FormCracker: Interactive Web-based Form Filling

Laurent Denoue, John Adcock, Scott Carter, Patrick Chiu, Francine Chen
FX Palo Alto Laboratory, Inc.
3400 Hillview Avenue, Building 4
Palo Alto, CA 94304 USA
{denoue,adcock,carter,chiu,chen}@fxpal.com

ABSTRACT

Filling out document forms distributed by email or hosted on the Web is still problematic and usually requires a printer and scanner. Users commonly download and print forms, fill them out by hand, scan and email them. Even if the document is form-enabled (PDFs with FDF information), to read the file users still have to launch a separate application which may not be available, especially on mobile devices.

FormCracker simplifies this process by providing an interactive, fully web-based document viewer that lets users complete forms online. Document pages are rendered as images and presented in a simple HTML-based viewer. When a user clicks in a form-field, FormCracker identifies the correct form-field type using lightweight image processing and heuristics based on nearby text. Users can then seamlessly enter data in form-fields such as text boxes, check boxes, radio buttons, multiple text lines, and multiple single-box characters. FormCracker also provides useful auto-complete features based on the field type, for example a date picker, a drop-down menu for city names, state lists, and an auto-complete text box for first and last names. Finally, FormCracker allows users to save and print the completed document.

In summary, with FormCracker a user can efficiently complete and reuse any electronic form.

Categories and Subject Descriptors

I.7.5 [Document and Text Processing]: Document Capture - document analysis; H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces; H.3.3 [Information Systems]: Information Search and Retrieval

General Terms

Algorithms, Human Factors.

Keywords

Form filling, interactive, document processing, image processing.

1. INTRODUCTION

Filling out electronic form documents that have not been appropriately authored for that purpose can be a cumbersome process. Filling out a read-only PDF form typically requires the user to print the document, fill it out by hand, and then scan it back in. Alternatively, users can import the document into Acrobat (or other image and or document editing software) and carefully overlay text boxes and checkmarks over the right locations on the document pages.

Editable document formats (such as Word) can also be awkward to use for form-filling tasks. Users can open the document and edit to enter form data, but must be careful not to alter the original document content or layout in the process.

These problems can be mitigated by the appropriate use of explicit form fields in Word and PDF documents, but this is not commonly done. While many PDFs provided by government agencies (such as tax returns) are now form-enabled, many forms distributed by local businesses such as hotels, restaurants, schools, and sport associations are not. To quantify this observation, we downloaded 445 PDFs returned by a search on Google for the query: “ext:pdf registration form”. We checked that each document was a form, and tested if they were form-enabled by using the Java library iText[1] to detect the presence of FDF data. 75% (335 out of 445) were not form-enabled.

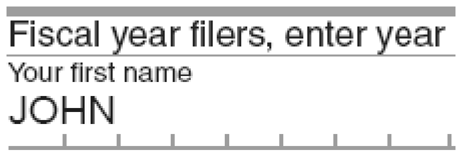

Fiscal year filers, enter year
Your first name
JOHN

Figure 1. Even FDF-enabled PDFs make it hard for user to correctly fill-in a series of single-character fields.

Finally, using even form-enabled PDFs as depicted in Figure 1 can be frustrating; characters do not appear neatly in each designated box. To enter one letter per box, the user must again carefully add spaces. Unfortunately, the Form Data Format (FDF) in this example form authorized a maximum of 26 characters, so after too many spaces, the user can no longer enter characters for his/her name.

FormCracker addresses these shortcomings with a web-based, plugin-free, document viewing and form-filling application. Each page of a document is converted into an image, and users simply select a location on a page where data needs to be entered, whether a text box, the first line of a multi-line field, or a box of a multiple single-character entry. The user experience resembles one of filling out a typical web-based form, or FDF-enabled form within Acrobat, but without the need for a browser plugin or external application, or even the need to download the original document at all.

Below we give an overview of the system and its use, detail the image processing and text analysis techniques used by FormCracker to detect form-fields, put the system in the context of related work, and describe ongoing work to improve the system.

2. SYSTEM DESCRIPTION

FormCracker is a web-based system that allows users to upload and fill out forms in Office Documents (including PPT, Doc, PDF). A bookmarklet and browser add-on also help users quickly view and fill out documents that are already online at publicly available URLs.

