Pandoc, Multiformat Publishing

Pandoc is a converter to and from different text formats. What attracted me to the tool is that it allows me to work with Microsoft Word documents and convert them to Markdown or any other formats.

Figure 1 shows the formts that Pandoc converts to and from.

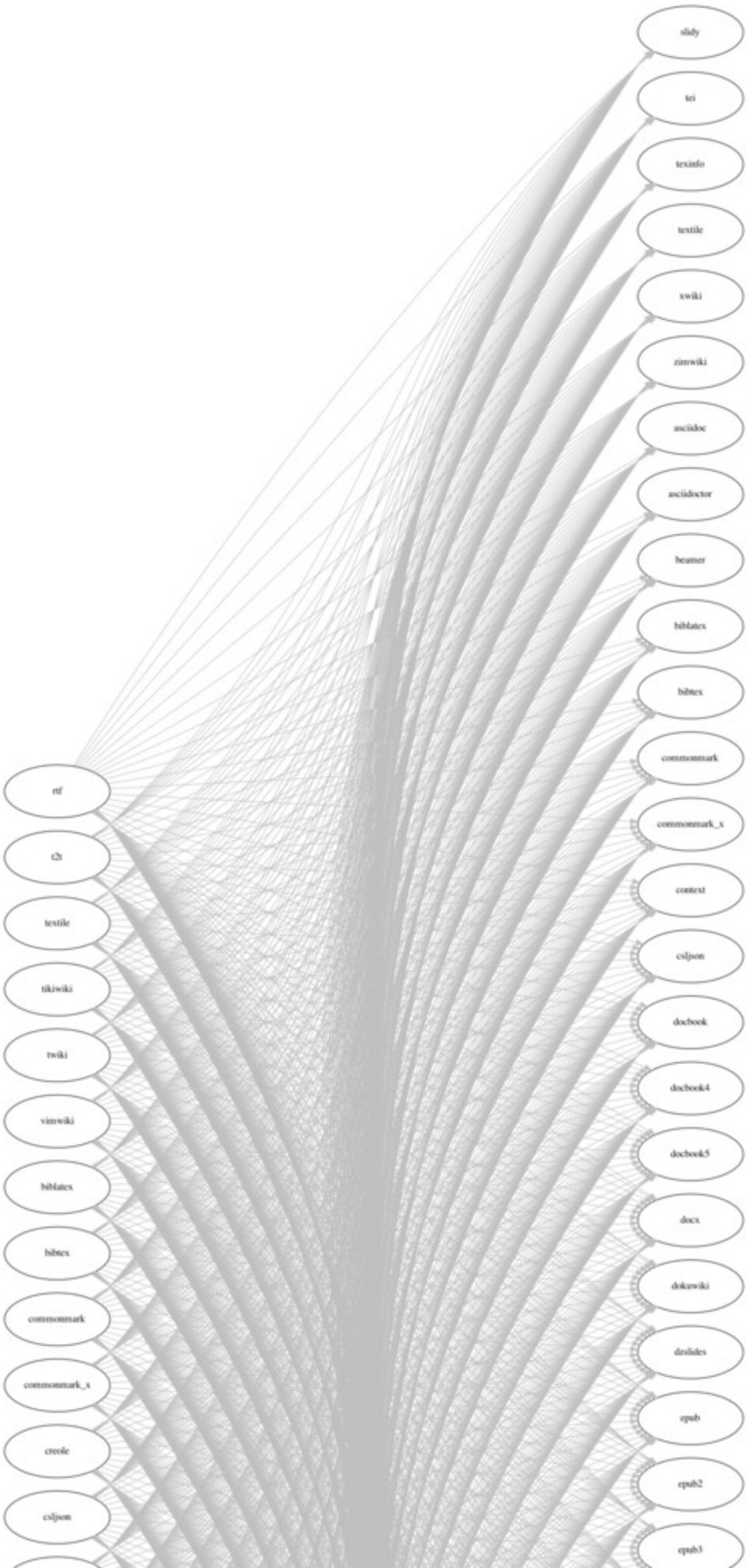
rtf
t2t
textile
tikiwiki
twiki
vimwiki
biblatex
bibtex
commonmark
commonmark_x
creole
csljson
docbook
docbook4
docbook5
docx
dokuwiki
dzslides
epub
epub2
epub3
fb2
gfm
haddock
html
html4
html5
icml
ipynb
jats
jats_archiving
jats_articleauthoring
jats_publishing
jira
json
latex
man
markdown
markdown_github
markdown_mmd
markdown_phpextra
markdown_strict
mediawiki
ms
muse
native
odt
opml
org
pdf
plain
pptx
revealjs
rst
s5
slidy
tei
texinfo
textile
xwiki
zimwiki
asciidoc
asciidoctor
beamer
biblatex
bibtex
commonmark
commonmark_x
context
csljson
docbook
docbook4
docbook5
docx
dokuwiki
dzslides
epub
epub2
epub3
fb2
gfm
haddock
html
html4
html5
icml
ipynb
jats
jats_archiving
jats_articleauthoring
jats_publishing
jira
json
latex
man
markdown
markdown_github
markdown_mmd
markdown_phpextra
markdown_strict
mediawiki
ms
muse
native
odt
opml
org
pdf
plain
pptx
revealjs
rst
s5
slidy
tei
texinfo
textile
xwiki
zimwiki

