Quality Assurance for Document Understanding Systems

Sherif Yacoub
Information Infrastructure Laboratory
HP Laboratories Palo Alto
HPL-2002-116
April 25th , 2002*

E-mail: sherif_yacoub@hp.com

quality assurance, document understanding, content remastering

Document understanding is a field that is concerned with semantic analysis of documents to extract human understandable information and codify it into machine-readable form. Document understanding systems provide means to automatically extract meaningful information from a raster image of a document. Those systems provide means to create information rich content that is usable in many end-user applications such as search and retrieval. To process a large volume of data, such as the collection of books and journals produced by a publisher, content understanding systems should run non-stop in an automated fashion and in an unattended operation mode. Ensuring the quality of the output of such system is a challenging task due to several factors including the unattended nature of the system and the mass amount of data (in terabytes) which could give rise to considerable number of exceptions. Automated quality assurance (QA) techniques are essential to the success of the operation of a large-scale document understanding system. In this paper, we propose QA techniques that are essentially needed for a document understanding system and their automation.

* Internal Accession Date Only
Approved for External Publication

Quality Assurance for Document Understanding Systems

Sherif Yacoub
Hewlett-Packard Laboratories
1501 Page Mill Rd., MS 1126
Palo Alto, CA 94304
sherif_yacoub@hp.com

Abstract

Document understanding is a field that is concerned with semantic analysis of documents to extract human understandable information and codify it into machine-readable form. Document understanding systems provide means to automatically extract meaningful information from a raster image of a document. Those systems provide means to create information rich content that is usable in many end-user applications such as search and retrieval. To process a large volume of data, such as the collection of books and journals produced by a publisher, content understanding systems should run non-stop in an automated fashion and in an unattended operation mode. Ensuring the quality of the output of such system is a challenging task due to several factors including the unattended nature of the system and the mass amount of data (in terabytes) which could give rise to considerable number of exceptions. Automated quality assurance (QA) techniques are essential to the success of the operation of a large-scale document understanding system. In this paper, we propose QA techniques that are essentially needed for a document understanding system and their automation.

Keywords: quality assurance, document understanding, and content remastering.

1 DOCUMENT UNDERSTANDING: APPLICATIONS VS SYSTEMS

The design of many software systems often involves the manipulation and processing of digital content. For instance, the ability to deliver electronic services through internet-based delivery channels requires that printed material such as books, journals, newspaper, or magazines be converted into forms suitable for electronic distribution. This type of content manipulation often includes preprocessing, transformation from one format to another, extraction of metadata, and in many cases verification and validation of the resulting content.

Document understanding [1] is one form of content understanding in which a system analyzes documents including books, journals, magazines, newspapers, or enterprise corporate documents. A document understanding system often starts with a scanned image representation of a document and delivers an information rich representation together with useful semantic information about its content. For instance, input channels could feed the system with raster image format of document pages (such

as the TIFF or BMP formats); the output channel could deliver searchable Portable Document Format (PDF) or eXtensible Markup Language (XML) documents. One application of such system could be the remastering of digitized (image) documents and re-purposing the content to become the backend of an online web community. A practical example of a community that uses a backend of re-purposed material is the Cognitive Science Online Community [18].

A document understanding system has a large number of components that work together in one or several workflows to achieve the system-level functionality. These components include (for example) algorithms for generation of digitized text, extraction of layout, extraction of logical information, text understanding, corpus-wide meta-data extraction, and journal and chapter splitting. These algorithms are developed in-house, acquired commercially, or obtained off-the-shelf as freeware or shareware components. The system uses this collection of algorithms to provide various functionalities.

Currently, we can identify several commercial applications and research tools that provide document understanding services. For instance, the WISDOM++ application [4] is an intelligent document processing system that transform paper documents into digital format such as HTML/XML. The system has several components for OCR, segmentation, classification, layout analysis, and transformation into web-accessible format. Roussel *et.al.* in [10] identify the requirements for a document understanding system that emphasizes the user-user interaction, user-algorithm interaction, and algorithm-algorithm interaction. Other commercial products such as Abbyy FineReader or Adobe Acrobat Capture also provide applications for transforming a raster image document into a searchable structured format like PDF or XML.

Existing solutions are considered document understanding *applications* or *programs*; they are not document understanding *systems*. Applications are useful for end-users since they provide graphical user interfaces through which the user interacts with the system, accepts or rejects results, or resubmits the document for processing in various configurations. We call these solutions *standalone* applications. In a standalone application, failures can be tolerated manually since the user can restart the program or repeat the remastering process.

Standalone applications are not suitable for processing massive volume of data and large number of input pages. This is often the case for commercial publishing. For instance, a publisher might want to provide online web access to a collection of books and journals that belong to some domain through a web community or through the publisher's web site. For this purpose, this collection of books and journals must be converted from paper format into searchable digital formats such as PDF documents. Using standalone applications to convert the collection is not a feasible solution due to the enormous

manpower required to use these applications to convert each and every document page in the collection.

As an example, consider a publisher who wants to convert the out-of-print collection from paper format to digital searchable format for the purpose of providing a print-on-demand (POD) service of out-of-print items for his customers. The publisher has an out-of-print collection of 5000 books and journals. The document images produced from scanning this collection (which is a small size collection) will be around 15 Terabytes of data of approximately 1.5 Million pages. The estimated size of the output data in PDF format would be in the 200 Gigabytes range depending on the content of the pages. To process such a massive amount of data, we need an automated system that has several key features:

1) The system runs in an unattended operation mode. Human intervention with system should be minimized.

