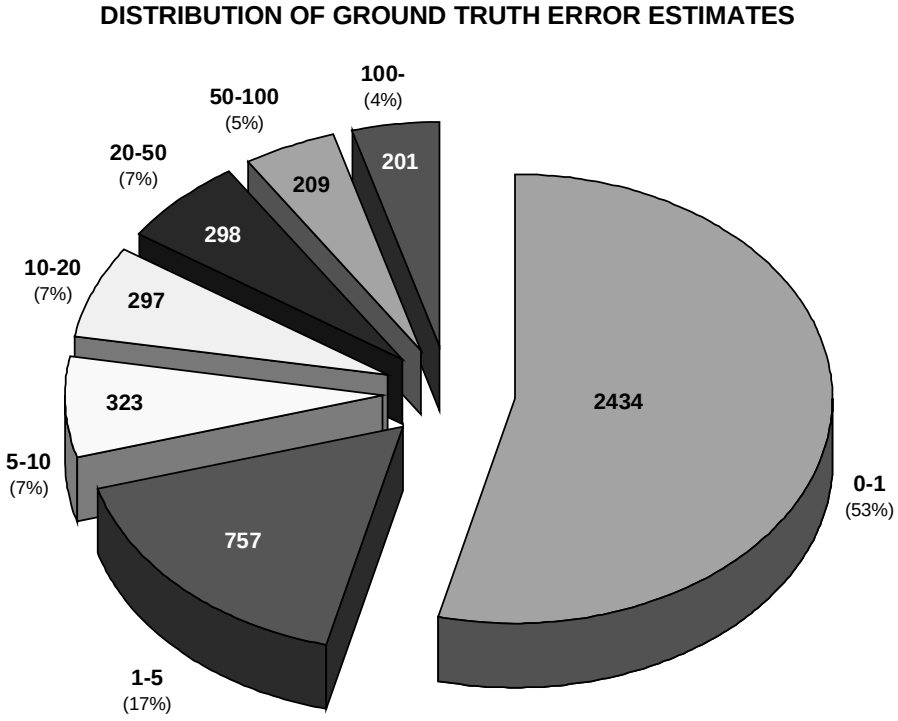

Figure 19. Ground truth error estimates for all of January.

These error estimates are provided so that experiments can be designed by selecting pages that have an "acceptable" level of potential error. As can be seen in the chart, 70% of the ground truth files have an error estimate below 5%, and 77% of the files are below 10%.

A host of data files are provided in addition to page images and their ground truth files. Commercial OCR results are provided on each page image, corresponding word error rates are

reported, image quality analysis results are included, and more. These files may be used to support a wide variety of experiments.

The software (in the form of Perl scripts and C programs) that was used to derive the ground truth files is also included in this distribution. These utilities serve as documentation and may be augmented and improved to produce similar data collections. With this new ground truthing technology, it is now feasible to produce much larger data sets, at much lower cost, than was ever possible with previous labor-intensive, manual data collection projects.

This is public domain code, and any portion of it, may be used without restrictions because it was created with U.S. government funding. This software was produced by NIST, an agency of the U.S. government, and by statute is not subject to copyright in the United States. Recipients of this software assume all responsibilities associated with its operation, modification, and maintenance.

In all, the distribution is 1.27 gigabytes in size, requiring two CD-ROMs. The first disc (614 megabytes in size) contains all the software and documentation for the distribution. The data files are divided across the CD-ROMs. The first disc contains the data files for FR days **01** through **13** (excluding day **05**, which is included on the second disc due to space limitations). The second disc contains day **05** and days **14** through **31**. The second CD-ROM (containing only data) is 649 megabytes in size. All the data files are compressed: image files are compressed using CCITT Group 4[16] and all other data files are compressed using GNU GZIP[13]. The GZIP-1.2.4 distribution is provided with the database.

Volume 1 of the Federal Register Document Image Database, *NIST Special Database 25*, is distributed by the Standard Reference Data Group at NIST. To get pricing information and/or to order the database on CD-ROM, please contact:

Standard Reference Data Program
National Institute of Standards and Technology
Bldg. 820, Rm. 113
Gaithersburg, MD 20899-0001
(301)975-2208 (voice)
(301)926-0416 (FAX)
srdata@nist.gov (e-mail)

This document references numerous NIST Internal reports. These papers are included in PostScript format in the distribution. They may also be accessed via our group's Web page, http://www.itl.nist.gov/div894/894.03 . Hardcopies may be requested from:

ITL Publications
National Institute of Standards and Technology
Bldg. 225, Rm. A216
Gaithersburg, MD 20899-0001
(301)975-2832 (voice)
(301)926-3696 (FAX)
itlab@nist.gov (e-mail)

5. REFERENCES

[1] M.D. Garris, "Document Image Recognition and Retrieval: Where Are We?," NIST Internal Report 6231, October 1998.

[2] ISO-8879, Information Processing – Text and Office Systems - Standard Generalized Markup Language (SGML), Standard by the International Organization for Standardization, 1986.

[3] E.M. Voorhees and D.K. Harman Eds., "The Fifth Text REtrieval Conference (TREC-5)," NIST Special Publication 500-238, November 1997.

[4] M. Cannon and P. Kelly, "An Automated System for Numerically Rating Document Image Quality," Proceedings of Document Recognition IV, Vol. 3027, SPIE, San Jose, CA, February 1997.

[5] M.D. Garris, W.W. Klein, "Creating and Validating a Large Image Database for METTREC," NIST Internal Report 6090, December 1997.

[6] H.G. Zwakenberg, "Inexact Alphanumeric Comparison," *The C Users Journal*, pp. 127-131, May 1991.

[7] "UW-II English/Japanese Document Image Database," CD ROM, University of Washington, 1995.

[8] ISO-9660, Information Processing – Volume and File Structure of CD-ROM for Information Interchange, Standard by the International Organization for Standardization, 1988.

[9] M.D. Garris, S. Janet, and W.W. Klein, "Impact of Image Quality on Machine Print Optical Character Recognition," NIST Internal Report 6101, December 1997.

[10] "TIFF Revision 6.0," Aldus Corporation, Seattle, WA, June 1992, off the Web at ftp://ftp.sgi.com/graphics/tiff/TIFF6.ps.Z .

[11] M.D. Garris, J.L. Blue, G.T. Candela, P.J. Grother, S.A. Janet, and C.L. Wilson, "NIST Form-Based Handprint Recognition System (Release 2.0)," NIST Internal Report and CD-ROM 5959, January 1997.

[12] "XDOC Data Format: Technical Specification," Part Number 00-08137-00, Xerox Corporation, March 1995.

[13] "GNU Project," Free Software Foundation, 59 Temple Place – Suite 330, Boston, MA, 02111, http://www.gnu.org .

[14] L. Wall, *et al*, "Practical Extraction and Report Language (Perl)," Ver. 5.004, Comprehensive Perl Archive Network (CPAN), http://www.perl.com .

[15] ISO/IEC 8859-1:1998, Information Technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1, Standard by the International Organization for Standardization, 1998.

[16] CCITT, "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus, Fascicle VII.3 - Rec. T.6," 1984.

APPENDIX A. SOFTWARE REFERENCE GUIDE

This appendix provides a reference guide to the software distributed with the Federal Register Document Image Database. All the software tools used to produce the data have been provided in this distribution.[3] The software is written as a collection of Perl scripts and C programs and is intended to document the data production process and enable recipients to augment and improve the process if desired.

This reference guide is comprised of two major items. The first is a sequence of tables that document the order in which the software utilities were invoked to produce the data. These tables are then followed by a series of manual pages that give an overview of each script or program and document their purpose and how they are invoked.

A.1 Order of Software Invocation

The illustration in Figure 1 depicts the processing of two sources of data (GPO-provided files on the left and OCR-generated files on the right) with the results of their matching in the middle at the bottom. Using this illustration as a reference, the software utilities are grouped into three categories: 1) utilities that process GPO Microcomp files; 2) utilities that generate OCR results; and 3) utilities that conduct matching between the GPO files and OCR results to generate pages of ground truth text. Each of these stages is represented by a separate table below. The middle column of these tables lists the file name of the utility; the left column lists the types of input to the utility by file extension; the right column lists the types of output (primarily by extension).

The first group listed in Table 3 are those scripts used to process the GPO Microcomp files. The top-level script, *gpomonth.pl*, processes all the Microcomp files for a month. The middle-level script, *gpoday.pl*, processes all the Microcomp files in each daily issue. The low-level scripts parse the Microcomp files for each day into SGML (*gpoparse.pl*), subsequently generating a master word index file (*mkwdindx.pl*) and corresponding word trigram files (*mktrigrm.pl and mkunqtri.pl*).

INPUT	**SCRIPT / PROGRAM**	OUTPUT
	gpomonth.pl	
	gpoday.pl	
xxx nnn	**gpoparse.pl**	sgm
sgm	**mkwdindx.pl**	wdx
wdx	**mktrigrm.pl**	*wd3*
wd3	**mkunqtri.pl**	*wf1*

Table 3. Order in which GPO Microcomp files are processed.

[3] All software has been provided except for the image quality assessment program developed at Los Alamos Laboratories [4]. This program was invoked in the Perl script *doimqual.pl*.

The second group is listed in Table 4 and contains the scripts and programs used to generate OCR results and predict OCR error. The top-level script, *ocrmonth.pl*, processes all the FR page images in a month. The middle-level script, *ocrbook.pl*, processes the FR page images scanned from a single book. The low-level program , *xreadpg*, runs ScanWorX on each page image; and the script, *doimgqual.pl*[3], assesses the image quality of the page image and computes a predicted word error rate.

INPUT	**SCRIPT / PROGRAM**	OUTPUT
	ocrmonth.pl	
	ocrbook.pl	
pct	**xreadpg**	xdc
pct	**doimqual.pl**[3]	qua

Table 4. Order in which OCR result files are generated.

The last group of utilities is listed in Table 5 and contains the scripts and programs used to match OCR results to the GPO files in order to generate page-level ground truth files. The top-level script, *trumonth.pl*, processes all the GPO files and OCR results in a month. The middle-level script, *trubook.pl*, generates all the ground truth files for all the pages of a FR book.

INPUT	**SCRIPT / PROGRAM**	OUTPUT
	trumonth.pl	
	trubook.pl	
xdc	**doxdcprs.pl**	oml
oml	**mkwdindx.pl**	odx
odx	**mktrigrm.pl**	*od3*
od3 *wf1*	**doqryhit.pl**	*hit*
hit	**rejhits.pl**	hit_g hit_b rejhits.ps
hit_g wdx	**cutchunk.pl**	*cdx*
odx *cdx* hit_g	**pgfrchnk**	gdx est
gdx	**wdx2sgml.pl**	gml
odx gdx	**wrdscore**	scr
est	**ploterr.pl**	errest.ps

Table 5. Order in which GPO files and OCR results are processed together.

