MaxTract: Converting PDF to LaTeX, MathML and Text

Josef B. Baker, Alan P. Sexton and Volker Sorge

School of Computer Science, University of Birmingham
Email: `j.baker|a.p.sexton|v.sorge@cs.bham.ac.uk`
URL: `http://www.cs.bham.ac.uk/~jbb|aps|vxs`

1 Introduction

In this paper we present the first public, online demonstration of MaxTract; a tool that converts PDF files containing mathematics into multiple formats including LaTeX, HTML with embedded MathML, and plain text. Using a bespoke PDF parser and image analyser, we directly extract character and font information to use as input for a linear grammar which, in conjunction with specialised drivers, can accurately recognise and reproduce both the two dimensional relationships between symbols in mathematical formulae and the one dimensional relationships present in standard text.

The main goals of MaxTract are to provide translation services into standard mathematical markup languages and to add accessibility to mathematical documents on multiple levels. This includes both accessibility in the narrow sense of providing access to content for print impaired users, such as those with visual impairments, dyslexia or dyspraxia, as well as more generally to enable any user access to the mathematical content at more re-usable levels than the merely visual. MaxTract produces output compatible with web browsers, screen readers, and tools such as copy and paste, which is achieved by enriching the regular text with mathematical markup. The output can also be used directly, within the limits of the presentation MathML produced, as machine readable mathematical input to software systems such as Mathematica or Maple.

2 MaxTract Process

Although the PDF documents that MaxTract works on are electronic documents with character and font information, actually extracting that information and then analysing it to construct an interpretation of the text and mathematical formulae contained is a somewhat involved process.

We start by using image analysis over an input file rendered to `TIF` to identify the precise bounding boxes of the glyphs, or connected components, on a page. This is necessary as the PDF format does not encode this information but precise bounding box information for the characters is critical to the two dimensional analysis necessary for mathematical formula recognition. The bounding boxes are mapped to the character and font information extracted via PDF analysis to produce a list of symbols with their detailed location and bounding box information. The extraction is completed by parsing the content streams and font objects comprising a PDF file with a bespoke PDF parser we have written, based upon the PDF specification [1]. A symbol consists of a name, bounding box, base point, font size and font name. Projection profile cutting is then used to identify lines of symbols which are passed to a linear grammar as lists of symbols.

The linear grammar creates a parse tree based upon the two dimensional relationships of the symbols and their appearance. The grammar has been designed to produce a parse tree rich enough to be translated by specialised drivers to produce a wide variety of output in various markup. The extraction process, analysis and grammar are explained in detail in [3,4].

3 Translation

We use three main output drivers to produce markup, namely a LaTeX, MathML and plain text driver. We combine these drivers in a number of ways to produce various output formats designed to be accessible to users with a wide variety of software. Here we explain the basic drivers.

3.1 Basic Drivers

Each of the basic drivers are used in conjunction with a layout analysis module, which identifies structures such as display mathematics, alignment and justification, columns and paragraphs.

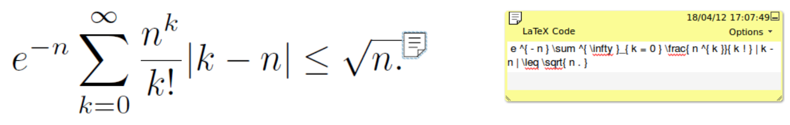
LaTeX This produces a `.tex` file which, when compiled, has been designed to reproduce the original formatting and style closely. All of the fonts identified from the original PDF file are reused, and reproduced together with layout and spacing where possible.

MathML The MathML driver returns an `.xhtml` file, containing standard HTML, interspersed with presentation MathML where appropriate. Unlike the LaTeX driver, styling is not closely reproduced, with standard HTML fonts and formatting used instead of the originals, however elements such as headings and paragraphs are retained.

Festival The festival driver produces plain text that can be given to text-to-speech engines directly. We have particularly focused on and experimented with Festival [5], and have added some optimisation for this particular engine.

3.2 Annotated PDF

The idea of annotated PDFs is not only to reproduce the original document using the LaTeX driver, but also allow the user to view and copy the markup for each mathematical formula when viewing the compiled PDF. By associating each formula with a `\pdfannot` command, which is processed by `pdflatex`, the user is presented with clickable notes containing the respective markup. These notes can be opened, closed, moved and edited by the user, and of course their content can be viewed and copied. An example is shown in Figure 1. The annotation features are not universally supported by PDF browsers, although they are part of the PDF specification and supported by Adobe Reader [2].



$$e^{-n}\sum_{k=0}^{\infty}\frac{n^k}{k!}|k-n| \leq \sqrt{n}.$$

```
e ^{ - n } \sum ^{ \infty }_{ k = 0 } \frac{ n ^{ k }}{ k ! } | k -
n | \leq \sqrt{ n . }
```

Fig. 1. Equation in a PDF file annotated with LaTeX

LaTeX Annotations A special version of the LaTeX driver is used to produced the LaTeX annotations. As the markup is designed to be viewed, copied and possibly edited, positioning and font commands are removed in order to improve the clarity and simplicity of the generated code.