2) The system runs in a non-stop operation mode. In order to be able to process this massive amount of data in a reasonable timeframe, the system should run in a 24x7 operation mode.

3) The system operates on multiple documents at the same time using a cluster of processing nodes that collaborate together to process the corpus.

In the following section, we discuss an example of a system that we developed for this purpose; the system is called *Digital Content ReMastering*, DCRM system.

Ensuring the quality of the output of such system is a challenging task. The system analyzes a massive amount of data (millions of pages); hence the output may contain a considerable number of exceptions. It is unrealistic to manually check the quality of each and every output page in a collection of million pages due to the huge human effort required for the process. In addition, the system runs in an unattended mode, therefore we should minimize any manual effort to interact with the system. The cost of post-correction of the output is often big [10] since it might be difficult to manage the rework tasks and the heterogeneity in the exceptions. Therefore, automated means for quality assurance (QA) are inevitable. The system should possess automated means for detection of quality imperfections and automated correction techniques. This paper proposes quality assurance techniques to be used in document understanding systems and their automation. We first describe the DCRM system. Then, we discuss automated quality assurance techniques, which include: the automated integrity check of input data, the automated defect detection in the output, automated means for using alternative algorithms for producing high-quality output, and finally the visual quality assurance process for suspect pages.

2 THE DCRM SYSTEM

The *Digital Content ReMastering* (DCRM) system is a complex system that provides automated non-stop unattended document understanding operations. Complexity arises from the number of software and hardware components that are integrated together as a solution for document understanding. The system has components that implement algorithms for document understanding. The system also has components for managing the operation and providing important non-functional features such as reliability and fault tolerance. The complexity also arises from management of a large volume of input data as well as management of the output data. The data flow and control flow in such a system is complex and the architecture should provide means to manage this complexity and facilitate interaction between functional components. Figure 1 illustrates the hardware configuration of the DCRM system.