The low-level utilities in this group begin with the script, *doxdcprs.pl*. This script parses the XDOC-formatted OCR results into SGML. *Mkwdindx.pl* converts the OCR SGML files into word index files. *Mktrigrm.pl* computes trigrams from each OCR word index file. *Doqryhit.pl* matches the OCR trigrams against the unique trigrams from the issue's GPO files and computes a hit for each page (the median of the clustered matches). *Rejhits.pl* rejects bad hits and then plots the results. (Accepted or "good" hits are labeled in Table 5 as hit_g, and rejected or "bad" hits are labeled as hit_b.) *Cutchunk.pl* extracts a chunk of GPO text centered around the hit location. *Pgfrchnk* trims the extracted chunk by aligning the OCR text to the GPO text, creating a ground truth word index file. *Wdx2sgml.pl* converts the ground truth word index file to SGML. *Wrdscore* reconciles the OCR word index file with the ground truth word index file, accumulating errors and computing a word error rate. Finally, *ploterr.pl* plots the ground truth error estimates computed by the program, *pgfrchnk*.

A.2 Manual Pages

This section contains manual pages for the software utilities included in this distribution. These manual pages are provided in plain text and in UNIX *nroff* format, both under the top-level distribution directory **man**. The utilities are listed here alphabetically. Refer to the previous section for the order of their execution.

NAME

cutchunk.pl - script for extracting a chunk of a book's GPO text assumed to contain a specific page

SYNOPSIS

cutchunk.pl *hit_file master_word_index*

DESCRIPTION

Cutchunk.pl is a PERL script that takes the GPO-related location stored in a page's hit file and extracts a conservatively large chunk of a book's GPO text. The extracted chunk is centered on the hit value and is assumed to contain the entire content of the corresponding page. The extracted passage of text is printed to *standard output*.

OPTIONS

hit_file

The hit file for the page currently being processed.

master_word_index

The master word index file containing all the GPO text for the corresponding book.

EXAMPLES

```
% cutchunk.pl $DAYDIR/book01/wrk/b0000100.hit \
     $DAYDIR/gpo/wrk/0103.wdx > $DAYDIR/book01/wrk/b0000100.cdx
```

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input

.hit	file with pinpoint location of page between OCR and GPO texts
.wdx	GPO master word index file for a book

output

.cdx	extracted GPO chunk word index file

SEE ALSO

trubook.pl(1), doqryhit.pl(1)

DIAGNOSTICS

BUGS

NAME

doqryhit.pl - script for pinpointing the location of a page with the GPO-provided text for an entire book

SYNOPSIS

doqryhit.pl *OCR_trigrams GPO_unique_trigrams*

DESCRIPTION

Doqryhit.pl is a PERL script that locates the area within a book's GPO text that holds the contents of a specific page. This is accomplished by matching word trigrams from a page's OCR text to the trigrams occurring uniquely in the book's GPO text. The location of each match is recorded, and the median matching trigram is selected as the *hit*. The location of the hit pinpoints the page within the book's text. In this way, the OCR results are used to "query" the GPO text.

The locations of the median matching trigram within the OCR and GPO texts are printed to *standard output* along with the total number of matching trigrams.

OPTIONS

OCR_trigrams

The OCR word trigram file used to query the book's GPO text.

GPO_unique_trigrams

The file containing all word trigrams (and their locations) that occur exactly once across the GPO text for a given book.

EXAMPLES

```
% doqryhit.pl $DAYDIR/book01/wrk/b0000100.od3 \
     $DAYDIR/gpo/wrk/0103.wf1 > $DAYDIR/book01/wrk/b0000100.hit
```

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input

.od3	OCR word trigram file
.wf1	GPO unique word trigram file

output

.hit	file with pinpoint location of page between OCR and GPO texts

SEE ALSO

trubook.pl(1), mkunqtri.pl(1)

DIAGNOSTICS

BUGS

NAME

doxdcprs.pl - script for parsing OCR results for a page from an XDOC file

SYNOPSIS

doxdcprs.pl [... *output flags* ...] *output_rootname XDOC_file*

DESCRIPTION

Doxdcprs.pl is a PERL script that is used to extract various types of information stored in an XDOC file. XDOC files contain the OCR results generated by Xerox's ScanWorX API. There are a number of flags which determine what types of information are to be parsed and extracted. If no flag is specified, the default is to extract the recognized text and output it in SGML format. If more than one flag is specified, then one output file is created for each flag specified. The complete set of output flags is defined below.

OPTIONS

... output flags ...

-sgml	SGML text (default)
-txtfmt	plain text (ISO 8859/1) with space formatting
-txtunfmt	plain text (ISO 8859/1) without space formatting
-wdcon	word confidence values
-chcon	character confidence values
-wdbox	word bounding boxes

output_rootname

The root file name to be used for output files.

XDOC_file

The XDOC file to be parsed.

EXAMPLES

```
% doxdcprs.pl -sgml $DAYDIR/book01/img/b0000100 > \
     $DAYDIR/book01/img/b0000100.xdc
```

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input	
.xdc	OCR XDOC file
output	
.oml	OCR SGML file
.fmt	plain text file with space formatting
.txt	plain text file without space formatting

- continued -

.wcn	word confidence file
.ccn	character confidence file
.wbx	word bounding box file

SEE ALSO

trubook.pl(1), xreadpg(1)

DIAGNOSTICS

BUGS

NAME

gpoday.pl - script for processing all GPO-provided files in a day

SYNOPSIS

gpoday.pl *install_dir* *src_GPO_dir* *dst_SGML_dir* *dst_WRK_dir*

DESCRIPTION

Gpoday.pl is a PERL script that processes all the GPO-provided files for a single daily issue of the *Federal Register*.

Step 1: (**gpoparse.pl**) parse GPO Microcomp files into SGML text files.

Step 2: (**mkwdindx.pl**) create a master word index file from the SGML files.

Step 3: (**mktrigrm.pl**) generate a file of word trigrams from the master word index file.

Step 4: (**mkunqtri.pl**) create a list of all unique trigrams from the word trigram file.

OPTIONS

install_dir
The full pathname to the directory in which the database distribution was installed.

src_GPO_dir
The pathname to the source directory containing GPO Microcomp files used to typeset the *Federal Register*. (*install_dir*/data/*MM*/*DD*/gpo/org)

dst_SGML_dir
The pathname to the destination directory where resulting SGML text files are to be stored. (*install_dir*/data/*MM*/*DD*/gpo/sgm)

dst_WRK_dir
The pathname to the destination directory where resulting word index and trigram files are to be stored. (*install_dir*/data/*MM*/*DD*/gpo/wrk)

EXAMPLES

**% gpoday.pl /usr/local/fr94 $DAYDIR/gpo/org $DAYDIR/gpo/sgm **
$DAYDIR/gpo/wrk
where $DAYDIR = /usr/local/fr94/data/01/03

- continued -

FILES

SEE ALSO

gpoparse.pl(1), mkwdindx.pl(1), mktrigrm.pl(1), mkunqtri.pl(1)

DIAGNOSTICS

BUGS

NAME

gpomonth.pl - master script for processing all GPO-provided files in a month

SYNOPSIS

gpomonth.pl *install_dir month*

DESCRIPTION

Gpomonth.pl is a PERL script that processes all the GPO-provided files for each issue of the *Federal Register* for an entire month. For each issue, this master script calls **gpoday.pl**.

OPTIONS

install_dir

The full pathname to the directory in which the database distribution was installed.

month

The month in the range [01..12] that is to be processed.

EXAMPLES

% gpomonth.pl /usr/local/fr94 01

FILES

SEE ALSO

gpoday.pl(1)

DIAGNOSTICS

BUGS

NAME

gpoparse.pl - convert Microcomp Federal Register data to SGML or readable text

SYNOPSIS

gpoparse.pl [*options*] *Microcomp-file* ...

DESCRIPTION

Gpoparse.pl is a Perl program that converts Microcomp files of *Federal Register* data to SGML (the default) or readable text.

All SGML output files contain ASCII text conforming to the Document Type Definition (DTD) that is present in the Federal Register Document Image Database distribution. The files contain text, as well as labels of structural elements such as agency names and footnotes. The text strings consist of standard SGML characters and many of the special characters defined in six ISO (International Standards Organization) character entity sets, such as Greek symbols.

The SGML output produced by processing each Microcomp data file from the 1994 *Federal Register* has been verified by the SGML parser *nsgmls* using the command:

% nsgmls -s -D *sgml-dir dtd-dir*/fr94gpo.dtd *sgml-file*

where *sgml-dir* is the location of the character entity set files and the inventory file *catalog.dat*, and *dtd-dir* is the location of the DTD file. Those files are supplied in the Federal Register Document Image Database distribution. The program *nsgmls* is part of the free software package *sp*, which is available at the URL:

http://www.jclark.com/sp/index.htm

By default, the program reads and processes every line of every Microcomp file specified on the command line. If no files are specified, the standard input is processed.

To simplify processing, all lines in the current Microcomp file are read up-front. Because text in the Microcomp format is largely free-format, the current file's lines are then concatenated into a single line. This behavior can be turned off with the *-o nocombine* option.

Next, that line is split into several lines each beginning with the sequence *\007F* or *\007S*, which correspond to *Federal Register* sections (e.g. Sunshine Act Meetings). (This behavior can be turned off with the *-o nosplit* option.) Those lines that are larger than 5 Kb are further split into pieces of approximately that length or less, each

- continued -

beginning with the sequence *\007I*, which corresponds to normal *Federal Register* text. (This behavior can be turned off with the *-o noresplit* option.) The 5 Kb size was chosen mainly because it appeared to be more efficient than 20 Kb or 100 Kb when doing later operations on the lines (edits and splits), but also because there appeared to be a bug in Perl's regular expression operations when handling much larger (e.g. many megabytes) sequences.

The processing of each line then begins with the following editing steps:

- garbage sequences such as \256MD[A-Z][A-Z]\257 are removed
- characters that have no impact (or unknown impact) on formatting (e.g. carriage returns, tabs, tildes, character *\000*) are removed
- sequences that have no impact (or unknown impact) on formatting (e.g. empty table elements, in-line graphics) are removed
- characters and sequences that seem to affect only spacing are mapped to blanks
- each sequence of multiple adjacent blanks is squeezed to a single blank

At that point, lines consisting only of zero or more blanks are discarded.

Each line is then broken into its components such as simple text, table begin and end codes, table elements, footnote text, and comments. This is done using a complicated regular expression matcher that is used to match from each known sequence beginning code (usually *\007* followed by one or more alphabetic characters) up to, but not including, the next sequence beginning code.