Upon receiving a document or its URL, FormCracker converts the document to PDF format using OpenOffice's PDF exporters (if necessary) and then uses XPDF to render each document page into an image (PNG or JPEG depending on its size).

Some steps in the FormCracker form-detection process require extracting the text bounding boxes of each page. This is accomplished using XPDF's pdf2html tool or Optical Character Recognition (OCR) if the source document is an image, such as a scanned document.

2.1 Form Field Recognition

There are two primary field types to be recognized: text fields, and selection fields. Text fields allow for typed input, while selection fields comprise checkboxes and radio buttons which have a binary on or off state.

2.1.1 Textfields and multiple-box characters

Upon receiving a coordinate (x,y) on a page, the server binarizes the corresponding page image using built-in PIL converters from RGB to binary mode. As shown in Figure 2, FormCracker then finds the first black pixel below and above the (x,y) location, and follows these left and right to determine the horizontal extent. If no upper baseline is found within a reasonable vertical offset (currently 40 pixels, but this may be adjusted based upon the size of the fonts in the document), only the bottom baseline is used and a fixed height is set for the text box.

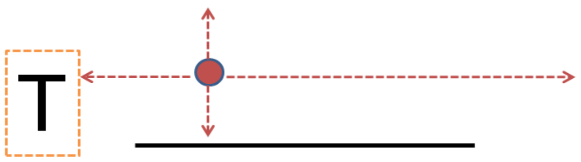


Figure 2. Searching for field boundaries from a user-selected point. Text with bounding box shown on the left, baseline shown on the bottom. Paths of search from the initial selection are shown in dotted line.

Similarly, FormCracker finds the foreground pixels on the left and right of the (x,y) location, and then determines the vertical extents of the form field, making possible the detection of multiple rectangular single-box characters shown in Figure 3.

FormCracker is not limited to black and white documents, and can handle forms with a uniform color background. FormCracker uses the color at the user-selected location and searches until it identifies a pixels in a different color in each of the directions.

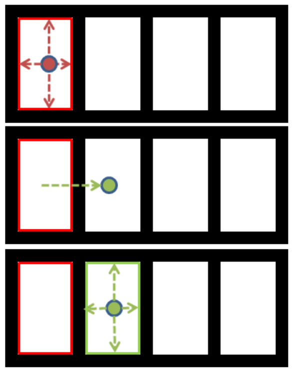
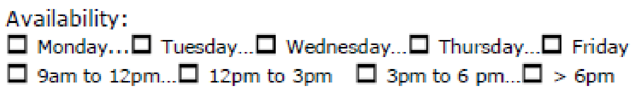


Figure 3. When a rectangular area is found, FormCracker tests for similarly sized boxes to the left and right of the first user-selected point. This allows it to detect the whole set of multiple-box character fields from a single user selection.

2.1.2 Checkboxes

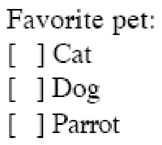
Checkboxes are detected when a box appears in isolation with an aspect ratio close to 1:1 and dimension within a threshold (Figure 4).



Availability:
☐ Monday... ☐ Tuesday... ☐ Wednesday... ☐ Thursday... ☐ Friday
☐ 9am to 12pm... ☐ 12pm to 3pm ☐ 3pm to 6 pm... ☐ > 6pm

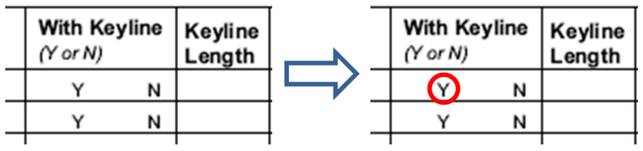
Figure 4. An example of checkboxes, distinguishable from text boxes by their aspect ratio and isolated appearance.

Text can also used to identify checkboxes. We found many examples of square brackets or parentheses used to form checkboxes. Using XPDF's pdf2html, FormCracker knows the location of each character on a page[1]. Common patterns, such as "( )", "[ ]", and "{ }", are then identified in the text.



Favorite pet:
[ ] Cat
[ ] Dog
[ ] Parrot

Figure 5. Common text patterns, such as "[ ]", used to represent checkboxes are recognized.