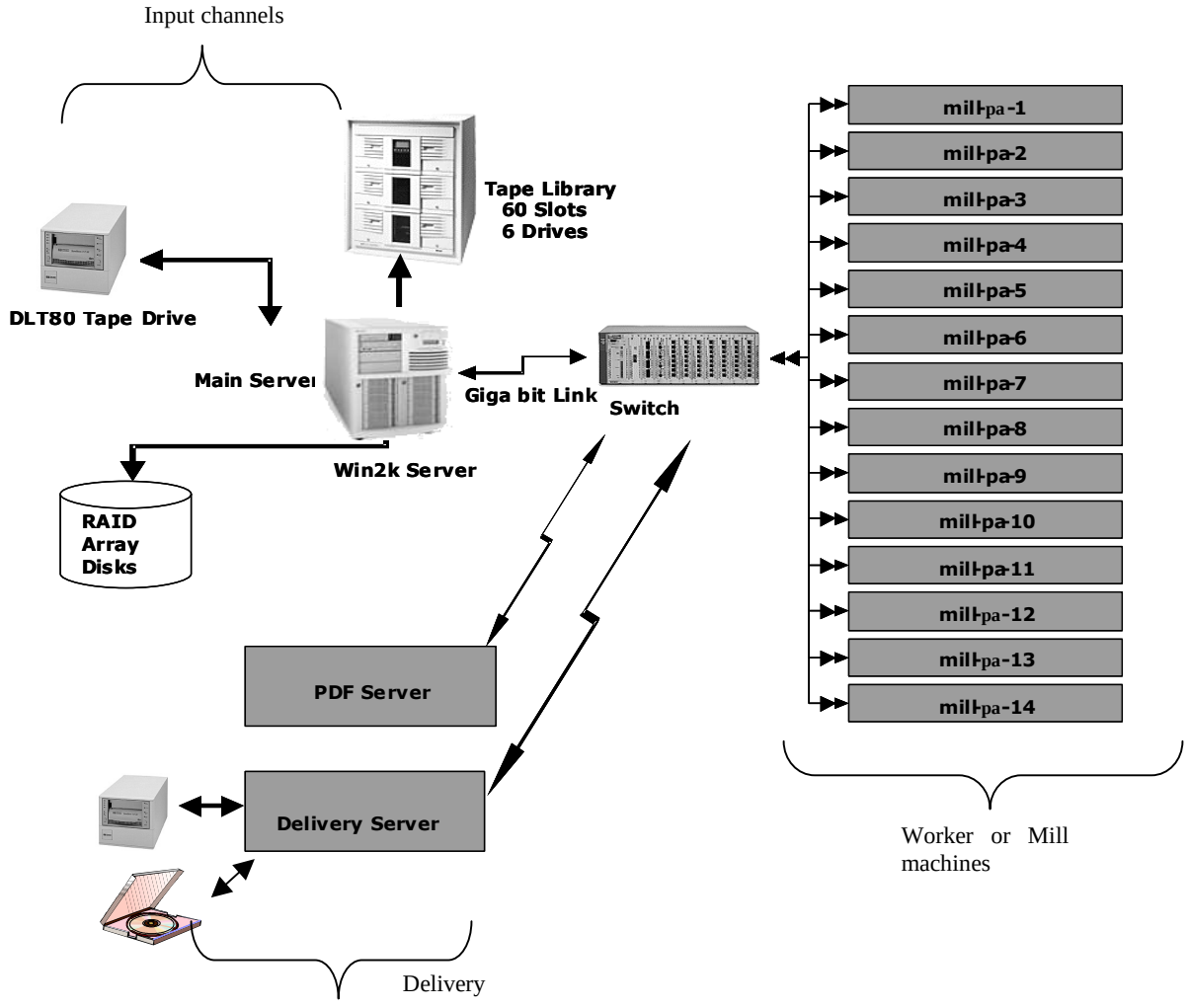

Figure 1 Hardware configuration of the DCRM system

The system is composed of a cluster of workstations that are connected together using a network switch. The workstations are called worker machines (or mill machines). They run components that perform the document understanding algorithms. The workstations are connected to a main server that hosts an external storage. The main server is responsible for managing the input data. There are several input channels to supply the system with scanned document images. In the above figure, we illustrate two input channels: a standalone tape drive and a tape library. Both input channels can read Digital Linear Tapes (DLTs). Each DLT contains up to 80GB of scanned document images, which constitute approximately 25-30 books or journals. The RAID storage is used to hold the input data while it is being processed by worker machines. The output data is stored on a separate server. A delivery server is used to collect output data and write them to a delivery media such as DLT tapes or Compact Disks (CDs).

The architecture of the system is based on component-based software engineering (CBSE) principles [11,17]. The architecture is a composition of components and scripts. Components perform the document understanding functions such as OCR, layout analysis, and logical structure analysis. Each component is a self-contained, fairly independent concrete realization that provides well-defined services and functionalities. Each component is wrapped using a script-wrapper that handles inputs, outputs and errors, and provides the context in which the component is executing. Components are glued together using scripts that manage the workflow execution.

The system has many components, for example:

- Access channel components: controllers to manage tape libraries and standalone tapes.
- Quality assurance components: such as the input integrity verifier, automated quality assurance of the output, and other components that we discuss in details in the following sections.
- Content understanding components: such as OCR engines, journal splitting components, page layout analysis components, logical analysis components, PDF concatenation component, journal article creation component, etc.
- Preprocessing components: such as bleed-through removal and auto-exposure components.
- Delivery channels components: such as DLT writers and CD writer components.
- Configuration utilities: such as tools to submit new titles for processing, checking the processing status of the corpus, and removing titles from processing channels.
- System components: such as reliable wire transfer, logger, watchdogs, process monitors, release synchronization, event notification, and performance analysis components.

To integrate these variety of components, the architecture is build using software framework principles [12] such as “don’t call us, we’ll call you”. In this architecture, none of the components has an explicit knowledge of the existence of other components in the system whether the same worker machine or another. Processes run in parallel over a distributed cluster of workstations. The data and control flows are separated in the architecture. The data flow is managed using the external RAID system. The control flow is managed using a set of communication channels where each process knows of an input channel to read tasks from and produce output in either a success channel that carries successful tasks or fail channels that carry failed tasks.

It is not the objective of this paper to describe the details of the DCRM system. Instead we focus on one important feature that is required in such system, the quality assurance techniques used to monitor the quality of the output data. In the following sections we discuss the quality assurance techniques used in the DCRM system.

3 INPUT QA

The DCRM system accepts scanned document images from several page acquisition channels. The paper documents are scanned, digitized, and stored as raster images. Any raster image format, such as TIFF format, can be used. When few document pages are to be scanned, the digitization procedure is simple and manageable by a single operator and probably using a desktop scanner. For large-scale industries, like commercial and academic publishers, digitization of a large corpus of books and journals is a complex operation that is usually uneconomical to manage in-house. Hence, the digitization operation is outsourced to a third party scanning vendor (for example, see Lason Inc [5]).

Figure 2 illustrates an example of the overall process involving the publisher, the scanning vendor, and the DCRM system. The documents are collected by the publisher and sent to a 3rd party-scanning vendor. The first step in the DCRM system is to load the scanned document images into the system. We have no control over the operation environment and the scanning process at the scanning vendor location. Therefore, quality assurance of the input document images is essential to ensure that the input complies with some predefined specifications. The system then applies the document understanding and remastering algorithms. The output is then checked for quality and delivered to the publisher.

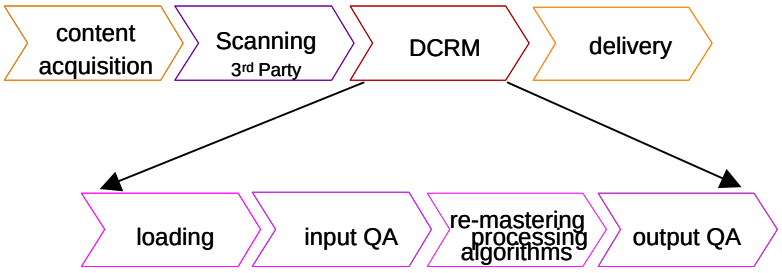

Figure 2 The Remastering process

A large number of pages are scanned (usually in the millions range, depending on the size of the publisher and the volume of the material to be digitized). Scanned page images are stored into electronic media that is then fed into the DCRM system. As part of the quality assurance process, the input data must be checked for integrity and quality. This upfront quality validation is important since a faulty input to the system might cause the system to fail or produce undesirable output.