Each component is then processed based on the sequence beginning code. For example, the sequence:

```
\007I10 \007T2for further information contact:\007T1
Jay Craft at (202) 299 \37709 9630
```

is first passed to the subroutine that handles *\007* codes, then to the subroutine that handles *\007 I* codes, and finally to the subroutine that handles *\007 I10* codes. That subroutine recognizes the sequence as one of the many sequences for plain text, and passes the remaining charac-

- continued -

ters in the sequence to the subroutine that, for each requested output format, converts it to the correct string and writes it to the appropriate output stream. So in the case of the example sequence, the characters between *\007 T2* and *\007 T1* codes are capitalized, and the *\377 09* code is converted to "-" for text output and/or "*‐*" for SGML output.

OPTIONS

-b The base name of each output file will be the base name of the corresponding input file(s), followed by the appropriate extension for that output type. If processing the standard input when this option or *-r* has been selected, output files will be named as if the base name of the input filename was *stdin*. (The base name of a file is the part remaining after removing all characters through the final "/", if any are present.)

-r The base name of each output file will be the base name root of the corresponding input file(s), followed by the appropriate extension for that output type. (The base name root is the base name up to, but not including, the final ".", if any are present. If none are present, the base name root is the same as the base name.)

-D *outdir*
Output files should be placed in the directory *outdir*. The default is the current directory.

-I *incdir*
Prepend the directory *incdir* to the Perl array *@INC*, used to locate required library *.pl* files.

-L *libdir*
Set the library directory, used for locating auxiliary text files.

-S *sgmldir*
Set the SGML directory, used for locating SGML character entity set files.

-R *rootdir*
Set the distribution root directory, used when any of the above three are not set.

-s Do not write SGML files.

- continued -

-u Write unformatted text files.

-t Write plain text files.

-w # Set the output line width, used when writing to text and formatted files. The default is 80 characters.

-l # Instead of processing all lines of all specified files, stop execution after the specified number of lines.

-U Do not write updates on progress to terminal (*/dev/tty*). The updates consist of the numbers and percentages of files and lines that have been processed.

-o *option[,option...]*
Set miscellaneous long-name options. To specify multiple long-name options, the command line should have a single *-o* followed by a string comprised of the desired long-name options separated by commas. (Passing multiple *-o* strings on the command line results in undefined behavior.) The long-name options are:

nocombine Do not combine all lines in a file into one line. This typically has the minor effect that text sequences split over multiple lines are processed separately, with all lines other than the first being treated as generic text. For example the lines:

```
\007I04 Acting Manager,
Engine and Propeller Directorate,
Aircraft Certification Service.
```

will be processed with the first line being passed to the function that handles and prints *\007 I04* sequences and the other lines being passed to the function that simply prints generic text.

nosplit If combining lines, do not then split into sections beginning with \007[FS] byte sequences, which are specific book sections such as Sunshine Act Meetings.

noresplit If splitting lines, split up very long sections into approximately 5 Kb pieces

fmt Write formatted text file(s)

- continued -

footrefs Write footnote reference file(s)

footnotes Write footnote text file(s)

contents Write files with summaries of input files' contents

sections Write section log file(s)

tables Write tables file(s)

comments Write comments file(s)

seqlog Write sequence log file(s)

wordlog Write word log file(s)

linelog Write line log file(s)

-h Print this help message to the standard output and exit with status of zero.

FILES

Input files

The Microcomp files with base names matching the following patterns typically contain data as described:

*r** text for the normal Federal Register sections

*[po]** text for Presidential Documents

*adv** checklist text

*a** (other than those files matching *adv**) text for Reader Aids sections

*u** Federal Register text (all 94 of these files comprise the November 14, 1994 Federal Register book)

Output files

The program can produce several types of output files. When neither the *-b* nor the *-r* options, are specified, the various output filenames are fixed - for each type of output requested, one file is created containing that output for all input files. The output type determines the filename prefix, and the suffix is *.out*. The filenames are listed below.

- continued -

When either the *-b* or the *-r* option is specified, the various output filenames are generated from the base names or base name roots of the input files, with extensions appended based on the output type. Those extensions are specified in the parentheses in the list below:

sgml.out (**.sgm**)
SGML output file

unfmt.out (**.unfmt**)
Unformatted text output file. No effort is made to make the output look like formatted Federal Register pages - there is no indenting, long lines are not wrapped, etc. This file will contain characters in the ISO 8859/1 character set beyond just those in the printable ASCII range. For example, the byte value hexadecimal 0xBD, when printed, displays on most terminals as single-character-sized "1/2".

tables.out (**.tables**)
List of each table's formatting code and table headings and elements

comments.out (**.comments**)
List of comment text found in the Microcomp codes, along with the location (file and line number)

seq.log (**.seq.log**)
Log file of Microcomp file sequences

word.log (**.word.log**)
Log file of Microcomp file text words

line.log (**.line.log**)
Log file of Microcomp file lines

Code to handle the production of the following files is present in *gpoparse.pl*. However their contents are incomplete, either because having the output was not as high a priority as SGML output, or due to time constraints.

fmt.out (**.fmt**)
Formatted text output file. Some effort is made to make the output look like formatted Federal Register pages - there is indenting, long lines are wrapped, etc.

- continued -

text.out (**.text**)
Plain text output file. Characters other than the printable ASCII set are printed in the closest possible representation or not printed at all. For example, the fraction one-half would be printed as the three character sequence "1/2".

contents.out (**.contents**)
List of elements found in the Microcomp (tables, graphics, footnotes, comments, etc.)

sections.out (**.sections**)
Log file of Federal Register book sections

footnotes.out (**.footnotes**)
Log file of footnote texts and their locations (file and line number)

footrefs.out (**.footrefs**)
Log file of footnote references and their locations (file and line number)

dictionary.out (**.dictionary**)
Log file of words found in the Microcomp codes

Auxiliary files

The program *gpoparse.pl* requires four library files, all of which are present in the Federal Register Document Image Database distribution:

octal.pl Functions for printing strings with each non-ASCII character replaced by "[*nnn*]", where *nnn* is the character's octal value.

timestmp.pl Function to return a string specifying the current time, meant to be prepended to the lines written to log files. The format of the string is:
<MONTH> <DAY> <HOUR>:<MIN>:<SEC>

sdata.pl Code to read SGML character entity set files. The files contain mappings of special character names to SDATA strings.

- continued -

For example, the mapping for the fraction one-half is specified as:

```
<!ENTITY frac12 SDATA "[frac12]">
```

sgmlpars.pl Operations to maintain a stack of open SGML tags. The stack is used, for example, at the end of processing a file, to close all SGML tags still open at that point.

This distribution provides six SGML character entity set files. In addition to being needed for the SGML parser *nsgmls* to verify SGML output, the files are read directly by *gpoparse.pl*. They contain mappings that are stored in an associative array, and values from that array are written to SGML output files as appropriate. This simplifies *gpoparse.pl*, since it does not need to contain hard-coded SGML-specific information. The character entity set files are:

isolat1.dat SGML character entity set file for Latin characters such as the grave-accented a, represented by "*à*"

isogrk1.dat SGML character entity set file for Greek characters such as *beta*, represented by "*&bgr;*"

isonum.dat SGML character entity set file for numeric and special graphic characters such as the plus-or-minus sign, represented by "*±*"

isopub.dat SGML character entity set file for publishing characters such as the fraction one-third, represented by "*⅓*"

isotech.dat SGML character entity set file for general technical characters and symbols such as the greater-than-or-equal-to symbol, represented by "*≥*"

isodia.dat SGML character entity set file for diacritical marks, such as "*˜*"

This distribution also provides two files needed by the SGML parser *nsgmls* along with the SGML character entity set files to verify the integrity of SGML output files:

- continued -

fr94gpo.dtd A Document Type Definition for the SGML files generated by *gpoparse.pl*.

catalog.dat A catalog of the SGML character entity set files.

ENVIRONMENT

The program recognizes several environment variables. Each of the variables has a corresponding command line option. The command line options override the environment variables.

GPOPARSE_LIBDIR

The files in this directory contain Federal Register section headers. A header contains the text at the beginning of certain book sections such as the masthead. There is no default value for this directory, so if the directory is not specified on the command line via the *-L* flag, either this environment variable must be defined or else the root directory must be specified.

GPOPARSE_SGMLDIR

The program reads the six SGML character entity set files from its SGML directory. There is no default value for this directory, so if the directory is not specified on the command line via the *-S* flag, either this environment variable must be defined or else the root directory must be specified.

GPOPARSE_INCDIR

If no include directory is specified via the *-I* command line flag, and if this variable is set, then the value is a string that is prepended to the list of directories that Perl searches when looking for library files specified by the *require* construct. By default, this list is Perl's *@INC* array, the value of which is whatever was compiled into Perl when it was installed.

The four library files that *gpoparse.pl* requires are part of the Federal Register Document Image Database distribution and are described in the Auxiliary Files section of this manual page.

- continued -

GPOPARSE_ROOTDIR
If any of the include, library or SGML directories are not specified (via either the command line options or the environment variables), they can be specified at once with this variable if they are in the same directory.

EXAMPLES

The following shell command result in the file *sgml.out* being written:

```
% gpoparse.pl -L/usr/local/fr94/lib/gpoparse \
          -S/usr/local/fr94/lib/sgml \
          -I/usr/local/fr94/lib \
          r03ja0.xxx >& log
```

(All examples assume that the user's shell is *csh* or one that recognizes its syntax.)

These shell commands also result in the file *sgml.out* being written:

```
% setenv GPOPARSE_LIBDIR /usr/local/fr94/lib/gpoparse
% setenv GPOPARSE_SGMLDIR /usr/local/fr94/lib/sgml
% setenv GPOPARSE_INCDIR  /usr/local/fr94/lib
% gpoparse.pl r03ja[023].xxx >& log
```

Note that only one output file is written even though there are multiple input files.

Assume that the above environment variables are still set for all further examples. Then the following command results in an SGML output file being written for each input file:

```
% gpoparse.pl -D/tmp -r r03ja[023].xxx >& log
```

The output files will be named *r03ja0.sgm*, *r03ja2.sgm* and *r03ja3.sgm*, and they will be created in the directory */tmp*.

Similarly, the following command turns off SGML output and turns on the unformatted output. The output files will have the extension *.unfmt*.

```
% gpoparse.pl -D/tmp -r -s -u r03ja[023].xxx >& log
```

DIAGNOSTICS

If any fatal or serious errors occur, the program exits with a non-zero status. Otherwise it exits with a status of zero. Non-serious errors can occur, for example, due to typographical errors present in the Microcomp files.

- continued -

BUGS

Microcomp sequences that have not been identified (usually infrequently occurring sequences) are typically ignored during processing.

Atypical mark-up such as some footnote references are not labeled in SGML output as anything other than regular text.

Microcomp sequences that have been identified as having one meaning may have different meanings in other contexts and therefore might be handled incorrectly.

NAME

mktrigrm.pl - script for generating word trigrams from a word index file

SYNOPSIS

mktrigrm.pl *word_index_file*

DESCRIPTION

Mktrigrm.pl is a PERL script that inputs a word index file and creates word trigrams, printing the resulting trigram list to *standard output*.

OPTIONS

word_index_file

The word index file from which the trigrams are to be generated.