With Keyline (Y or N)		Keyline Length
Y	N	
Y	N	

With Keyline (Y or N)		Keyline Length
[✓ Y]	N	
Y	N	

Figure 6. Selection fields that use circling to indicate a choice. Clicking within the text box selects or deselects the text by overlaying a circle.

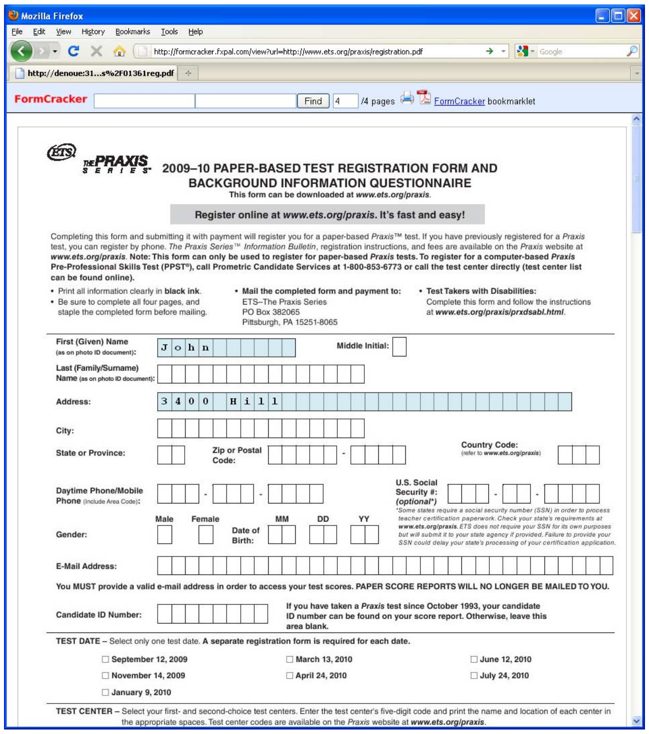
[1] If text is not available in the original document, Optical Character Recognition from Microsoft Office Document Imaging is performed to get character bounding boxes.

Another commonly used form input method is to provide a list of brief text choices for the user to choose between by circling their choice. See Figure 6. To accommodate this situation, when FormCracker detects a user selection within a text bounding box, this is interpreted as selecting or deselecting the chosen text. The server returns the bounds of the text as the form field extent and indicates the field type as a CIRCLEBOX.

2.2 Rendering form types and editing

After pre-processing the document to extract document images and text, the system presents an HTML view of each page image. When the user selects a point on the page, javascript determines the coordinates of the click and communicates it back to the server along with the document ID and page number.

The server (currently implemented in Python and Python Imaging Library) analyses the region surrounding the user selection and returns back to the client a form type (described in section 2.3) and its extent. The client creates an HTML DIV element over the page image and, if the element is a TEXTFIELD or MULTIPLE BOXED characters, individual sub-DIVs representing one character each are added as well. In case of a CHECKBOX, or CIRCLEBOX no further DIV element is created.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://formcracker.fxpal.com/view?url=http://www.ets.org/praxis/registration.pdf
Google
http://denoue:31...s%2F01361reg.pdf
FormCracker
Find
4
/4 pages
FormCracker bookmarklet
ETS
THE PRAXIS SERIES

2009–10 PAPER-BASED TEST REGISTRATION FORM AND BACKGROUND INFORMATION QUESTIONNAIRE

This form can be downloaded at *www.ets.org/praxis*.

Register online at *www.ets.org/praxis*. It's fast and easy!

Completing this form and submitting it with payment will register you for a paper-based *Praxis*™ test. If you have previously registered for a *Praxis* test, you can register by phone. *The Praxis Series*™ *Information Bulletin*, registration instructions, and fees are available on the *Praxis* website at ***www.ets.org/praxis***. **Note: This form can only be used to register for paper-based *Praxis* tests. To register for a computer-based *Praxis* Pre-Professional Skills Test (PPST®), call Prometric Candidate Services at 1-800-853-6773 or call the test center directly (test center list can be found online).**

- Print all information clearly in **black ink**.
- Be sure to complete all four pages, and staple the completed form before mailing.
- **Mail the completed form and payment to:** ETS–The Praxis Series, PO Box 382065, Pittsburgh, PA 15251-8065
- **Test Takers with Disabilities:** Complete this form and follow the instructions at ***www.ets.org/praxis/prxdsabl.html***.