The system has three quality assurance and integrity validations that are performed on the scanned document images before they are submitted for processing:

a) *Integrity of Data*. The system is capable of handling specific type of images with specific formats. The system accepts files that comply with type and format specifications. For instance, the specification of the scanned images could include the type and format of the document images (such as TIFF), the resolution of the scan (such as 400x400 dpp), compression schema (uncompressed, LZW compression, etc.), and color depth (gray scale, RGB color palate, etc.). The input quality assurance process validates the format as well as the specifications of the input files and rejects those files that do not comply with the specification. It also checks the integrity of the file and ensures that it is a valid data file.

b) *Consistency of Naming and Organization.* The scanned document pages should follow pre-defined naming convention that allows the system to automatically recognize files that belong to particular ISBNs or ISSNs. For instance, a simple directory structure could be implemented in which the ISBN number is used as the directory name. Names are chosen for individual page files such that a file name consists of two parts: the ISBN to which the page belongs and the page number. Several other naming schemas could be used. It is important to follow some specific schema such that the files and their order are automatically recognized by the system.

c) *Logical Verification.* A quality assurance process is required to check that all pages of the book have been scanned and no page has been dropped mistakenly. The system has a quality assurance component to verify that the scan was performed for all pages of the original book. This verifier process works on the scanned document images and detects the page number on every page. It then ensures that page numbers for a particular book are in sequence and consistent with the page numbers given to the files carrying the page image.

The input quality assurance processes are applied to each input page. Pages that pass the input QA test will be fed to the system for remastering. Failed pages are returned to the scanning vendor for correction or rescan.

4 AUTOMATED QA OF OUTPUT DATA

4.1 Purpose

In the heart of the DCRM system, a document-remastering engine is used to process and convert the input document images from a raster format (TIFF) to a searchable information-rich format (PDF). The process involves the application of several document-understanding algorithms. These algorithms include: *pre-processing algorithms* to prepare the document image for analysis such as skew detection and correction and noise removal algorithms; *region analysis algorithms* to detect regions in the page and identify their types; *layout analysis* to understand the document structure; *logical analysis* to understand the content of each region; *text extraction algorithms* such as OCR algorithms; and finally *re-purposing and reformatting* algorithms to produce various output formats.

Errors are inevitable. The document-remastering engines used in the DCRM system are general-purpose engines that can be used with various document types. Even though these engines have been tested thoroughly, errors will eventually occur. This is because the system is used to process a large number of document pages. Due to the volume and diversity of the input documents, we should expect that the algorithms might fail on some of the input pages and produce unacceptable output. Moreover, some of these algorithms are acquired off-the-shelf and hence we have no control over their accuracy. Therefore, we have to accept the fact that some errors will occur and we have to tolerate and recover from these errors.

Error rates. The DCRM system is used to analyze large volume of data. For such volume, even if the error rate is low *percentage*-wise it could be large *size*-wise. For example, a failure rate of 0.5% in a collection of two million pages is still 10,000 pages. Given that the average size of a book is about 400 pages, this 0.5% failure rate represents 25 books size-wise.

Type of errors. For QA purposes, we are not concerned with operation failures, which are related to the reliability of the components and the system. Instead, we are concerned here with *unacceptable output quality* that is due to errors produced by the document analysis engine. Examples of common errors that we find in the output of several document-understanding systems include: loss of content information, incorrect output text, or misclassification of region types.

Error detection. To assess the correctness of the output, a quality assurance person inspects the output and determines its correctness as compared to the input document. For example, the QA person compares the output PDF file to the input TIFF image file to determine whether any content information has been lost in the conversion process. For a system that analyzes a large volume of

documents in an unattended mode, such human intervention is undesirable due to performance constraints. Hence, automated support for quality check is a mandate.

Detectable errors. Based on our experience in QA for document understanding systems, we recognize that some of the errors could be automatically detected. For example, some of these errors include:

a) *Loss of content*. Some content might be lost during the remastering process. This could occur due to errors in the segmentation algorithm or errors in the region classification algorithm. For example, we find that a part of an image in the input TIFF file is misclassified as a noise-region instead of being classified as image-region. This is due to an error in the statistical analysis method used within the classification algorithm to make the distinction between image, text, noise, graph, or table regions. This is an unacceptable error specially when the missing region is a page number or part of an image as we found in some output pages.

b) *Defects in line drawings*. Some line and art drawings are not as sharp and clear (and sometimes are missing) in the output document as compared to the original raster document. This could be due to errors in the preprocessing algorithms or in the segmentation and region analysis algorithms.

c) *Skew errors*. The system detects skews in the input TIFF image and corrects them in the output PDF. When comparing the input and output pages we might find huge skew differences due to errors in the skew correction algorithm. These skew errors could make the output PDF unreadable.

4.2 The AutoQA Component

We have developed an Automated Quality Assurance (AutoQA) component as part of the DCRM system. The AutoQA is capable of detecting the inconsistencies between the output document page and the input raster image and the type of errors related to missing regions, blurred line art drawings, and skew problems.

Figure 1 illustrates the structure of the AutoQA component. The idea behind the AutoQA component is to use a closed-loop workflow to check the *content* of the output document as compared to the *content image* in the input document. The operation of the component is described as follows.

1) The digitized document page is called the Scanned Document Image (SDI). The automated quality assurance operation is performed for each of those pages as they are processed by the system.

2) The SDI document is analyzed and processed using a document-remastering engine that extracts relevant information using document analysis and understanding algorithms.