EXAMPLES

```
% mktrigrm.pl $DAYDIR/gpo/wrk/0103.wdx > \
     $DAYDIR/gpo/wrk/0103.wd3

% mktrigrm.pl $DAYDIR/book01/wrk/b0000100.odx > \
     $DAYDIR/book01/wrk/b0000100.od3
```

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input-1

.wdx GPO master word index file

output-1

.wd3 GPO word trigram file

input-2

.odx OCR word index file

output-2

.od3 OCR word trigram file

SEE ALSO

gpoday.pl(1), trubook.pl(1)

DIAGNOSTICS

BUGS

NAME

mkunqtri.pl - script for locating the unique trigrams in a word trigram file

SYNOPSIS

mkunqtri.pl *trigram_file*

DESCRIPTION

Mkunqtri.pl is a PERL script that takes a word trigram file and locates all trigrams that occur exactly once. Each unique trigram is printed to *standard output* along with its location within the input file.

OPTIONS

trigram_file

The file within which unique word trigrams are to be searched.

EXAMPLES

```
% mkunqtri.pl $DAYDIR/gpo/wrk/0103.wd3 > \
     $DAYDIR/gpo/wrk/0103.wf1
```

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input

.wd3 GPO word trigram file

output

.wf1 GPO unique trigram file

SEE ALSO

gpoday.pl(1)

DIAGNOSTICS

BUGS

NAME

mkwdindx.pl - script for creating a word index file from a list of SGML files

SYNOPSIS

mkwdindx.pl [**-sgml2iso**] ... *SGML files* ...

DESCRIPTION

Mkwdindx.pl is a PERL script that processes a list of SGML files and concatenates them together, reformatting the text and tags into a word index file. The resulting word index is printed to *standard output*.

Each row in the word index file corresponds to each successive word in the SGML files. A row has the following fields:

Field 1: Global word index - incremented for each word across all SGML files.

Field 2: Local word index - reset for each SGML file and incremented for each word within the file.

Field 3: Word string - the current word.

Field 4: SGML file name - file from which the word was extracted.

Fields 5-N: SGML tags - those tags open when the word was encountered.

This representation is critical in tracking text throughout the ground truth extraction process. It provides cross-referencing and preserves the logical context for each word.

The *Federal Register* document image database uses SGML entities to represent ISO Latin-1 characters. Specifying the optional *-sgml2iso* flag causes these SGML entities to be converted to the ISO 8859/1 8-bit character set.

OPTIONS

-sgml2iso

Converts ISO Latin-1 SGML character entities to their equivalent ISO 8859/1 8-bit values.

... SGML files ...

One or more SGML formatted text files in the order to be converted and concatenated into a word index file.

- continued -

EXAMPLES

```
     % mkwdindx.pl $DAYDIR/gpo/sgm/*.sgm > \
          $DAYDIR/gpo/wrk/0103.wdx

     % mkwdindx.pl $DAYDIR/book01/wrk/b0000100.oml > \
          $DAYDIR/book01/wrk/b0000100.odx

     where $DAYDIR = /usr/local/fr94/data/01/03
```

FILES

input-1
.sgm GPO SGML files
output-1
.wdx GPO master word index file

input-2
.oml OCR SGML file
output-2
.odx OCR word index file

SEE ALSO

gpoday.pl(1), trubook.pl(1)

DIAGNOSTICS

BUGS

NAME

ocrbook.pl - script for conducting OCR on all the pages published in a book

SYNOPSIS

ocrbook.pl *server install_dir src_image_dir dst_XDOC_dir*

DESCRIPTION

Ocrbook.pl is a PERL script that invokes a commercial OCR package to recognize each page image scanned from a *Federal Register* book. This script calls the OCR application program **xreadpg**.

OPTIONS

server

The designated OCR server.

install_dir

The full pathname to the directory in which the database distribution was installed.

src_image_dir

The pathname to the source directory containing the book's image files. (*install_dir*/data/*MM*/*DD*/book*BB*/img)

dst_XDOC_dir

The pathname to the destination directory where resulting OCR results are to be stored. (*install_dir*/data/*MM*/*DD*/book*BB*/wrk)

EXAMPLES

% ocrbook.pl hostname /usr/local/fr94 $DAYDIR/book01/img \ $DAYDIR/book01/wrk

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

SEE ALSO

xreadpg(1)

DIAGNOSTICS

BUGS

NAME

ocrmonth.pl - master script for conducting OCR on all the pages/books published in a month

SYNOPSIS

ocrmonth.pl *install_dir month*

DESCRIPTION

Ocrmonth.pl is a PERL script that runs a commercial OCR package across all pages of the *Federal Register* published in a month. For each scanned book, this master script calls **ocrbook.pl**.

OPTIONS

install_dir

The full pathname to the directory in which the database distribution was installed.

month

The month in the range [01..12] that is to be processed.

EXAMPLES

% ocrmonth.pl /usr/local/fr94 01

FILES

SEE ALSO

ocrbook.pl(1)

DIAGNOSTICS

BUGS

NAME

pgfrchnk - program for trimming a chunk of a book's GPO text to be a page's ground truth text

SYNOPSIS

pgfrchnk [**-v**] *OCR_word_index chunk_word_index hit_file output_file*

DESCRIPTION

Pgfrchnk is a C program that string aligns a page's OCR text with an extracted chunk of GPO text. The alignment between the two passages of text starts at the hit point and proceeds backwards and forwards. Page boundaries are determined to occur where the alignment sufficiently degrades. The chunk is then trimmed at the page boundaries and assumed to be the ground truth for the page.

Errors in the OCR text and anomalies in the GPO text make it very difficult to accurately determine page boundaries. To help manage this uncertainty, an error estimate for the extracted ground truth is computed. The OCR text is realigned with the extracted ground truth text, and out-of-order blocks of text are matched up. The number of remaining deleted and inserted words are accumulated and divided by the total number of words recognized on the page by OCR. This ratio serves as a conservative indicator of how well the alleged ground truth text matches the true contents of the page. The program prints this error estimate to *standard output*.

OPTIONS

-v Turns on verbose mode so that progress of the program may be tracked.

OCR_word_index
The OCR word index file for the page being processed.

chunk_word_index
The GPO chunk word index file extracted from the corresponding book's GPO text.

hit_file
The pinpointing location of a page between its OCR and GPO texts.

output_file
The extracted ground truth word index file.

- continued -

EXAMPLES

```
% pgfrchnk $DAYDIR/book01/wrk/b0000100.{odx,cdx,hit,gdx} > \
     $DAYDIR/book01/wrk/b0000100.est

where $DAYDIR = /usr/local/fr94/data/01/03
```

FILES

input

.odx	OCR word index file
.cdx	GPO chunk word index file
.hit	file with pinpoint location of page between OCR and GPO texts

output

.gdx	ground truth word index file
.est	ground truth error estimate file

SEE ALSO

trubook.pl(1), doqryhit.pl(1)

DIAGNOSTICS

BUGS

NAME

ploterr.pl - script for plotting the ground truth error estimates for an entire book

SYNOPSIS

ploterr.pl *install_dir error_est_dir*

DESCRIPTION

Ploterr.pl is a PERL script that takes all the error estimates, one for each derived ground truth file, and plots them in sorted order on a log scale. The resulting plot is stored in a PostScript formatted file. This provides a visual representation of the quality of ground truth for all the pages in a book. Because the points are plotted on a log scale, error estimates exactly equal to 0 are plotted at 0.01 in the graph.

WARNING: This script should only be used if the host system has the program **gnuplot** installed and it is in the environment's execution path.

OPTIONS

install_dir

The full pathname to the directory in which the database distribution was installed.

error_est_dir

The pathname to the directory containing the error estimate files for all the ground truth files derived for the pages of a book. (install_dir/data/MM/DD/bookBB/wrk)

EXAMPLES

% ploterr.pl /usr/local/fr94 $DAYDIR/book01/wrk

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input

.est ground truth error estimate files

output

errest.pts sorted points file containing all error estimates for a book

ploterr.ps plot of an entire book's ground truth error estimates

SEE ALSO

trubook.pl(1), pgfrchnk(1)

DIAGNOSTICS

BUGS

NAME

rejhits.pl - script for rejecting hits that fail to accurately pinpoint the location of a page with a book's GPO text

SYNOPSIS

rejhits.pl [**-gnuplot** *install_dir*] *hit_dir*

DESCRIPTION

Rejhits.pl is a PERL script that automatically determines if a derived hit between a page's OCR text and its book's GPO text is to be assumed accurate or not. A hit file contains 3 values. The first value is the location of the median matching word trigram in the page's OCR text. The second value is the location of the median matching trigram in the corresponding book's GPO text. The third value is the total number of matching trigrams.

A hit is rejected if its GPO-related trigram location (the second value) does not follow the linear trends established by its immediate neighbors. All pages with rejected hits are removed from further processing and have no ground truth text extracted. Rejected hit files are renamed with extension **bad**.

The optional flag *-gnuplot* directs the script to generate a plot of all accepted and rejected hits after processing an entire book. Each point in the plot is the GPO-related location (the second value) in a page's hit file. The resulting plot is stored in a PostScript formatted file.

WARNING: The plot option should only be used if the host system has **gnuplot** installed, and its executable is in the environment's execution path. If the plot flag is used, then the *install_dir* argument must also be provided.

OPTIONS

-gnuplot

Directs the script to generate a gnuplot after processing all the pages in a book.

install_dir

The full pathname to the directory in which the database distribution was installed. (Must be specified if the plot flag is used.)

hit_dir

The pathname to the directory containing all the hit files for a book. (*install_dir*/data/*MM*/*DD*/book*BB*/wrk)

- continued -

EXAMPLES

% rejhits.pl -gnuplot /usr/local/fr94 $DAYDIR/book01/wrk

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input

.hit	file with pinpoint location of page between OCR and GPO texts

output

.bad	rejected hit files
goodhits.pts	points file containing accepted hit values
badhits.pts	points file containing rejected hit values
rejhits.ps	plot of a book's accepted and rejected hits

SEE ALSO

trubook.pl(1), doqryhit.pl(1)

DIAGNOSTICS

BUGS

NAME

trubook.pl - script for matching OCR page results to GPO-provided files and extracting ground truth text for all the pages published in a book

SYNOPSIS

trubook.pl *install_dir book_wrk_dir*

DESCRIPTION

Trubook.pl is a PERL script that matches OCR page results to its corresponding book's GPO parsed text, producing ground truth text for each page in a *Federal Register* book.

Step 1: (**doxdcprs.pl**) parse OCR results from a page's XDOC file into an SGML text file.

Step 2: (**mkwdindx.pl**) convert the OCR SGML file into a word index file.

Step 3: (**mktrigrm.pl**) compute word trigrams from OCR word index file.

Step 4: (**doqryhit.pl**) match the page's OCR word trigrams to its book's GPO unique word trigrams, locating a hit.