MathML Annotations These are produced by the same MathML driver as described previously, containing valid snippets of MathML for each formula.

Double Annotations In addition to the singly annotated files we can also produce doubly annotated files, where each formula is associated with both the MathML and LaTeX.

3.3 Layered PDF

By taking advantage of the Optional Content features of PDF, we can create files that contain multiple layers, allowing a user to switch between, say the rendered file and the underlying LaTeX. To achieve this we make use of two additional packages, `OCG` [7] and `textpos` [6], the first of which is used to produce the separate layers and the second to position each layer correctly. Again these official features of PDF are not widely supported by PDF browsers other than Adobe Reader. Figure 2 shows the layer choice dialog in Adobe Reader with the various layers available in a document produced by MaxTract.

Text Layer This layer is produced by using the festival driver. The layer has been designed to work with the read-out-loud facility in Adobe Reader, allowing the screen reader to accurately read the whole document, including mathematics which is usually garbled in standard documents.

LaTeX Code Layer In a similar manner to the annotated PDF, this allows the user to see and copy the underlying LaTeX code. However, the layer shows the code for the whole page rather than just the formulae. This is the simple LaTeX code, without fonts and spacing commands.

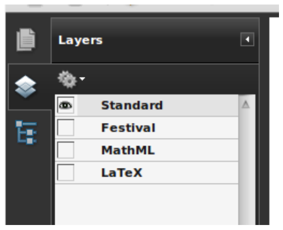


Fig. 2. Selection of layers in a PDF file

Both Layers Again, we can also produce double layered PDF files containing both the LaTeX and text.

3.4 Accessibility Formats

The accessibility formats that we produce are designed to be compatible with all screen readers. This is achieved by producing standard, plain text files, with none of the special characters or formatting that are often incompatible with accessibility tools. Any non ASCII symbols, or groups of symbols with non-linear relationships are replaced with alternative ASCII based text.

Text only This is the direct output of the festival driver, producing a plain text file that can be given to text-to-speech engines.

Text only as Latex Text only as LaTeX wraps the text described above with a standard LaTeX header and footer so that it can be compiled into a PDF file. No other commands are used and any mathematics is replaced by alternative text. This is essentially the same as the text layer from the layered PDF.

Text only as HTML This produces a standard `.html` file to be used with speech enabled browsers. The text is wrapped in HTML with a standard header and footer, and line break is the only tag used. Mathematical equations are replaced by in line alternative text.

4 MaxTract Online Interface

The MaxTract demonstration consists of an HTML form to select and upload a PDF file for extraction, select an output format and enter an email address. It can be found at `http://www.cs.bham.ac.uk/research/groupings/reasoning/sdag/maxtract.php`.

A PDF file is compatible with MaxTract if it contains only fonts and encodings that are embedded and of type 1. This can be checked by viewing the fonts tab in the file properties within Adobe Reader. Once uploaded, the file is processed if it is found to be compatible with MaxTract, and the user is emailed with a link to download the output. If the file cannot be processed, this will be confirmed via email.

An example of each type of output is also available to be viewed or downloaded from the MaxTract web site. As stated in section 3, some of these formats make use of advanced features of PDF which are not supported by all readers, however they are compatible with Adobe Reader which should be used to view our output.

References

1. Adobe. *PDF Reference fifth edition Adobe Portable Document Format Version 1.6*. Adobe Systems, 2004.
2. Adobe. *Adobe Reader X*. Adobe Systems, 2012. `http://get.adobe.com/uk/reader/`.
3. Josef B. Baker, Alan P. Sexton, and Volker Sorge. A linear grammar approach to mathematical formula recognition from PDF. In *Proceedings of the Conferences in Intelligent Computer Mathematics, CICM 2009*, volume 5625 of *LNAI*, pages 201–216. Springer, 2009.
4. Josef B. Baker, Alan P. Sexton, and Volker Sorge. Towards reverse engineering of PDF documents. In Petr Sojka and Thierry Bouche, editors, *Towards a Digital Mathematics Library, DML 2011*, pages 65–75, Bertinoro, Italy, July 2011. Masaryk University Press.
5. Alan W. Black and Paul A. Taylor. The Festival Speech Synthesis System: System documentation. Technical Report HCRC/TR-83, Human Communciation Research Centre, University of Edinburgh, Scotland, UK, 1997. Available at `http://www.cstr.ed.ac.uk/projects/festival.html`.
6. Norman Gray. Textpos, 2010. `http://purl.org/nxg/dist/textpos`.
7. Robert Marik. OCGtools, 2012. `http://ctan.org/pkg/ocgtools`.