First (Given) Name (as on photo ID document): J o h n **Middle Initial:**

Last (Family/Surname) Name (as on photo ID document):

Address: 3 4 0 0 H i l l

City:

State or Province: **Zip or Postal Code:** - **Country Code:** (refer to ***www.ets.org/praxis***)

Daytime Phone/Mobile Phone (Include Area Code)**:** - - **U.S. Social Security #: *(optional*)*** - -

Some states require a social security number (SSN) in order to process teacher certification paperwork. Check your state's requirements at **www.ets.org/praxis. ETS does not require your SSN for its own purposes but will submit it to your state agency if provided. Failure to provide your SSN could delay your state's processing of your certification application.*

Gender: **Male** **Female** **Date of Birth:** **MM** **DD** **YY**

E-Mail Address:

You MUST provide a valid e-mail address in order to access your test scores. PAPER SCORE REPORTS WILL NO LONGER BE MAILED TO YOU.

Candidate ID Number: **If you have taken a *Praxis* test since October 1993, your candidate ID number can be found on your score report. Otherwise, leave this area blank.**

TEST DATE – Select only one test date. **A separate registration form is required for each date.**

☐ **September 12, 2009** ☐ **March 13, 2010** ☐ **June 12, 2010**
☐ **November 14, 2009** ☐ **April 24, 2010** ☐ **July 24, 2010**
☐ **January 9, 2010**

TEST CENTER – Select your first- and second-choice test centers. Enter the test center's five-digit code and print the name and location of each center in the appropriate spaces. Test center codes are available on the *Praxis* website at ***www.ets.org/praxis***.

Figure 7. FormCracker lets users click inside any character box and start typing. FormCracker automatically detects multiple adjacent single character boxes. The cursor is automatically positioned in the first empty box.

Sub-DIV elements are chained: when the user clicks one, the javascript client determines which DIV is the first, e.g. in a TEXTFIELD, and makes it the current DIV. When the user types a key, the client inserts the corresponding character into this current DIV and advances to the next DIV in the group. If the user clicks inside a TEXTFIELD where characters have already been entered, the cursor is placed in the first empty DIV in the group.

A default font in a typesize similar to adjacent text is selected for the entered characters. If the user prefers a different typesize, it can be modified using the same key sequences used for Office documents: CTRL+SHIFT+> for increasing the type size and CTRL+SHIFT+< for decreasing the type size.

When the user clicks on a DIV with a CHECKBOX type, the client adds (or removes) the image of a check in the inner HTML of that DIV element. The same technique is used to toggle CIRCLEBOX buttons, but with a circle image rather than a check.

When a checkbox is found in the page, the server also identifies and returns to the client the locations of other checkboxes in the page with similar appearance. The client uses this information to chain CHECKBOXes, allowing users to tab through them. BACKSPACE and DELETE keys are also supported, using the chain of DIV elements, providing users with a familiar typing and editing experience.

2.3 Form completion

To mimic the experience of filling out online HTML-based forms, FormCracker can also provide completion support for certain form types. For example, if the server detects the text string "date" next to a FORMFIELD, it tells the client to display a tooltip with the current date below the corresponding DIV; and when the user presses ENTER, the TEXTFIELD is automatically populated with the date. Other kinds of completion are supported, such as date pickers, and lists of countries or states.

FormCracker can also restrict the type of content allowed to be typed (e.g. alpha or numeric): if a TEXTFIELD is preceded on the page by the $ sign or immediately followed by %, then only digits and a decimal point are allowed.

3. RELATED WORK

Online services exist [3][4] which allow the user to augment an existing PDF document with text and other objects including form fields, but they require the user to choose the type of field to insert and to manually size and position the field on the original document.

Adobe provides a tool to support PDF document authors in adding form fields to existing PDF documents, but Acrobat's "Automatic Form Recognition" [2] functionality still requires manual authoring and correction to accurately create a form-enabled PDF, and was designed as a tool for form publishers to augment their existing documents, not as a tool for end-users to quickly fill out a form.

Bagley et al. [5] describe a system for making any document image editable by reconstructing the positions of all the contained characters, but does not help with form filling.