Step 5: (**rejhits.pl**) reject inaccurate hits and plot rejection results.

Step 6: (**cutchunk.pl**) if hit is accepted, extract chunk of GPO text centered on the hit.

Step 7: (**pgfrchnk**) string align OCR page text to the GPO chunk text, determine page boundaries in the chunk, extract ground truth text from the chunk, store extracted text to a word index file, and compute an error estimate.

Step 8: (**wdx2sgml.pl**) convert ground truth word index file to an SGML text file.

Step 9: (**wrdscore**) compute word error rate between page's OCR word index file and its derived ground truth word index file.

Step 10: (**ploterr.pl**) plot ground truth error estimates for all the pages in a book in sorted order on a log scale.

- continued -

OPTIONS

install_dir
The full pathname to the directory in which the database distribution was installed.

book_wrk_dir
The full pathname to the directory containing all the OCR results for all the pages in a book. (*install_dir*/data/*MM*/*DD*/book*BB*/wrk)

EXAMPLES

% trubook.pl /usr/local/fr94 $DAYDIR/book01/wrk

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

SEE ALSO

doxdcprs.pl(1), mkwdindx.pl(1), mktrigrm.pl(1), doqryhit.pl(1), rejhits.pl(1), cutchunk.pl(1), pgfrchnk(1), wdx2sgml.pl(1), wrdscore(1), ploterr.pl(1)

DIAGNOSTICS

BUGS

NAME

trumonth.pl - master script for matching OCR page results to GPO-provided files and extracting ground truth text for each page in a month

SYNOPSIS

trumonth.pl *install_dir month*

DESCRIPTION

Trumonth.pl is a PERL script that matches OCR page results to its corresponding book's GPO parsed text, producing ground truth text for each *Federal Register* page published in a month. For each published book, this master script calls **trubook.pl**.

OPTIONS

install_dir

The full pathname to the directory in which the database distribution was installed.

month

The month in the range [01..12] that is to be processed.

EXAMPLES

% trumonth.pl /usr/local/fr94 01

FILES

SEE ALSO

trubook.pl(1)

DIAGNOSTICS

BUGS

NAME

wdx2sgml.pl - script for converting a word index file into an SGML text file

SYNOPSIS

wdx2sgml.pl *word_index_file sgml_file*

DESCRIPTION

Wdx2sgml.pl is a PERL script that takes a word index file and converts its contents into an SGML formatted text file. This is the inverse process to **mkwdindx.pl**.

OPTIONS

word_index_file

The word index file that is to be converted.

sgml_file

The resulting SGML text file.

EXAMPLES

```
% wdx2sgml.pl $DAYDIR/book01/wrk/b0000100.gdx \
     $DAYDIR/book01/wrk/b0000100.gml
```

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input

.gdx ground truth word index file

output

.gml ground truth SGML file

SEE ALSO

trubook.pl(1), mkwdindx.pl(1), wrdscore.pl(1)

DIAGNOSTICS

BUGS

NAME

wrdscore - program for assessing word-level OCR performance

SYNOPSIS

wrdscore [**-v**] [**-sm**] [**-na**] *OCR_word_index truth_word_index*

DESCRIPTION

Wrdscore is a C program that reconciles an OCR generated passage of text with its corresponding ground truth, accumulating word-level error statistics. The input files are word index files.

The first optional flag enables a verbose mode for tracking the program's progress. The next two flags alter the method in which the texts are aligned and scored. The first of these options, *-sm* (stands for "splits and merges"), directs the alignment algorithm to account for splits and merges when assessing scores. The second option, *-na* (stands for "no anchoring"), directs the alignment algorithm not to do an initial anchoring of the beginning of lines between the OCR and ground truth texts. Instead, the program simply aligns the entire text passages as two long strings.

The string alignment technology used in this scoring package was designed to align large passages of text that are formatted in heterogeneous units. For example in the *Federal Register* document image database, the OCR results are reported line-by-line whereas the ground truth files are in general formatted paragraph-by-paragraph.

To accomplish this, the beginning of OCR lines are matched to the beginning of ground truth paragraphs. These matches are designated as "anchors". The intervening OCR and ground truth texts between anchors are then treated as single strings and aligned. In general, partitioning the alignment improves execution efficiency (especially on low quality OCR results), and the anchors act as synchronization points so that the alignment process may recover from particularly poor OCR.

By default, the program will not account for splits and merges and will derive a cursory set of anchors before further string alignment. The program produces a scoring report which is printed to *standard output*. This program was used to score every page's OCR word index file in the database with its corresponding ground truth word index file.

- continued -

OPTIONS

-v Turns on verbose mode so that progress of the program may be tracked.

-sm Directs the alignment algorithm to account for splits and merges when accumulating error statistics.

-na Directs the alignment algorithm to not match anchors, but directly align the two passages of text as a pair of long strings.

OCR_word_index
The OCR word index file for the page being processed.

truth_word_index
The ground truth word index file for the page being processed.

EXAMPLES

**% wrdscore $DAYDIR/book01/wrk/b0000100.odx **
$DAYDIR/book01/wrk/b0000100.gdx > $DAYDIR/book01/wrk/b0000100.scr

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input
.odx OCR word index file
.gdx ground truth word index file
output
.scr scoring report file

SEE ALSO

trubook.pl(1)

DIAGNOSTICS

BUGS

NAME

xreadpg - program for running Xerox's ScanWorX API on a page image file

SYNOPSIS

xreadpg [**-fr94hdr**] [**-**(**i** | **f** | **l** | **x**)] *server image_file output_file* [*lexicon_file*]

DESCRIPTION

Xreadpg is an application program written in C that invokes Xerox's ScanWorX API on a specified image file. The image file is expected to contain a full-page image scanned binary and stored in the NIST IHead format.

There are a number of options controlling this program. If the optional flag *-fr94hdr* is used, then the program searches the image for a specific page header included on most *Federal Register* pages. If the header is located, it is removed from the image prior to recognition. The location of the detected header is stored in a header cut file. If no header is located, a location value of "-1" is stored.

The next set of optional flags *-(i | f | l | x)* dictate what format the OCR results are to be stored in. These flags are defined below. Plain text, the first option, is the default.

This program was used to process all the page images in the *Federal Register* document image database, with the results stored as XDOC files. **Doxdcprs.pl** is a PERL script that is capable of parsing out many of the various types of information (including recognized text) stored in XDOC files.

OPTIONS

-fr94hdr

Directs the program to locate and erase a *Federal Register* page header from the input image before OCR processing occurs, storing the detected location to a header cut file with the same root file name as the specified *output_file* with extension **hdy**.

-(**i** | **f** | **l** | **x**)

This flag designates the output file format:

-i plain ISO 8859/1 text (default)
-f Xerox FORMS format
-l XDOC-LITE format
-x full XDOC format

server

The designated ScanWorX OCR server.

- continued -

image_file
The IHead binary page image to be recognized.

output_file
The OCR results output file.

lexicon_file
Optional lexicon file to help improve recognition performance.

EXAMPLES

% xreadpg -fr94hdr -x hostname $DAYDIR/book01/img/b0000100.pct \ $DAYDIR/book01/wrk/b0000100.xdc

where $DAYDIR = /usr/local/fr94/data/01/03

FILES

input
.pct IHead image file
output
.xdc XDOC OCR results file
.hdy Header cut file

SEE ALSO

ocrbook.pl(1), doxdcprs.pl(1)

DIAGNOSTICS

BUGS

APPENDIX B. IHEAD IMAGE FILE FORMAT

Image file formats and effective data compression and decompression are critical to the usefulness of image archives. In this application, a raster image is a digital encoding of light reflected from discrete points on a scanned document page. The 2-dimensional area of the form is divided into discrete locations according to the resolution of a specified grid. Each cell of this grid, which is called a pixel, is represented by a single bit value 0 or 1; 0 represents a pixel predominately white, 1 represents a pixel predominately black. This 2-dimensional sampling grid is then stored as a 1-dimensional vector of pixel values in raster order, left to right, top to bottom. Successive scan lines (top to bottom) contain the values of a single row of pixels from the grid.

Certain attributes of a raster image are required to interpret the 1-dimensional pixel data as a 2-dimensional image. Examples of such attributes are the pixel width and pixel height of the image. These attributes are stored in a machine readable header prefixed to the raster bit stream. A program that is used to manipulate the raster data must first read the header to determine the proper interpretation of the data which follows it.

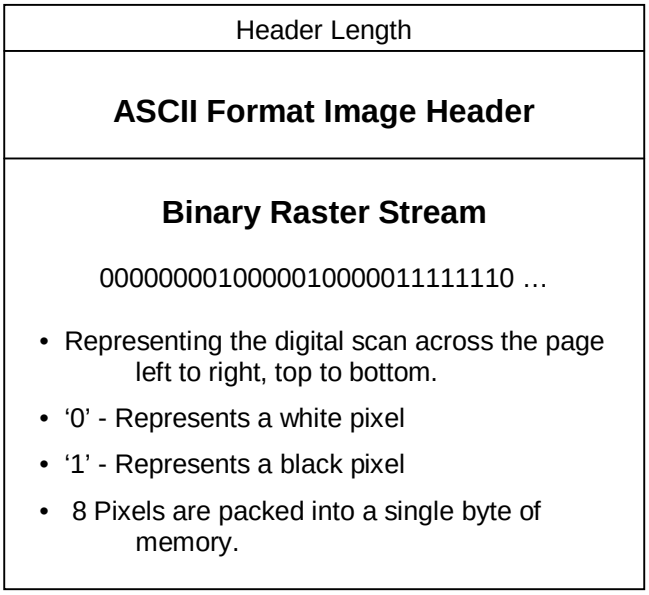

Figure 20. An illustration of the IHead raster file format.

Numerous image formats exist, but most image formats are proprietary. Some are widely supported on small personal computers and others are supported on larger workstations. A header format named IHead has been developed for use as a general purpose image interchange format. The IHead header is an open image format which can be universally implemented across heterogeneous computer architectures and environments. Both documentation and source code for the IHead format are publicly available and included with this database. IHead has been designed with an extensive set of attributes in order to adequately represent both binary and gray level images, to represent images captured from different scanners and cameras, and to satisfy the image requirements of diversified applications including, but not limited to, image

archival/retrieval, character recognition, and fingerprint classification. Figure 20 illustrates the IHead format.

