ABSTRACT
Many companies still operate critical business processes using paper-based forms, including customer surveys, inspections, contracts and invoices. Converting those handwritten forms to symbolic data is expensive and complicated. This paper presents an overview of the Image-Based Document Management (IBDM) system for analyzing handwritten forms without requiring conversion to symbolic data. Strokes captured in a questionnaire on a tablet are separated into fields that are then displayed in a spreadsheet. Rows represent documents while columns represent corresponding fields across all documents. IBDM allows a process owner to capture and analyze large collections of documents with minimal IT support. IBDM supports the creation of filters and queries on the data. IBDM also allows the user to request symbolic conversion of individual columns of data and permits the user to create custom views by reordering and sorting the columns. In other words, IBDM provides a “writing on paper” experience for the data collector and a web-based database experience for the analyst.

Categories and Subject Descriptors
1.7.5 Document Capture---Document analysis
J.1 ADMINISTRATIVE DATA PROCESSING---Business
H.3.5 Online Information Services---Web-based services
H.2.8 Database Applications---Image databases

General Terms

Keywords
Forms, handwriting, data capture, document management, tables.

1. INTRODUCTION
According to a 2012 survey\(^1\), even with the implementation of electronic health records, over 80% of healthcare organizations still relied on paper records and 10% of those organizations used paper as the primary method for recording information.

Converting to electronic records usually requires the development of complicated systems along with transcription of data collected on paper, both of which are expensive. We have created a system called Image-based Document Management (IBDM) that generically supports the capture of electronic strokes on forms displayed on a tablet and presents the results of the data collection using a spreadsheet view. The need for IT support is greatly reduced and in some cases eliminated, giving the owner of the business process the ability to design forms, collect data and analyze it independently.

1.1 Example Use
Home health care providers (over 33,000 agencies and 1.3 million workers in the U.S.\(^2\)) typically have dozens of forms that must be filled out and signed by patients. For example, in the U.S., home health care providers are required by law to get a privacy (HIPAA) notice signed by the patient. Even large agencies use paper forms for these activities and home care nurses deliver those forms by hand to the office.

Using IBDM, the patient writes on an image of the form, which is displayed on a commercially available stylus-based tablet. The strokes are captured and sent over a wireless network back to the server. An administrator can access and review the forms over the web using special client software. The administrator has access to each individual document as well as an overview of all the documents (Figure 1). The overview displays the captured strokes from each document in a row. Each column represents a specific field for a form. For instance, the administrator can confirm that all HIPAA forms have been signed by reviewing the signature column of the HIPAA form collection.

1.2 Related work
A number of approaches have been taken to capture data from paper sources but most of those efforts have focused on complete conversion from handwritten strokes to symbolic data. The system most like IBDM grew out of a Berkeley Ph.D. thesis by Chen [1]. Chen’s system uses probabilistic models for supporting automated data conversion combined with crowd sourcing. The system is built as a web service, accepting images of completed forms and providing an interface for human workers to either convert or verify conversions on the web and a dashboard for exploring the results. All conversion results are symbolic, although the original handwritten data is available. Hansen, et al. describe the FamilySearch\(^TM\) indexing effort [2] which used hundreds of thousands of volunteers to transcribe billions of records, including census, birth, death, and marriage records from around the world.

---


http://www.bls.gov/web/emp/siteseesebl1a.htm
A survey of techniques for handwriting recognition and conversion was published in 2000 [3]. A number of researchers have worked on the automatic recognition of handwritten data [4,5] or systems for separating handwritten data from a completed form [6]. The Anoto™ system captures strokes electronically using a unique pen containing a camera and processor but works only with specially printed paper. The Anoto™ pen is used in the ButterflyNet system developed at Stanford [7]. In ButterflyNet, the system goal was to integrate notes with digital data, including photos, audio recordings and video, but not necessarily to convert handwriting to a symbolic representation (although it supports transcription).

Another area with a focus on accurately converting handwriting to symbolic information is bank checks, 18.3 billion of which were processed in the US in 2012. Palacios et al. describe a system [8] that uses neural networks to convert digits written on checks and a context-aware post-processor to enhance the accuracy of the results.