Figure 1: Pandoc Input and Output Format Visualization

Since I already work with Markdown this is a valued added tool as it allows me to convert Markdown to formats that would be very difficult or impossible to do without a tool.

We'll explore converting Markdown to epub3, an ebook standard and the starting point for Kindle conversion using Kindlegen, convert the same Markdown document to Latex and then explore an alternative to my CSS for Paged Media and PrinceXML way to create PDF documents.

Are these solutions perfect? No, definitely not. They are good starting points for future work.

- Using Pandoc to create epub books saves me from having to do the grunt work of manually creating the XML files required for epub.
- The Latex conversion gives me a working latex file that I can then further customize by adding packages and additional environments.
- The PDF conversion is a backup in case PrinceXML changes its current practice of not charging for development, only for production work

Markdown to epub

Epub, and more specifically, epub 3, is an ebook formatcreated by IDPF and now being submitted to the W3C as paart of the merger of the two institutions.

The format itself is a zipped file with a `application/epub+zip` mimetype. The contenxt of an example ebook are show in the following listing. We'll disect it below

```
peter-pan-sourcecode-sans
├── META-INF
|   ├── com.apple.ibooks.display-options.xml
|   └── container.xml
├── OEBPS
|   ├── ch01.xhtml
```

```
|   ├── ch02.xhtml
|   ├── cover.xhtml
|   ├── css
|   ├── images
|   ├── notes.xhtml
|   ├── package.opf
|   ├── toc.ncx
|   └── toc.xhtml
└── mimetype
```

The META-INF directory

The `META-INF` directory contains information about the book.

The iBooks propietary `com.apple.ibooks.display-options.xml` tells iBooks about special characteristics for one or more versions of the application (macOS, iPad or iPhone).

In this case we tell it that for all platforms we want to use custom fonts and we don't want to make this an interactive book. The code to do this is this:

```
<display_options>
  <!-- all devices -->
  <platform name="*">
    <platform name="*"><!-- set to "true" when embedding fonts -->specifi
    <!-- set to "true" </option><!-- set to "true" when using javascript c
    <option name="interactive">false</option>
  </platform>
</display_options>
```

In `container.xml` we tell the epub reader where to find the root of the book. This is the `package.opf` file, not an `index.html` or similar file. In our example content, the file looks like this and it points to the `package.opf` file inside the OEBPS directory (discussed in the next section):

```
<?xml version="1.0"?>
<container version="1.0" xmlns="urn:oasis:names:tc:opendocument:xmlns:cont
```

```
  <rootfiles>
    <rootfile full-path="OEBPS/package.opf"
      media-type="application/oebps-package+xml"/>
  </rootfiles>
</container>
```

If you're not targeting iBooks with your file you can remove `com.apple.ibooks.display-options.xml`from the `META-INF` directory but then iBooks will always use system fonts, even if you've packaged the fonts in your system.

The OEBPS directory

The OEBPS directory contains the actual book content plus a few XML files used to describe and define the structure to the book.

It's important to note that the contenxt is written in XHTML, either 1.1 or the XHTML version of HTML5. This poses additional restrictions and requirements.

1. All XHTML documents must have specific elements
    1. DOCTYPE declaration
    2. HTML element
    3. HEAD elemen
    4. TITLE element
    5. BODY element
2. All XHTML tag names & attribute names must be in lowercase
3. All XHTML elements must close. If it doesn't have a closing element then it must be closed within the opening tag like this: `<br/>`
4. All XHTML elements must be properly nested
5. All XHTML attribute values must be quoted

IMAGES and CSS contain the associated resources for the content.

The `package.opf` file is the core of the book. It provides the ebook reader with metadata for the publicaation as well as navigation and table of content structure.

The final file in this section, `toc.ncx`, acts as a backwards compatible bridge to epub 2, the previous version of the specification and still used by many ppublishers around the world.