Since the header is represented by the ASCII character set, IHead has been successfully ported and tested on several systems including UNIX workstations and servers, DOS personal computers, and VMS mainframes.[1] All attribute fields in the IHead structure are of a fixed length with all multiple character fields null-terminated, allowing the fields to be loaded into main memory in two distinct ways. The IHead attribute fields can be parsed as individual characters and null-terminated strings, an input/output format common in the C programming language, or the header can be read into main memory using record-oriented input/output. A fixed-length field containing the size in bytes of the header is prefixed to the front of an IHead image file as shown in Figure 20.

```
/*************************************************************
           File Name: IHead.h
           Package:   NIST Internal Image Header
           Author:    Michael D. Garris
           Date:      2/08/90
*************************************************************/

/* Defines used by the IHead structure */
#define IHDR_SIZE             288     /* len of hdr record (always even bytes) */
#define SHORT_CHARS    8      /* # of ASCII chars to represent a short */
#define BUFSIZE               80      /* default buffer size */
#define DATELEN               26      /* character length of data string */

typedef struct ihead{
   char id[BUFSIZE];                  /* identification/comment field */
   char created[DATELEN];             /* date created */
   char width[SHORT_CHARS];           /* pixel width of image */
   char height[SHORT_CHARS];          /* pixel height of image */
   char depth[SHORT_CHARS];           /* bits per pixel */
   char density[SHORT_CHARS];         /* pixels per inch */
   char compress[SHORT_CHARS];        /* compression code */
   char complen[SHORT_CHARS];         /* compressed data length */
   char align[SHORT_CHARS];           /* scanline multiple: 8|16|32 */
   char unitsize[SHORT_CHARS];        /* bit size of image memory units */
   char sigbit;                       /* 0->sigbit first | 1->sigbit last */
   char byte_order;                   /* 0->highlow | 1->lowhigh*/
   char pix_offset[SHORT_CHARS];      /* pixel column offset */
   char whitepix[SHORT_CHARS];        /* intensity of white pixel */
   char issigned;                     /* 0->unsigned data | 1->signed data */
   char rm_cm;                        /* 0->row maj | 1->column maj */
   char tb_bt;                        /* 0->top2bottom | 1->bottom2top */
   char lr_rl;                        /* 0->left2right | 1->right2left */
   char parent[BUFSIZE];              /* parent image file */
   char par_x[SHORT_CHARS];           /* from x pixel in parent */
   char par_y[SHORT_CHARS];           /* from y pixel in parent */
}IHEAD;
```

Figure 21. IHead C language definition.

The IHead structure definition written in the C programming language is listed in Figure 21. Referencing the structure members listed in this figure, the first attribute field of IHead is the identification field, **id**. This field uniquely identifies the image file, typically by a file name. The attribute field, **created**, is the date on which the image was captured or digitized. The next three fields hold the image's pixel **width**, **height**, and **depth**. A binary image has a pixel depth of 1 whereas a grayscale image containing 256 possible shades of gray has a pixel depth of 8.

The attribute field, **density**, contains the scan resolution of the image; in the case of this database, 15.75ppm (400ppi). The next two fields deal with compression.

In the IHead format, images may be compressed with virtually any algorithm. The IHead header is always uncompressed, even if the image data is compressed. This enables header interpretation and manipulation without the overhead of decompression. The **compress** field is an integer flag which signifies which compression technique, if any, has been applied to the raster image data that follows the header. If the compression code is zero, then the image data is not compressed, and the data dimensions: **width**, **height**, and **depth**, are sufficient to load the image into main memory. However, if the compression code is nonzero, then the **complen** field must be used in addition to the image's pixel dimensions. In order to load the compressed image data into main memory, the value in **complen** is used to load the compressed block of data into main memory. Once the compressed image data has been loaded into memory, the appropriate decompression technique can be used to produce an image which has the pixel dimensions consistent with those stored in its header. The images in this database have been 2-dimensionally compressed using CCITT Group 4 compression [16].

The attribute field, **align**, stores the alignment boundary to which scan lines of pixels are padded. Pixel values of binary images are stored 8 pixels (or bits) to a byte. Most images, however, are not an even multiple of 8 pixels in width. In order to minimize the overhead of ending a previous scan line and beginning the next scan line within the same byte, a number of padded pixels are provided in order to extend the previous scan line to an even byte boundary. Some digitizers extend this padding of pixels out to an even multiple of 8 pixels, other digitizers extend this padding of pixels out to an even multiple of 16 pixels. This field stores the image's pixel alignment value used in padding out the ends of raster scan lines.

The next three attribute fields identify binary interchanging issues among heterogeneous computer architectures and displays. The **unitsize** field specifies how many contiguous pixel values are bundled into a single unit by the digitizer. The **sigbit** field specifies the order in which bits of significance are stored within each unit; most significant bit first or least significant bit first. The last of these three fields is the **byte_order** field. If **unitsize** is a multiple of bytes, then this field specifies the order in which bytes occur within the unit. Given these three attributes, binary incompatibilities across computer hardware and binary format assumptions within application software can be identified and effectively dealt with.

The **pix_offset** attribute defines a pixel displacement from the left edge of the raster image data to where a particular image's significant image information begins. The **whitepix** attribute defines the value assigned to the color white. For example, the binary images in this database are black text on a white background and the value of the white pixels is 0. This field is particularly useful to image display routines. The **issigned** field is required to specify whether the units of an image are signed or unsigned. This attribute determines whether an image with a pixel depth of 8 should have pixel values interpreted in the range of -128 to +127, or 0 to 255. The orientation of the raster scan may also vary among different digitizers. The attribute field, **rm_cm**, specifies whether the digitizer captured the image in row-major order or column-major order. Whether the scan lines of an image were accumulated from top to bottom, or bottom to top, is specified by the field, **tb_bt**, and whether left to right, or right to left, is specified by the field, **rl_lr**.

The final attributes in IHead provide a single historical link from the current image to its parent image; the one from which the current image was derived or extracted. The **parent** field typically contains the full path name to the image from which the current image was extracted. The **par_x** and **par_y** fields contain the origin point (upper left hand corner pixel coordinate) from where the extraction took place from the parent image. These fields provide a historical thread through successive generations of images and subimages. If the image has no parent, these three fields contain a null string. The IHead image format contains the minimal amount of ancillary information required to successfully manage binary and gray scale images.

C source code modules are provided in this distribution to decompress, read, and write IHead images.

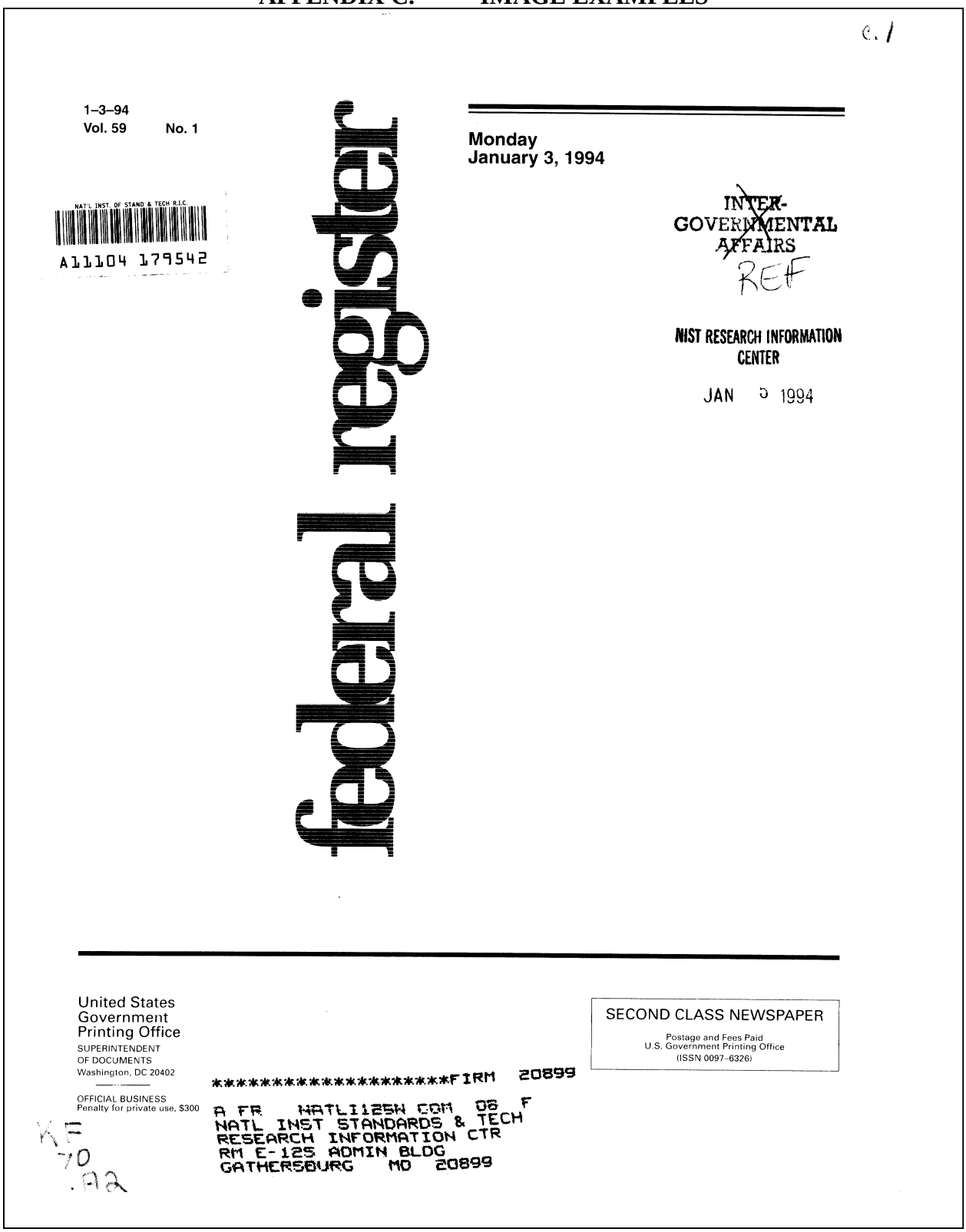

Figure 22. Hard cover page (data/01/03/book01/img/a0000000.pct).