2. SYSTEM ARCHITECTURE

IBDM consists of a web-based service along with three applications. The form preparation application (FPA) allows a user to specify the location of handwritten fields on a picture of a form. The form capture application (FCA) runs on a tablet and is used for capturing strokes on electronic forms and uploading the forms and strokes to the server. The third application is a data analysis client (DAC) and runs on a PC. The DAC is used by administrators to access and analyze processed form data.

A MySQL® database and CherryPy web server were installed on an Ubuntu® virtual machine. Python® software is used to provide web services and application logic. The open source Lipi toolkit3 is used for stroke classification and the Python® Boto library is used for integration with the Amazon Mechanical Turk® web service. A proprietary file format was adopted for capturing strokes and importing strokes into the server. Strokes can also be extracted from certain types of PDF files.

2.1 Workflow

Imagine that a retail company wants to conduct a survey with its customers. The company can scan in an existing blank paper survey or create an electronic document containing the questions leaving blank spaces for written responses.

The system administrator prepares an electronic version of the form by importing a PDF file of the survey into the form preparation application (FPA). Fields are marked on the form to indicate expected locations of handwritten responses. If desired, the administrator is able to add “widgets” to capture checkboxes, radio buttons, photos from the camera and text input from a soft keyboard. The document is pushed to the form capture application on the tablet.

A customer uses a stylus to fill out the form on the tablet. The customer can write anywhere on the form, including on photos captured by the camera. When complete, the form is pushed back to the form management server.

Each set of forms is stored as a “collection” on the server. The server extracts the strokes and symbolic data, including symbolic data and photos from the widgets and adds it to the collection. Additional metadata is captured, including stroke timing information and the form submission time and date.

Strokes are categorized and assigned to the predefined fields. Images of the strokes are created. Where widgets have been defined, images of the widgets are captured along with symbolic data. For instance, if a checkbox widget is checked, an image of

3 http://lipitk.sourceforge.net/
the check is created and a “true” symbolic value is stored in the database.

When the data analysis client (DAC) connects to the server and selects a collection, the client can request the images representing the stroke and widget data for all or for a subset of the forms. The DAC constructs a table view of the data. Each row contains the stroke and widget data from a single survey and each column represents the strokes from corresponding fields across all documents. For instance, a “Name” column can contain the handwritten name data from every HIPAA form in the system.

3. CREATING FORMS
Forms can be based on any scanned paper document or electronic form. Dense forms are difficult to fill out on a tablet because handwriting tends to be larger on an electronic display than on paper. A typical 10.1” LCD is only half the size of an 8.5 x 11 sheet of paper. Users may choose to redesign their form to fit on multiple pages since there is no need to save paper.

The IBDM system accepts multi-page PDF files as input. After opening the document with the FPA, the user places rectangles on each page of the form indicating a “field” where handwriting is expected. For instance, if a form has a name field, the field rectangle will surround the location where the name is expected to be written.

The FPA also provides widgets like radio buttons, check boxes, text fields and image fields. Text fields pop up a soft keyboard when activated and image fields allow a photo taken using the camera on the tablet to be inserted into the form.

4. ASSIGNING STROKES TO FIELDS
After the form is filled out and the IBDM server receives a document, all of the strokes from the document are identified and assigned either to a specific field or identified as non-field strokes.

To assign strokes to fields, the server looks at “runs” of stroke assignments and minimizes the number of transitions between fields in a given run of strokes as described below.

If the bounding box of a single stroke is completely contained in a single field rectangle as defined above, the stroke is assigned to that field. If a bounding box of a stroke overlaps multiple fields, those fields are listed as alternatives for the stroke. Once all of the strokes are categorized, “runs” of strokes are created—sets of strokes where two adjacent strokes overlap the same fields. Each run is analyzed to minimize transitions between fields and maximize the overlap between the bounding boxes of the strokes and the field.

Once all of the strokes are assigned to fields, a bounding box for the strokes assigned to a field is calculated and an image the size of that bounding box is created containing just the strokes. Since the stroke image is typically smaller than the field’s bounding box, the amount of data transferred to the client is reduced substantially. For one collection of documents, the original documents used 500 Mbytes of storage space, but the cropped images only required 5 Mbytes. We experimented using vector representations of strokes instead of semi-transparent PNG images and found almost no difference in storage requirements.

