Understanding pandoc lua filters

by Abhishek Ulayil

Abstract pandoc supports intermediate modification of the Abstract Syntax Tree (AST) between the parsing and writing phase using filters. This supplement paper will highlight the use cases of such filters written in Lua Language on LaTeX,markdown and native AST.

1 Introduction

To fully understand pandoc filters, one must first grasp the document conversion process. A document in one format is read/parsed into an intermediate form. An abstract syntax tree (AST) is what pandoc calls it. A document writer is a specialized type-setter to read an AST and produces the contents in the chosen article format. Converting markdown to HTML, for example, will entail these pandoc operations. Figure1 shows the conversion workflow withing pandoc.

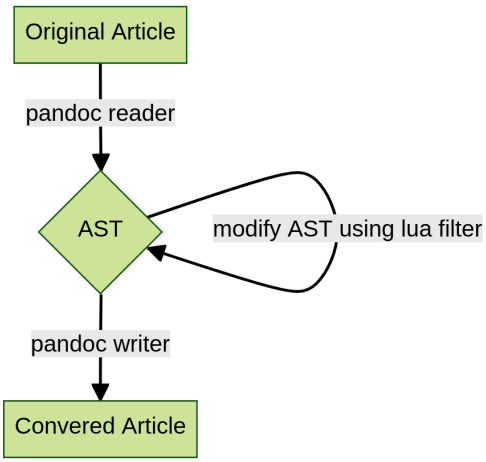

Figure 1: Conversion Workflow

When specific elements require adjustment, the filters enter the picture. Assume you're converting markdown to HTML and want the page to have automated numbering for figures or tables. If you use markdown, there is no system for automatically numbering figures/tables; to retain 1:1 conversion, the HTML output will also lack numbering.

At this point, you might wish there was a way to add such a feature. Then there is the possibility of unlimited choices for customization, including every customization as an option in pandoc or the writer is not viable. To address this, the idea of filters emerged, in which you could change the AST partially and modify it before the writer could read it. As a result, you get the desired result (MacFarlane and other pandoc authors , 2023).

2 Need of lua filters

Continuing the example above lets create a dummy article, where we need to add numbering. One way could be to keep a counter of Images, and add a prefix `"Figure X :"` to each figure caption. This will serve the purpose of numbering the Images in the end result.

```
![R logo](Rlogo-5.png){width="10%"}

![penguins](penguins.png){width="15%"}


pandoc example.md --from markdown --to html5 --output example.html
```

Figure 2: pandoc command without lua filter or extensions

Now If we convert the above markdown file to HTML5 using pandoc command 2, we get 3

R logo

penguins

Figure 3: Vanilla HTML5 output

As we can see, there is no figure numbering done automatically, which is generally the expected result. However if we want to include numbering to it, we would need to write a lua filter. This lua filter will modify the AST and make the changes we desire.

We call a lua filter in the pandoc command 2 using the `--lua-filter name_of_filter.lua` option in pandoc.

Now we write a lua filter to manipulate the figures in 4

3 Writing pandoc lua filters

```
figures = 0
is_fig = 0

function Figure(el)
      local label = ""
      pandoc.walk_block(el,{ Image = function(el)
                    is_fig = 1
                    end})
      if is_fig == 1 then
        figures = figures + 1
        label = "Figure " .. tostring(figures) .. ":"
      end
    local caption = pandoc.utils.stringify(el.caption)
    if not caption then
          caption = label
     else
          caption = label .. " " .. caption
     end
     el.caption.long[1] = caption
     is_fig = 0
     return el
end
```

Figure 4: A lua filter to add figure numbering

Everything in Lua revolves around tables; for example, the pandoc AST or the document is one giant table with sub-tables.

In pandoc, there are two sorts of elements: `Blocks` and `Inlines`. `Blocks` are complicated constructions constructed from simple pieces (`Inlines`). A `Para`, for example, is a `Block` made up of several `Str Inlines`.

The first step in writing a filter is to select a target type, that is, which block or Inline this filter will target. Following the selection, we name the function after the target type and each element in the argument as el. For instance, in the case of our filter, it would be `function Figure(el)`. Pandoc is smart enough to match the names of filter functions to the AST elements.

Now within the filter function we can assume that `el` will contain the table object of a Figure element from the document. we can use many functions over it or `Inlines` contained in it. One such function to walk over all the elements inside the block is `walk_block`. walk functions are great to check or count the presence of certain elements in a block. we use `walk_block` to check if there are any Image inline elements within the Figure block. This is because after pandoc 3 (Krewinkel, Lucero , 2023) Figure blocks can now contain elements other than Images as well.

If there is an Image then we append `"Figure X :"` to the caption of the Figure element. and return the modified element.

4 Interpreting changes using lua filters

The filter 4 when included in the pandoc command will generate 6 using the 5 command.

```
pandoc example.md --from markdown --to html5 --output filtered-example.html
--lua-filter image_numbering_filter.lua
```

Figure 5: pandoc command with lua filter

Figure 1: R logo

Figure 2: penguins

Figure 6: HTML5 output with desired filtering

5 Usage of pandoc lua filters in texor package

6 Summary

7 Supplementary materials

The example conversion files described here are included as supplementary materials in the Rjournal article of texor.

Bibliography

A. Krewinkel and A. Lucero, pandoc 3.0 Release notes, *pandoc* 2023 URL https://pandoc.org/releases.html [p3]

John MacFarlane and other pandoc authors, Pandoc Lua filters, *pandoc documentation* 2023 URL https://pandoc.org/lua-filters.html [p1]

Abhishek Ulayil
Student, Institute of Actuaries of India
Mumbai, India
ORCiD: 0009-0000-6935-8690