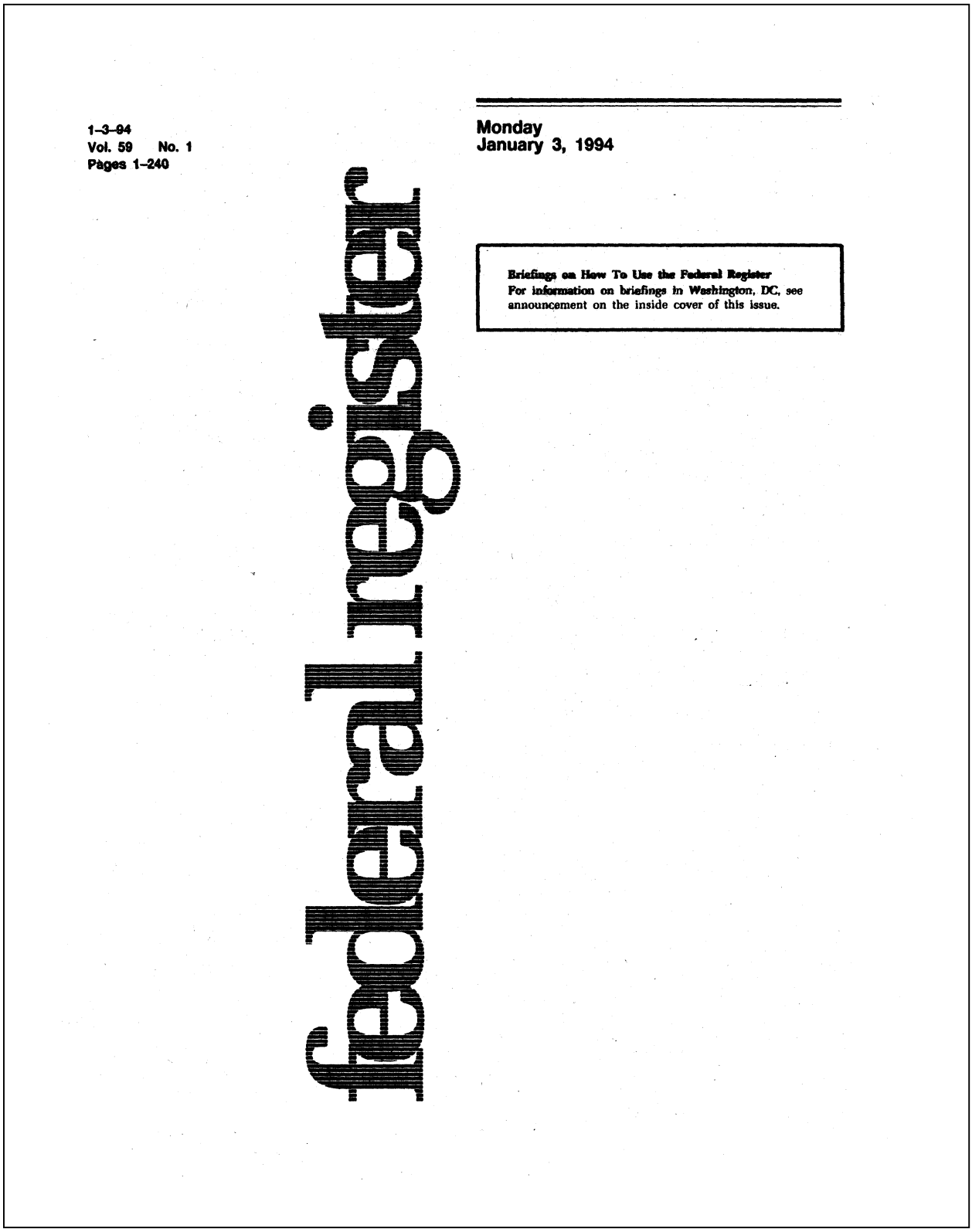

Figure 23. Inside soft cover page (data/01/03/book01/img/a0000001.pct).

Contents

Federal Register
Vol. 59, No. 1
Monday, January 3, 1994

Figure 24. Prefix content page (data/01/03/book01/img/a0000003.pct).

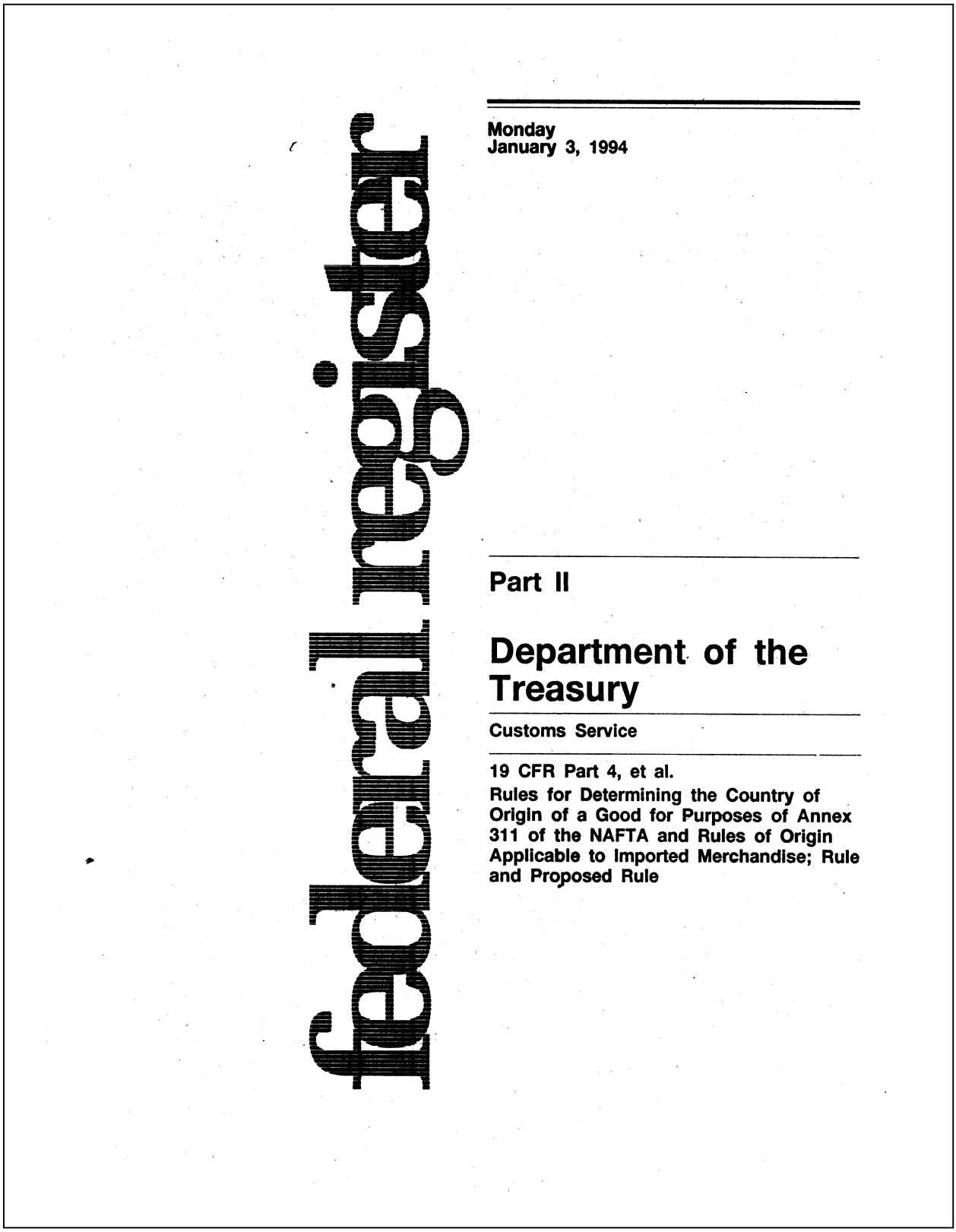

Figure 25. Section page (data/01/03/book01/img/b0000109.pct).

—Chairman's Report
—Committee Reports
—Wrap-up/Annual Schedule of Committee Meetings.

It is imperative that the meeting be held on this date to accommodate the scheduling priorities of the key participants.

Dated December 28, 1993.

Timothy M. Sullivan,

Advisory Committee Management Officer.

[FR Doc 93-32068 Filed 12-30-93; 8:45 am]

BILLING CODE 7510-01-M

NATIONAL FOUNDATION ON THE ARTS AND THE HUMANITIES

Challenge and Advancement Advisory Panel; Meeting

Pursuant to section 10(a)(2) of the Federal Advisory Committee Act (Public Law 92-463), as amended, notice is hereby given that a meeting of the Challenge and Advancement Advisory Panel (Literature Section) to the National Council on the Arts will be held on January 13, 1994 from 9 a.m. to 5:30 p.m. This meeting will be held in room 714, at the Nancy Hanks Center, 1100 Pennsylvania Avenue, NW., Washington, DC 20506.

A portion of this meeting will be open to the public from 3 p.m. to 5:30 p.m. for a policy discussion.

The remaining portion of this meeting from 9 p.m. to 3 p.m. is for the purpose of panel review, discussion, evaluation, and recommendation on applications for financial assistance under the National Foundation on the Arts and the Humanities Act of 1965, as amended, including information given in confidence to the agency by grant applicants. In accordance with the determination of the Chairman of November 24, 1992, this session will be closed to the public pursuant to subsection (c) (4), (6) (B) of section 552b of title 5, United States Code.

Any person amy observe meetings, or portions thereof, of advisory panels which are open to the public, and may be permitted to participate in the panel's discussions at the discretion of the panel chairman and with the approval of the full-time Federal employee in attendance.

If you need special accommodations due to a disability, please contact the Office of Special Constituencies, National Endowment for the Arts, 1100 Pennsylvania Avenue, NW., Washington, DC 20506, 202/682-5532, TYY 202/682-5496, at least seven (7) days prior to the meeting.

Further information with reference to this meeting can be obtained from Ms. Yvonne Sabine, Committee Management Officer, National Endowment for the Arts, Washington, DC 20506, or call 202/682-5439.

Dated: December 27, 1993.

Yvonne M. Sabine,

Director, Office of Panel Operation, National Endowment for the Arts.

[FR Doc. 93-32050 Filed 12-30-93; 8:45 am]

BILLING CODE 7537-01-M

NATIONAL LABOR RELATIONS BOARD

Appointments of Individuals to Serve as Members of Performance Review Boards

5 U.S.C 4314(c)(4) requires that the appointments of individuals to serve as members of performance review boards be published in the **Federal Register** Therefore, in compliance with this requirement, notice is hereby given that the individuals whose names and position titles appear below have been appointed to serve as members of performance review boards in the National Labor Relations Board for the rating year beginning October 1, 1992 and ending September 30, 1993.

Name and Title

Robert E. Allen—Associate General Counsel, Advice
Frank V. Battle—Deputy Director of Administration
Harold J. Datz—Chief Counsel to Board Member
Yvonne T Dixon—Acting Deputy General Counsel
Frederick Freilicher—Chief Counsel to Board Member
John E. Higgins—Solicitor
Susan Holik—Chief Counsel to Board Member
Gloria Joseph—Director of Administration
Nicholas E. Karatinos—Acting Associate General Counsel, Enf. Lit.
Barry J. Kearney—Deputy Associate General Counsel, Advice
Joseph E. Moore—Deputy Executive Secretary
W. Garrett Stack—Associate General Counsel, Operations-Management
Elinor H. Stillman—Chief Counsel to the Chairman
Berton B. Subrin—Director, Office of Representation Appeals
John C. Truesdale—Executive Secretary

Dated: Washington, DC, November 8, 1993.

By Direction of the Board.

John C. Truesdale,

Executive Secretary.

[FR Doc. 93-32004 Filed 12-30-93; 8:45 am]

BILLING CODE 7545-01-M

NUCLEAR REGULATORY COMMISSION

[Docket No. 50-261]

Carolina Power & Light Co.; H.B. Robinson Steam Electric Plant, Unit No. 2; Environmental Assessment and Finding of No Significant Impact

The U.S. Nuclear Regulatory Commission is considering issuance of a scheduler exemption from the requirements of 10 CFR part 50, Appendix E, to Carolina Power & Light Company (CP&L or the licensee), for H.B. Robinson Steam Electric Plant, Unit No. 2 (HBR), located in Darlington County, South Carolina.