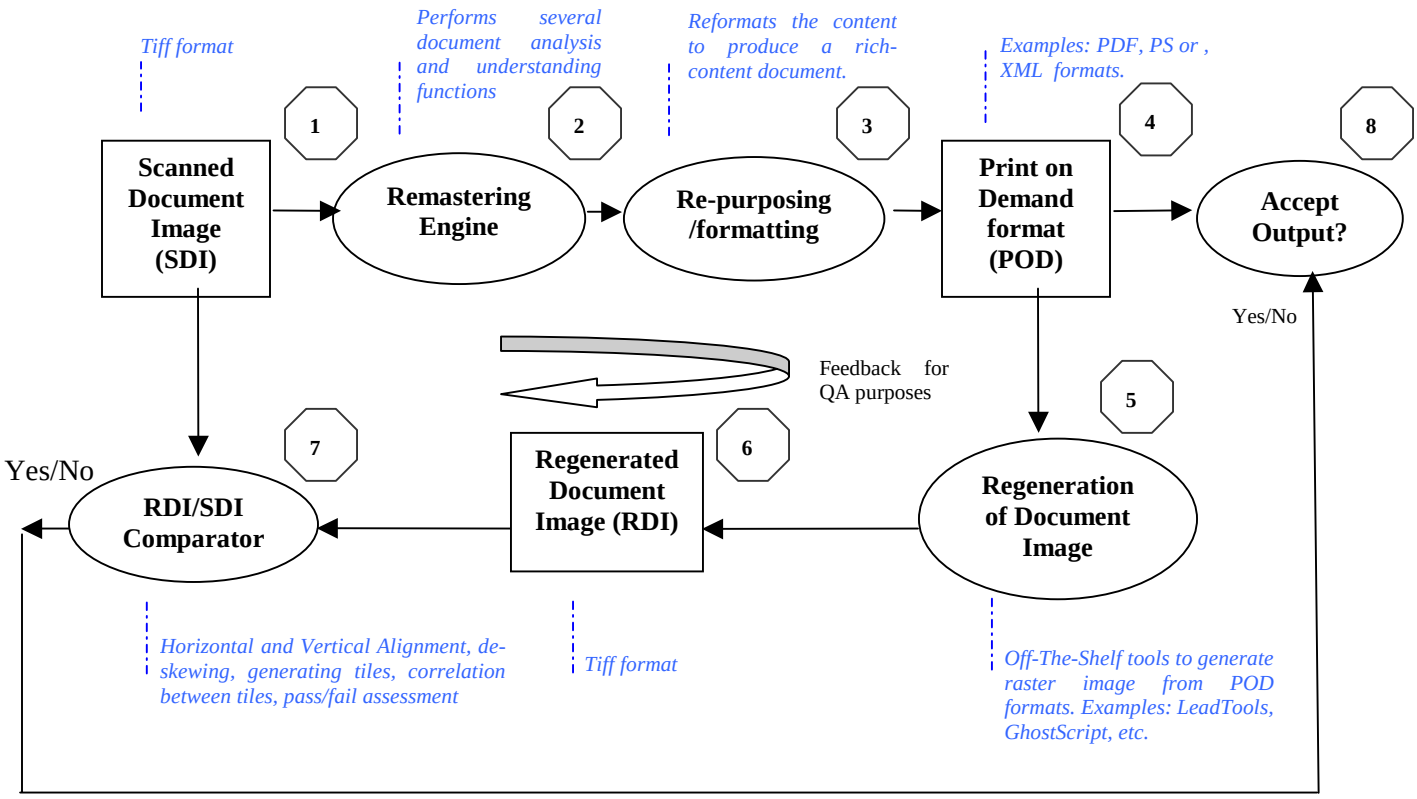

Figure 3 The automated quality assurance system using a feedback loop

3) Using the information produced by the document-remastering engine, a formatter (or re-purposing engine) is used to aggregate the segmented information to produce a viewable, readable, printable, and searchable "Print-on-Demand" (POD) quality output. Usually a Portable Document format PDF or Postscript formats are used.

4) The POD output is stored in a file called the POD document. This output is the desired output for a print on demand program, and so its quality is checked for errors that are possibly introduced during the content remastering operation.

5) A feedback loop is then started by using the output POD file to regenerate a raster image of the output (stepping back from information-rich format such as PDF to raster low information format such as TIFF). Standard file conversion components could be used in this operation such as the tools provided by Lead Technologies [2] or shareware software such as GhostScript [3].

6) As a result of previous step, a regenerated image is produced, which we call the Regenerated Document Image (RDI).

7) The original TIFF image (SDI) and the regenerated TIFF image (RDI) are then compared to determine if any content has been lost during the remastering and understanding process. Comparison is performed through correlation. If the remastering engine do not make mistakes in recognizing parts of the document, then the SDI and RDI images will be correlated. Correlation is performed using the following procedure:

   a) Skew is detected for both documents using any standard skew detection technique. The skew angles calculated are compared, and if the difference between these two angles is more than a threshold, then one image is rotated to line up with the other.

   b) The horizontal & vertical shifts (x and y offsets) between SDI and RDI are detected and the two images are horizontally & vertically aligned (one image is shifted to line up with the other).

   c) The SDI and the RDI images are down sampled to 100 dpi (from, for example, 400 or 200 ppi in the original files) and then partitioned into tiles. There are 20 tiles in each direction. For an 8.5 x 11 inch original, these tiles will be 0.425 inches x 0.55 inches.

   d) For each tile, the correlation (E(XY) - E(X)E(Y))/sqrt(Var(X)*Var(Y)) is calculated between the corresponding tiles in each image (SDI and RDI). Edge tiles are ignored from the correlation. If the variance is small, special care is used to avoid division by zero.

   e) If a tile has a correlation factor bigger than some threshold (say 0.7) then the tile is considered a pass tile, otherwise the tile fails the correlation. An overall correlation index can be computed by taking the mean correlation of all the tiles. The percentage of tiles that pass compared to the overall number of tiles can also be used as a metric.