The mimetype file

At the root of the book directory we must place a `mimetype` file. It has no extension and the only content in the file is the string `application/epub+zip` without a carriage return

Why Pandoc? How to create an epub

I've worked in creating epub and Kindle ebooks from scratch. Pandoc doesn't provide the cleanest ebook in the market but it creates all the XML files and it's just a matter of deciding how much cleanup you want to do.

The basic command is simple. Using a Markdown file as the source we use the following command.

```
pandoc sample.md -o sample.epub
```

We can add metadata using a syntax similar to YAML

```
---
title:
- type: main
text: My Book
- type: subtitle
text: An investigation of metadata
creator:
- role: author
text: John Smith
- role: editor
v 1.19.2.1 64
Pandoc User's Guide Creating EPUBs with pandoc
text: Sarah Jones
identifier:
- scheme: DOI
text: doi:10.234234.234/33
publisher: My Press
rights: © 2007 John Smith, CC BY-NC
```

```
---
```

Pandoc supports the following data types:

- **identifier** Either a string value or an object with fields text and scheme. Valid values for scheme are ISBN-10, GTIN-13, UPC, ISMN-10, DOI, LCCN, GTIN-14, ISBN-13, Legal deposit number, URN, OCLC, ISMN-13, ISBN-A, JP, OLCC
- **title** Either a string value, or an object with fields file-as and type, or a list of such objects. Valid values for type are main, subtitle, short, collection, edition, extended
- **creator** Either a string value, or an object with fields role, file-as, and text, or a list of such objects. Valid values for role are MARC relators, but pandoc will attempt to translate the human-readable versions (like “author” and “editor”) to the appropriate marc relators
- **contributor** Same format as creator
- **date** A string value in YYYY-MM-DD format. (Only the year is necessary.) Pandoc will attempt to convert other common date formats
- **lang** (or legacy: language) A string value in BCP 47 format. Pandoc will default to the local language if nothing is specified
- **subject** A string value or a list of such values
- **description** A string value
- **type** A string value
- **format** A string value
- **relation** A string value
- **coverage** A string value
- **rights** A string value
- **cover-image** The path to the cover image
- **stylesheet** The path to the CSS stylesheet
- **page-progression-direction** Either `ltr` or `rtl`. Specifies the page-progression-direction attribute for the spine element

By default, pandoc will download linked media (including audio and video) and include it in the EPUB container, providing a complete epub package that will work regardless of network connectivity and other external facotrs.

If you want to link to external media resources instead, use raw HTML in your source and add data-external=“1” to the tag with the src attribute.

For example:

```
<audio controls="1">
<source src="http://example.com/music/toccata.mp3"
  data-external="1" type="audio/mpeg">
</source>
</audio>
```

I recommend against linking to external resources unless you provide alternative feedback as this will make your book dependent on your nextwork connectivity and that far from reliable.

Markdown to Latex

LaTeX is a document preparation system. When writing, the writer uses plain text as opposed to the formatted text found in WYSIWYG word processors like Microsoft Word, LibreOffice Writer and Apple Pages. The writer uses markup tagging conventions to define the general structure of a document (such as article, book, and letter), to stylise text throughout a document (such as bold and italics), and to add citations and cross-references. A TeX distribution such as TeX Live or MikTeX is used to produce an output file (such as PDF or DVI) suitable for printing or digital distribution. Within the typesetting system, its name is stylised as LATEX.

I've always been interested in ways to move away from word processors into more convenient and easier to use than the bloated binary files resulting from Word, Pages and other Word Processors. My current favorite is Markdown because it's easier to read and I've worked on toolchains to convert the markdown to HTML and PDF.

LaTex is a good backup option that allows me to create PDF (and will be the intermediary step when we convert Markdown to PDF) and HTML from LaTex sources.

The command to convert Markdown to LaTex is simple:

```
pandoc -s sample.md -o sample.tex
```

The `-s` flag will make sure that we generate a complete document rather than a fragment. Otherwise the LaTex content will not work with other items in the toolchain.

An alternative: Markdown to PDF

The final task I want to discuss is converting Markdown to PDF with a toolchain other than what I currently use (Markdown to HTML and HTML through CSS Paged Media to PDF using PrinceXML). This process provides an alternative tool chain going from Mardown to LaTex and from LaTex to PDF.

The format of the PDF looks too much like a LaTex document and I've never been a fan. But the toolchain is open source (eventhough it's my least favorite license, GPL) so I don't have to worry about the vendor changing its mind about the licensing for the tool.

```
pandoc -s sample.md -o sample.pdf
```

Further thoughts

We've just scratched the surface of what Pandoc can do. One interesting idea is to convert Markdown to ICML (In Copy Markup Language) that we can then import to an InDesign template that we've set up in advance.