Environmental Assessment

Identification of the Proposed Action

The proposed exemption would allow temporary relief from the requirements of 10 CFR part 50, Appendix E, Section IV.F.2., for the licensee at HBR to annually exercise its emergency plan. By letter dated November 21, 1993, as supplemented November 22, 1993, the licensee requested an exemption from the requirements of 10 CFR part 50 Appendix E, to conduct an annual exercise of the HBR Radiological Emergency Plan in 1993. The licensee had planned to conduct a full-participation exercise involving the State of South Carolina and local response organizations on November 30 1993. The licensee requested that an exemption be granted for the conduct of the onsite portion of the exercise because the licensee would not have sufficient staff to conduct a meaningful exercise of the HBR Emergency Plan due to resource constraints caused by an unscheduled outage to investigate and address core design issues. This proposed delay will prevent HBR from meeting the requirement to conduct an annual exercise of the HBR Emergency Plan. However, the licensee proposed that the offsite portion of the exercise involving the State of South Carolina and local governmental authorities be conducted as scheduled on November 30, 1993. The licensee requested a delay of the onsite portion of the 1993 annual exercise from November 30, 1993, to the week of March 21, 1994. The request to move the exercise date was originated by the licensee because they would be unable to participate on November 30, 1993, as a result of their involvement at that time in addressing safety issues identified during startup after refueling outage 15. The reactor was ascending to full power operation after a refueling outage when low power physics testing revealed an improper configuration

Figure 26. Body detail page (data/01/03/book01/img/b0000100.pct).

CFR CHECKLIST

This checklist, prepared by the Office of the Federal Register, is published weekly. It is arranged in the order of CFR titles, stock numbers, prices, and revision dates.

An asterisk (*) precedes each entry that has been issued since last week and which is now available for sale at the Government Printing Office.

A checklist of current CFR volumes comprising a complete CFR set, also appears in the latest issue of the LSA (List of CFR Sections Affected), which is revised monthly.

The annual rate for subscription to all revised volumes is $829.00 domestic, $207.25 additional for foreign mailing.

Mail orders to the Superintendent of Documents, Attn: New Orders, P.O. Box 371954, Pittsburgh, PA 15250–7954. All orders must be accompanied by remittance (check, money order, GPO Deposit Account, VISA, or Master Card). Charge orders may be telephoned to the GPO Order Desk, Monday through Friday, at **(202) 783–3238** from 8:00 a.m. to 4:00 p.m. eastern time, or FAX your charge orders to **(202) 512-2233.**

Title	Stock Number	Price	Revision Date
1, 2 (2 Reserved)	(869–019–00001–1)	$15.00	Jan. 1, 1993
3 (1992 Compilation and Parts 100 and 101)	(869–019–00002–0)	17.00	[1] Jan. 1, 1993
4	(869–019–00003–8)	5.50	Jan. 1, 1993
5 Parts:			
1–699	(869–019–00004–6)	21.00	Jan. 1, 1993
700–1199	(869–019–00005–4)	17.00	Jan. 1, 1993
1200–End, 6 (6 Reserved)	(869–019–00006–2)	21.00	Jan. 1, 1993
7 Parts:			
0–26	(869–019–00007–1)	20.00	Jan. 1, 1993
27–45	(869–019–00008–9)	13.00	Jan. 1, 1993
46–51	(869–019–00009–7)	20.00	Jan. 1, 1993
52	(869–019–00010–1)	28.00	Jan. 1, 1993
53–209	(869–019–00011–9)	21.00	Jan. 1, 1993
210–299	(869–019–00012–7)	30.00	Jan. 1, 1993
300–399	(869–019–00013–5)	15.00	Jan. 1, 1993
400–699	(869–019–00014–3)	17.00	Jan. 1, 1993
700–899	(869–019–00015–1)	21.00	Jan. 1, 1993
900–999	(869–019–00016–0)	33.00	Jan. 1, 1993
1000–1059	(869–019–00017–8)	20.00	Jan. 1, 1993
1060–1119	(869–019–00018–6)	13.00	Jan. 1, 1993
1120–1199	(869–019–00019–4)	11.00	Jan. 1, 1993
1200–1499	(869–019–00020–8)	27.00	Jan. 1, 1993
1500–1899	(869–019–00021–6)	17.00	Jan. 1, 1993
1900–1939	(869–019–00022–4)	13.00	Jan. 1, 1993
1940–1949	(869–019–00023–2)	27.00	Jan. 1, 1993
1950–1999	(869–019–00024–1)	32.00	Jan. 1, 1993
2000–End	(869–019–00025–9)	12.00	Jan. 1, 1993
8	(869–019–00026–7)	20.00	Jan. 1, 1993
9 Parts:			
1–199	(869–019–00027–5)	27.00	Jan. 1, 1993
200–End	(869–019–00028–3)	21.00	Jan. 1, 1993
10 Parts:			
0–50	(869–019–00029–1)	29.00	Jan. 1, 1993
51–199	(869–019–00030–5)	21.00	Jan. 1, 1993
200–399	(869–019–00031–3)	15.00	Jan. 1, 1993
400–499	(869–019–00032–1)	20.00	Jan. 1, 1993
500–End	(869–019–00033–0)	33.00	Jan. 1, 1993
11	(869–019–00034–8)	13.00	Jan. 1, 1993
12 Parts:			
1–199	(869–019–00035–6)	11.00	Jan. 1, 1993
200–219	(869–019–00036–4)	15.00	Jan. 1, 1993
220–299	(869–019–00037–2)	26.00	Jan. 1, 1993
300–499	(869–019–00038–1)	21.00	Jan. 1, 1993
500–599	(869–019–00039–9)	19.00	Jan. 1, 1993
600–End	(869–019–00040–2)	28.00	Jan. 1, 1993
13	(869–019–00041–1)	28.00	Jan. 1, 1993
14 Parts:			
1–59	(869–019–00042–9)	29.00	Jan. 1, 1993
60–139	(869–019–00043–7)	26.00	Jan. 1, 1993
140–199	(869–019–00044–5)	12.00	Jan. 1, 1993
200–1199	(869–019–00045–3)	22.00	Jan. 1, 1993
1200–End	(869–019–00046–1)	16.00	Jan. 1, 1993
15 Parts:			
0–299	(869–019–00047–0)	14.00	Jan. 1, 1993
300–799	(869–019–00048–8)	25.00	Jan. 1, 1993
800–End	(869–019–00049–6)	19.00	Jan. 1, 1993
16 Parts:			
0–149	(869–019–00050–0)	7.00	Jan. 1, 1993
150–999	(869–019–00051–8)	17.00	Jan. 1, 1993
1000–End	(869–019–00052–6)	24.00	Jan. 1, 1993
17 Parts:			
1–199	(869–019–00054–2)	18.00	Apr. 1, 1993
200–239	(869–019–00055–1)	23.00	June 1, 1993
240–End	(869–019–00056–9)	30.00	June 1, 1993
18 Parts:			
1–149	(869–019–00057–7)	16.00	Apr. 1, 1993
150–279	(869–019–00058–5)	19.00	Apr. 1, 1993
280–399	(869–019–00059–3)	15.00	Apr. 1, 1993
400–End	(869–019–00060–7)	10.00	Apr. 1, 1993
19 Parts:			
1–199	(869–019–00061–5)	35.00	Apr. 1, 1993
200–End	(869–019–00062–3)	11.00	Apr. 1, 1993
20 Parts:			
1–399	(869–019–00063–1)	19.00	Apr. 1, 1993
400–499	(869–019–00064–0)	31.00	Apr. 1, 1993
500–End	(869–019–00065–8)	30.00	Apr. 1, 1993
21 Parts:			
1–99	(869–019–00066–6)	15.00	Apr. 1, 1993
100–169	(869–019–00067–4)	21.00	Apr. 1, 1993
170–199	(869–019–00068–2)	20.00	Apr. 1, 1993
200–299	(869–019–00069–1)	6.00	Apr. 1, 1993
300–499	(869–019–00070–4)	34.00	Apr. 1, 1993
500–599	(869–019–00071–2)	21.00	Apr. 1, 1993
600–799	(869–019–00072–1)	8.00	Apr. 1, 1993
800–1299	(869–019–00073–9)	22.00	Apr. 1, 1993
1300–End	(869–019–00074–7)	12.00	Apr. 1, 1993
22 Parts:			
1–299	(869–019–00075–5)	30.00	Apr. 1, 1993
300–End	(869–019–00076–3)	22.00	Apr. 1, 1993
23	(869–019–00077–1)	21.00	Apr. 1, 1993
24 Parts:			
0–199	(869–019–00078–0)	38.00	Apr. 1, 1993
200–499	(869–019–00079–8)	36.00	Apr. 1, 1993
500–699	(869–019–00080–1)	17.00	Apr. 1, 1993
700–1699	(869–019–00081–0)	39.00	Apr. 1, 1993
1700–End	(869–019–00082–8)	15.00	Apr. 1, 1993
25	(869–019–00083–6)	31.00	Apr. 1, 1993
26 Parts:			
§§ 1.0–1–1.60	(869–019–00084–4)	21.00	Apr. 1, 1993
§§ 1.61–1.169	(869–019–00085–2)	37.00	Apr. 1, 1993
§§ 1.170–1.300	(869–019–00086–1)	23.00	Apr. 1, 1993
§§ 1.301–1.400	(869–019–00087–9)	21.00	Apr. 1, 1993
§§ 1.401–1.440	(869–019–00088–7)	31.00	Apr. 1, 1993
§§ 1.441–1.500	(869–019–00089–5)	23.00	Apr. 1, 1993
§§ 1.501–1.640	(869–019–00090–9)	20.00	Apr. 1, 1993
§§ 1.641–1.850	(869–019–00091–7)	24.00	Apr. 1, 1993
§§ 1.851–1.907	(869–019–00092–5)	27.00	Apr. 1, 1993
§§ 1.908–1.1000	(869–019–00093–3)	26.00	Apr. 1, 1993
§§ 1.1001–1.1400	(869–019–00094–1)	22.00	Apr. 1, 1993
§§ 1.1401–End	(869–019–00095–0)	31.00	Apr. 1, 1993
2–29	(869–019–00096–8)	23.00	Apr. 1, 1993
30–39	(869–019–00097–6)	18.00	Apr. 1, 1993
40–49	(869–019–00098–4)	13.00	Apr. 1, 1993
50–299	(869–019–00099–2)	13.00	Apr. 1, 1993
300–499	(869–017–00100–0)	23.00	Apr. 1, 1993
500–599	(869–019–00101–8)	6.00	[4] Apr. 1, 1990

Figure 27. Appendix page (data/01/03/book01/img/c0000002.pct).