Other systems [1][6] address the problem of collecting data from scanned forms by matching against a library of known form layouts. The focus is here is on scan-based data entry rather than the simple ad-hoc form filling afforded by FormCracker.

Most closely related is a system [8] which automatically finds the boundaries of a form field starting from a user selected location in a document image. This work differs significantly from

FormCracker in that it requires the user to choose the type of field (text or checkbox) before selection. Also significantly the fact that it does not link multiple fields together (such as groups of single character boxes, and check boxes with similar appearance) for enhanced document navigation.

4. CONCLUSION AND FUTURE WORK

With FormCracker we have endeavored to design a system to simplify the electronic form filling task which often proves to be needlessly complicated due to the difficulty of authoring a proper electronic form. By bootstrapping the process by having the user identify the position of the desired form fields on the page, lightweight image processing to bound and classify those fields is enabled. We are investigating enhancements to the system to further improve the user experience.

4.1 Metadata

We are extending FormCracker so that it detects and uses existing form data within FDF-enabled PDFs because we suspect users will find the online nature of FormCracker useful even for FDF enhanced PDF files.

4.2 Interpreting Document Semantics

Approve
Deny

Figure 8. FormCracker uses lightweight lexical analysis of nearby text to distinguish radio buttons from checkboxes.

Nearby text can be used to disambiguate between a checkbox (multiple selections allowed) and a radio button (only one selection allowed within a group). If mutually exclusive or antonymic terms, such as “approve” and “deny” in Figure 8, are associated with checkbox candidates, then the group of selection boxes can be treated as a radio button group and behave accordingly.

Tables are another interesting aspect of form filling, especially when numeric amounts are entered and could be automatically summed for the user, mimicking what spreadsheet programs already support.

4.3 Interface Extensions

Another worthwhile extension is to auto-fill fields that are repeated across multiple pages, such as a name or social security number which has to be filled out on each page.

It would also be useful for users to override the system’s choices. For example, radio buttons may be wrongly detected as checkboxes, or a field detected as a checkbox may in fact need to contain a single character or number. In this latter case, if the user types a key, the image of the checkbox could easily be replaced by what the user typed and the form type changed to a single-character text field.

Although FormCracker has been robust in identifying boxes and baselines for entering text for the forms that we have tried, we plan to implement a method to handle unusual cases, such as a form with a patterned background or non-existent baseline. The method we envision is similar to that used in [3][4] where the user manually positions the field. This insures that a user will be able to fill in all forms with FormCracker.

4.4 Collaboration

Finally, we would like to explore the collaborative nature of form filling: especially inside an enterprise setting where it would be useful to remember previous users’ actions to help new users fill out the same form. Similarly, it may be useful to populate the form with previously entered values, e.g. a book reimbursement form might contain a field titled “amount spent up to today” and help the user fill out the new book reimbursement form.

5. REFERENCES

[1] “iText PDF: your Java-PDF library”, Retrieved Jun, 2010. http://itextpdf.com

[2] Swanson, C. (2007, Sep 29). “Acrobat 8's new Automatic Form Recognition”. http://www.creativetechs.com/iq/acrobat_8s_new_automatic_form_recognition.html

[3] “PDFfiller: On-line PDF form filler, Editor, Type on PDF”, Retrieved Jun, 2010. http://www.pdffiller.com/

[4] “PDFescape: Free Online PDF Editor, PDF Form Filler & PDF Viewer”, Retrieved Jun, 2010. http://www.pdfescape.com

[5] Bagley, S. C. and Kopec, G. E. 1994. Editing images of text. *Commun. ACM* 37, 12 (Dec. 1994), 63-72. http://doi.acm.org/10.1145/198366.198382

[6] Casey, R., Ferguson, D., Mohiuddin, K., and Walach, E. 1992. Intelligent forms processing system. *Mach. Vision Appl.* 5, 3 (Jul. 1992), 143-155. http://dx.doi.org/10.1007/BF02626994

[7] Taylor, S. L., Fritzson, R., and Pastor, J. A. 1995. Extraction of data from preprinted forms. In *Document Image Analysis*, L. O'Gorman and R. Kasturi, Eds. IEEE Computer Society Press, Los Alamitos, CA, 391-402.

[8] Gugler, S. 1995. Method and apparatus for identifying text fields and checkboxes in digitized images. US Patent 5815595.