Document-Encoding and PDF-Generation Software to Improve Usability (and Machine Readability) of PDF Documents

Academic Publishing

OVERVIEW:

Although most applications use PDF only for supplemental content, such as user manuals or digital publications, *interactive* PDF technology would significantly expand the use-cases where PDF documents can be an essential part of the User Interface for software applications. The relatively restricted adoption of PDF is driven partly because traditional PDF offers only limited options for interactivity, through features such as PDF scripting, and in fact scripting itself is limited because many PDF users/viewers disable scripting for security reasons. Another reason for the restricted use of PDF interactivity is that PDF generation tools are not programmed to identify document locations and actions where interactive capabilities are needed. Such limitations can, however, be rectified by embedding metadata within PDF documents which identify special actions to associate with individual PDF coordinate regions, and by implementing callback handlers in PDF viewers to support user actions and context menus linked to those coordinate regions.

PDF files augmented with these extra features can make the PDF experience much more interactive and user-friendly, allowing PDFs to play a larger role in application development. For exampe, PDF documents may be connected to external applications that provide graphics and multimedia, or application-specific features such as diagram/plotting engines and practice test questions.

More generally, interactive PDF technology can integrate PDF viewers with a wide range of software applications. This interoperability can be achieved by encoding and sharing the textual content from which a PDF document is generated, so that this raw text is available to user-action handlers within the PDF application, and by annotating external data and code which may be linked to the PDF application so as to cross-reference these external resources with the document text. The corresponding relationship between text encoding and data/code annotation is outlined below.

TECHNOLOGY STACK:

1. **A New Text Encoding Protocol** — Introducing a novel text encoding for document markup, including PDF generation, which produces user-friendly and machine-readable text representation and metadata that may be embedded as a PDF attachment. The resulting supplemental material, which a compatible PDF viewer can automatically extract, would consist of two parts: (1) a direct encoding of document text in the form of a character stream with a fixed character size (e.g. one byte per character); and (2) metadata representing the character positions/ranges and PDF coordinates of document content such as sentences, paragraphs, citations, keywords, and figures/graphics.

   With respect to document text, having a character stream (or several character stream "layers") as a direct representation of textual content — rather than "scraping" characters from the PDF directly — leads to more accurate searching and UI features (e.g. copy/select). For example, with one single context menu action users can copy an entire sentence/paragraph with just one click. This is made possible because the character streams employ a customizable encoding (instead of a generic format like Unicode) so that the document can specify ahead of time how text segments should be translated into character sequences for searches or selection-copy features; notably, how to handle foreign letters, equations, footnotes, and so forth, details which

ordinarily complicate basic functionality such as "copy a range of text to the clipboard." Fixed-width character encoding ensures that all text segments can be identified by simple numeric ranges, and text can be decoded by iterating over one character at a time in isolation, in contrast to Unicode multi-bytes or XML character entities. With respect to metadata, supplemental files encode annotations which map character-stream positions to PDF coordinates for document content that would be a target for user interactions, such as sentence boundaries (to enable automatic sentence-copy features), figures/graphics (for context menus providing figure information), citations (for capabilities such as searching cited publications' DOI) and so forth. We have put together a demo PDF viewer which reads the character and metadata files and uses this data to provide new capabilities that are not found in other PDF programs.

2. **Dataset Integration and "Microcitations"** — When integrating data sets with PDF documents, our text-encoding protocol is able to support fine-grained cross-references between individual parts of the data set and individual locations or regions of the document. This contrasts with the conventional approach wherein connections between data sets and publications are noted only coarsely, such as when a "Supplemental Materials" section of a paper provides a link to download an accompanying data set, but only with a cursory overview of what is inside the data set. By contrast, my team envisions granular cross-references where, for example, a concept embodied in a data set (for instance, a statistical parameter instantiated by a table column or datatype field) can be mapped to the sentence in the publication where that concept is defined or is first introduced. Likewise, tables or figures diagrammed in a document could be linked to the relevant part of their accompanying data set, such as individual tables which generate a statistical plot, or source code files where statistical distributions are calculated. So, in order to support these granular cross-references — which are a form of "microcitation" as this term is used in science — we have prototyped novel "dataset creator" tools that build customized "dataset applications" for viewing and analyzing dataset contents. In the spirit of initiatives such as Research Objects and FAIRsharing (Findable, Accessible, Interoperable, Reusable) — standards which have been prioritized by institutions such as the NIH, the Bill and Melinda Gates Foundation, and the Chan Zuckerberg Initiative — dataset applications combine code, data, and visualization/GUI tools and, moreover, can be launched directly from the PDF viewer itself. A standardized protocol cross-referencing PDF files with data sets (annotating specific parts of the document and the data set which are conceptually related) could also include specifications for launching external applications to view embedded multi-media content, such as 3D graphics or video files. In effect, support for dataset annotations could be merged with an annotation system whereby files embedded in a PDF document — potentially linked to visible PDF elements such as a 2D figure representing a view onto a 3D model or a still frame from a video — are annotated with metadata allowing the PDF viewer to export resources to an external application.

3. **Cloud Hosting** — Projecting forward, with the idea that authors and publishers start to provide granularly connected documents and data sets, it will be necessary to implement new hosting platforms which can leverage such document/data integration protocols. At present, data-hosting sites (such as Dryad, Sciverse, or Open Science) and document repositories (such as Science Direct, Springer Nature, or ScienceOpen) usually operate in parallel, but a next generation of content providers may need to host both data and documents simultaneously, in an integrated fashion, allowing for searches across both publications and data sets. This means that data sets have to be indexed and searchable as well. When a document is linked to a data set, the document viewer (canonically a PDF viewer) would then have the option of accessing the document source and retrieving dataset information — for example, identifying software prerequisites for using the data set and explaining to readers how they can acquire and access the dataset contents — and once the data set is downloaded and the relevant software is identified (either as an application embedded in the data set or as an external program) the PDF viewer could then connect with that software component as part of its user-interaction interface. These kinds of workflows would require APIs and multi-application networking protocols that could be concretized by providing cloud-hosting services or projects such as a reusable docker container, which demonstrate API and protocol implementations as well as document and dataset hosting capabilities.