8) Using the percentage of correlated tiles, the system makes a decision about whether the SDI and the RDI are considered correlated and thus accepts the output, or uncorrelated and thus rejects the output.

4.3 Results

Currently, every document understanding application or system involves either (a) manual evaluation of the output by a quality assurance person, or (b) manual correction of the output in case of erroneous documents. In “closing the loop” on the quality assurance by regenerating a TIFF file from the output, we are able to automatically detect defects in the output documents that are related to errors in

processing one or more of the document regions. For each of those defective documents the system generates a difference image that can be examined by a quality assurance person.

This AutoQA component is currently used in the quality assurance phase of the DCRM system. Because the AutoQA component generates weighted correlation data for all pages in the corpus, those pages requiring the most significant manual correction, if any, are identified in order, allowing the fastest improvement of overall corpus appearance.

As an example of the value-added of the AutoQA component, consider a real world example where the DCRM system is used to process one million pages of out-of-print books and journals for an academic publisher. Without the AutoQA component, the publisher has to make available the person to check the quality of the million output pages. Using the AutoQA component, 95% (950,000 pages) automatically passed quality assurance checks and only the remaining 5% are prioritized for manual quality checks using the correlation indices. This is a considerable saving in the human effort required for the quality assurance. The 5% pages that do not pass the AutoQA phase are not manually processed; other quality assurance techniques are used as discussed in the next sections to reduce the number of pages to be manually checked.

5 ALGORITHM COMBINATION FOR IMPROVED QUALITY

In a perfect world, the output quality of a document-understanding algorithm can be assessed by measuring the output against accurate results or facts. The process is called Ground Truthing and the facts about the output are called Ground Truth data. For example, ground truthing an OCR engine refers to the assessment of the correctness of the characters produced by the algorithm compared to the correct characters.

To be able to assess the correctness of the output of a content remastering engine, we should measure the results of the algorithm against the ground truth data. In the DCRM system, it is not acceptable to halt the system operation to check the quality of the output. Such system runs in an unattended mode and thus manual intervention is unacceptable. At runtime, there are no means to certify that the remastering engine is 100% correct because the absolute correct output data from applying the algorithm to specific input document is not available. Since we cannot be 100% sure about the results of an algorithm at run time. we compromise the 100% confidence and seek the highest possible correctness level instead. We can call techniques that try to increase the confidence in obtaining correct results runtime ground truthing.

The DCRM system is composed of several document-understanding algorithms. Algorithms are obtained commercially or developed in house. The system will usually possess several algorithms performing the same function but obtained from different sources. For example, it may have multiple engines each implementing an algorithm to perform OCR; one OCR engine is acquired commercially [6,7], another is obtained as a freeware [8], and another engine is developed internally by a document understanding team. Similarly, it may have multiple algorithms for region analysis, table of content (TOC) recognition, journal splitting into articles, and others. In such cases, the system has to automatically discern how to use those different (in implementation and technology) yet similar (in functionality) algorithms. The system has to decide which algorithm to use, whether to use only one algorithm or multiple algorithms for redundancy, and how to combine those algorithms. In such domain, there are several techniques to do algorithm combination for OCR applications [13-15].

Algorithm combination techniques can be used to improve the output quality using the results from competing algorithms. Competing algorithms are algorithms performing the same function (e.g. OCR), however, each uses a different technique, has different implementation, or has been acquired from different vendor. Therefore, combination increases the chance of obtaining a better quality output as compared to the output of a single algorithm.

As an additional quality assurance stage, the system is capable of perform runtime ground truthing using combination of competing algorithms and provides mechanisms for assessing the correctness of the output of an algorithm at runtime without manual intervention. There are several common techniques that are used by domain experts to combine algorithms. The DCRM system makes use of these common combination techniques. As an example, the DCRM system can use several content remastering engines and vote among the results produced by each. In this section, we limit our discussion to one technique that builds on top of the AutoQA component discussed in the previous section, we call it: the *TryAnother* technique.

Figure 4 illustrates the schematic diagram for the TryAnother technique. In this technique, we have alternative algorithms that can perform the same function. The idea is to simply try one engine at a time. The output is then measured using the AutoQA component (discussed in section 4). If the output is satisfactory according to the AutoQA acceptance criteria, then the output of that engine is selected as the final output and the rest of the engines are ignored. If the output does not meet the AutoQA quality criteria, the system selects another engine from the list and tries it. The system keeps repeating the process until a satisfactory output is reached.

When the *TryAnother* technique is used with the automated quality assurance technique, we find that the number of document pages that are submitted for manual QA is reduced significantly. Using the DCRM system and the same document collection discussed in section 4, we recall that 5% percentage of pages are submitted for manual QA due to errors produced by using only one document remastering engine. Using the *TryAnother* technique with eight alternative document-remastering engines, the percentage is reduced from 5% to 0.8% at the final output stages, which is a significant improvement given the size of the collection.