5. CREATING A TABLE
At launch, the data analysis client (DAC) requests the data for a collection, which is returned in JSON format, including information about every form and all of the strokes in those forms.

The client also requests the stroke image data for the collection. The stroke image data is cached locally to minimize bandwidth usage. The client constructs a table (Figure 1) that displays the stroke images and other data. The height of each row depends on the height of the stroke images and other data displayed in the row. The original width of each column is equal to the stroke image width.

6. DAC INTERFACE
The user interface of the DAC application appears somewhat like a spreadsheet but there are some important differences.

Like a spreadsheet, columns can be hidden and column widths and row heights adjusted. The column names can be modified and individual documents (rows) can be deleted.

There are a few operations that are significantly different than a spreadsheet or database. Each non-empty cell in the table can contain either symbolic or image data or both. The cells containing both are highlighted by the application as shown in the Figure 3. The “Display Priority” button lets the user choose whether they prefer to see symbolic data or image data if both are available.

The “Single Form – Edit” tab shown in Figure 4 allows the user to add or update symbolic data for a selected document.

The DAC automatically updates the table when new data is received from the server. The user can review and download a blank version of the form or any selected form in PDF format.
7. TRANSCRIPTION

Even though IBDM does not require transcription, it may be convenient to transcribe some portion of the data to make the collection of documents searchable. For instance, if HIPAA forms are filtered symbolically by last name, an administrator can quickly narrow the search for a specific patient’s form.

The Amazon Mechanical Turk® web service (AMT) is integrated with IBDM to support automated conversion by human workers. IBDM is a “requestor” of crowd-sourced jobs. A DAC user can indicate that a specific column should be automatically converted to symbolic data. When a server receives a document with strokes for that column, the stroke image is created and a Human Intelligence Task (HIT) is created on AMT. An example HIT is shown in Figure 5 for the zip code field of a survey. As images are converted, the data is fed back to the server. In our experience, most of our zip code HITs were completed within 60 seconds.

8. SORTING

In the data analysis client, a user can click on the column heading to initiate a sort. If the column has been supplemented with symbolic data, by default the column is sorted in alphabetic order. When cells only contain stroke images or contain a combination of strokes and data, more information is required from the user. Each cell can be in one of four states: empty (E), symbolic data (S), image/stroke data (I) or both symbolic and image data (B). The user selects a sort priority—which types of cells come first, E, S, or I. Cells that have both (B) are always lumped with the higher priority cells S or I.

Symbolic data can be sorted alphanumerically, alphabetically or numerically. Strokes and image data can be sorted by image metadata: width, height, image file size or number of pixels.

An example sort would be 1) symbolic first in alphabetic order, 2) image by classification and then 3) empty cells.

9. QUERIES

As in sorting, queries require special handling depending on the column type. In the lower right part of the client application window (Figure 1), the user can select a date filter, which is always applied directly to submission timestamp.

Additional column filters can be created based on symbolic or image information. For instance, it is possible to filter using a regular expression on symbolic data or by the existence of an image. For instance, the user may want to review all HIPAA forms that have no signature. Filters can be applied to any column by using the column filter interface not described here.

10. RESULTS

The full system, including the applications and server were completed early in 2013 and the system was used for an electronic visitor log at Ricoh Innovations. The administrator responsible for the visitor log created the form, marked the stroke regions, added an email widget and photo widget and deployed the form to several tablets. The DAC application was installed on the PCs of several administrators at RIC. During the 3 month deployment, 65 visitors signed in using the tablets. In a separate experiment, over 100 surveys were administered to gather handwriting samples and provide a sample database for experimentation.

11. CONCLUSION

We implemented and demonstrated a system that allows people to fill out forms on an electronic tablet in a manner that feels like working with a paper and pen with a web-based backend that supports review and analysis. The simple interface allowed a non-technical administrator to develop, deploy and query a complete handwriting-based visitor log system.

12. ACKNOWLEDGMENTS

FamilySearch is a trademark of Intellectual Reserve, Inc. Anoto is a registered trademark of Anoto. MySQL is a registered trademark of Oracle and/or its affiliates. Ubuntu is a registered trademark of Canonical Ltd. Python is a registered trademark of Python Software Foundation. Amazon Mechanical Turk is a registered trademark of Amazon Technologies, Inc. in the United States and/or other countries.

13. REFERENCES