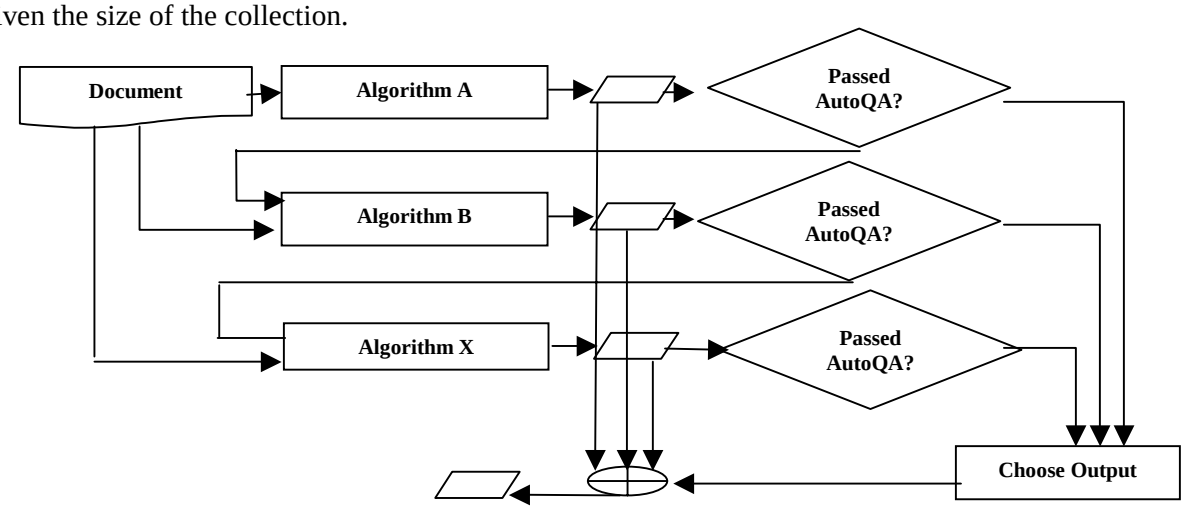

Figure 4 A schematic diagram for the *TryAnother* technique

6 VISUAL QA

6.1 Purpose

As we discuss earlier, it is not possible to obtain a 100% correct output for a collection of millions of document pages in an automated fashion. This is due to the large volume of data being processed by the system, which in practice would give rise to several exceptions. However, from the end-user perspective the data produced by the system has to be flaw-free. With regard to the quality assurance processes, the DCRM system utilizes the AutoQA component described in section 4 and the algorithm combination techniques described in section 5 to detect and tolerate processing errors. Despite the effectiveness of the combination of these two automated approaches, there will still be defective document pages that could not be fixed automatically. Therefore, human intervention is inevitable and a quality assurance person is needed to perform the quality assurance on the remaining document pages. We call this process the Visual Quality Assurance (VisualQA) process.

The VisualQA process is also required to examine another type of error that is not detected by the AutoQA process. Due to region analysis errors in the content remastering engine, some text regions

might get misclassified. As a result those regions will appear in the final output (PDF page) as gray-scale regions and not black and white text. A mechanism is required to check for possible gray-scale text in the output and present those suspect pages to the quality assurance person for verification.

6.2 The VisualQA System

The interaction of the quality assurance person with this massive number of document is not simple and is error-prone. To simplify the VisualQA process, we develop a web-based tool that facilitates the interaction of the QA person with the DCRM system. The VisualQA process is a four-phase process, which involves interaction between the VisualQA person and the system.

Phase 1: Generation and Notification

When the analysis of each page is completed, the system checks for possible gray-scale text defects in the output. It also checks whether or not the output fails the AutoQA test. In either of these two cases, the page is marked as a suspect page and is logged to a suspect file.

When the analysis of all pages of a book or a journal is completed, an HTML generator reads the suspect list for that particular book and generates an HTML file that contains an entry for each suspect page and a link to the output file (or a thumbnail version of the output is embedded in the HTML file). The page also contains form fields to accept input from the user as of what to do with the suspect page. An email is automatically sent to the VisualQA manager with the name of the book and the URL to the HTML of suspect pages.

Phase 2: Web-based Visual Quality Assurance

After receiving a notification message that a book is ready, the VisualQA manager uses a standard web browser to view the suspect PDF pages and check their quality. For each PDF page, the VisualQA manager is presented with several options to accept the page (no defects), reject the page (errors in the output page are irrecoverable), or submit the page for rework process that forces the system to reprocess using some specific options. The VisualQA manager checks suspect PDF pages and submits the form. The process is done asynchronously while the system is processing other pages.

Phase 3: Processing and Notification.

After the VisualQA form is submitted by the VisualQA manager, the system analyzes each entry and acts accordingly. Pages that pass the VisualQA are removed from the suspect list and are considered good pages. The raster images for pages that failed the VisualQA are kept aside for further manual rework. The document images for pages that are submitted for rework will be rescheduled in the

system for reprocessing with the options selected by the VisualQA manager. In all cases, the system notifies the VisualQA manager when processing the HTML form is completed.

Phase 4: Rework Cycle

Some of the suspect pages are select for rework by the VisualQA manager. There are several rework options. A very common rework cycle is to force the page to be processed as text-only page and hence all the regions identified by the remastering engine are classified as text regions. Those pages are rescheduled for processing and another VisualQA cycle is used to check the quality of the output.

6.3 Results

As an example of he volume managed by the VisualQA components, consider the application that we discussed in the previous sections. About 3% of the output pages were identified as suspect pages, whether because they failed the AutoQA criteria or because they possibly contain text regions that were misclassified as photo. The following figure illustrates the percentage of those pages the passed/failed the VisualQA and those submitted/passed/failed the text-only rework process. We find that out of the 3% suspect pages, 60% passed the VisualQA, 18% failed, and 22% are submitted for text-only rework process. Out of the pages submitted for rework, 85% passed the second round VisualQA and 15% failed. As a final result, only 0.12% of the whole collection is considered defective output.

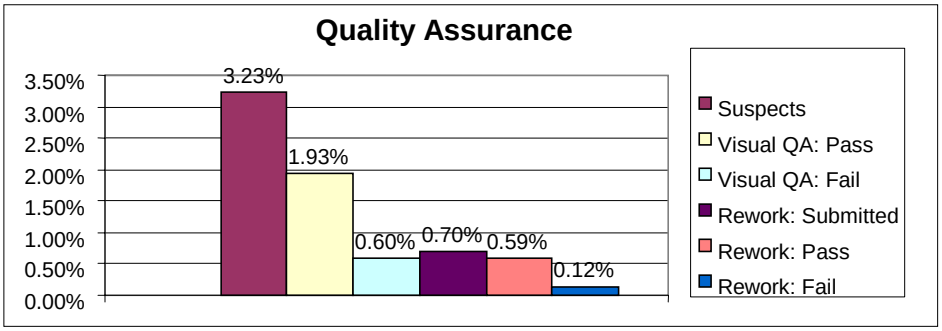

Figure 5 Results from the VisualQA procedures.

7 CONCLUSION

Quality assurance techniques are essential to the success of the operation of a large-scale document understanding system. Due to the large volume of data to be processed, the system should run in a non-stop unattended operation mode. We presented the DCRM system as an example of a document remastering system as compared to standalone document processing applications. In the DCRM system, automated means for quality assurance are used to minimize user interventions. We discussed automated means for quality assurance; namely: automated quality assurance of the input data,

automated quality assurance of the output data, and algorithm combination techniques for improving the output quality. We also illustrated a VisualQA feature in the DCRM system. Using the combination of these techniques, the DCRM system is able to: a) detect unsatisfactory quality of output due to errors in a content remastering engine, b) tolerate errors in one content remastering engine by using alternative engines, c) provide automated means to obtain high quality output, and d) significantly minimize the number of documents to be manually inspected.

REFERENCES

[1] A list of Document Understanding Research Groups. http://documents.cfar.umd.edu/cgi-bin/groups, 2002

[2] Lead Technologies, Inc. http://www.leadtools.com, 2002

[3] Ghostscript Freeware, http://www.ghostscript.com, 2002

[4] O. Altamura, F. Esposito, & D. Malerba. "WISDOM++: An Interactive and Adaptive Document Analysis System," *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, 366-369, IEEE Computer Society Press, Los Vaqueros, CA, 1999.

[5] Lason, The Information Management Company, Capture Service, http://www.lason.com/capture.htm , 2002

[6] ScanSoft OCR engine from Caere Inc., http://www.caere.com, 2002

[7] Abbyy OCR engine from Abbyy Inc., http://www.abbyy.com/, 2002

[8] A collection of OCR resources http://hosc.net/ocr/index.html, 2002

[9] O. Altamura, F. Esposito, and D. Malerba, "Transforming Paper Documents into XML Format with Wisdom++," *International Journal of Document Analysis and Recognition*, 3(2):175-198, 2000.

[10] Nicolas Roussel, Oliver Hitz, and Rolf Ingold, "Web-based Cooperative Document Understanding", Proceedings of the Sixth International Conference in Document Analysis and Recognition, ICDAR-6, Seattle Washington, September 2001, pp 368.

[11] Clemens Szyperski, "Component Software: Beyond Object-Oriented Programming", Addison Wesley, 1998.

[12] Mohamed Fayad and Douglas C. Schmidt, "Building Application Frameworks : Object-Oriented Foundations of Framework Design," John Wiley & Sons; 1999

[13] Xiaofan Lin, "Reliable OCR Solution for Digital Content Re-mastering," in the Proceedings of the *SPIE Conference on Document Recognition and Retrieval* IX, San Jose, CA 20-25 January 2002.

[14] S. Klink and T. Jäger, "MergeLayouts - Overcoming Faulty Segmentations by a Comprehensive Voting of Commercial OCR Devices", Proceedings of 5th *International Conference in Document Analysis and Recognition*, Bangalore, India, Aug 1999.

[15] L. Lam and C.Y. Suen, "A Theoretical Analysis of the Application of Majority Voting to Pattern Recognition", *Proceedings of IEEE*, pp. 418-420, 1994.

[16] Grady Booch, Ivar Jacobson, and Jim Rumbaugh, "The Unified Modeling Language User Guide," Addison Wesley 1998

[17] The 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction, Toronto, Canada, May 14-15, 2001

[18] The MIT Cognitive Science electronic community. http://cognet.mit.edu/