Parsing a PDF with no /Root object using PDFMiner

Asked 11 years, 4 months ago Modified 4 years, 11 months ago Viewed 21k times

14

I'm trying to extract text from a large number of PDFs using PDFMiner python bindings. The module I wrote works for many PDFs, but I get this somewhat cryptic error for a subset of PDFs:

ipython stack trace:

```
/usr/lib/python2.7/dist-packages/pdfminer/pdfparser.pyc in set_parser(self,
parser)
    331                 break
    332         else:
--> 333             raise PDFSyntaxError('No /Root object! - Is this really a
PDF?')
    334         if self.catalog.get('Type') is not LITERAL_CATALOG:
    335             if STRICT:

PDFSyntaxError: No /Root object! - Is this really a PDF?
```

Of course, I immediately checked to see whether or not these PDFs were corrupted, but they can be read just fine.

Is there any way to read these PDFs despite the absence of a root object? I'm not too sure where to go from here.

Many thanks!

Edit:

I tried using PyPDF in an attempt to get some differential diagnostics. The stack trace is below:

```
In [50]: pdf = pyPdf.PdfFileReader(file(fail, "rb"))
---------------------------------------------------------------------------
PdfReadError                              Traceback (most recent call last)
/home/louist/Desktop/pdfs/indir/<ipython-input-50-b7171105c81f> in <module>()
----> 1 pdf = pyPdf.PdfFileReader(file(fail, "rb"))

/usr/lib/pymodules/python2.7/pyPdf/pdf.pyc in __init__(self, stream)
    372         self.flattenedPages = None
    373         self.resolvedObjects = {}
--> 374         self.read(stream)
    375         self.stream = stream
```

```
711
712         # find startxref entry - the location of the xref table
```

```
PdfReadError: EOF marker not found
```

Quonux suggested that perhaps PDFMiner stopped parsing after reaching the first EOF character. This would seem to suggest otherwise, but I'm very much clueless. Any thoughts?

python pypdf pdf-parsing pdf-manipulation

Share Improve this question Follow

edited Jul 13, 2012 at 21:24

Ravindra S
6,342 12 71 109

asked Jul 8, 2012 at 16:06

Louis Thibault
20.5k 25 85 153

maybe PDFMiner terminates the search for the Root Node after the first %%EOF label *but* after that label can come more Nodes so it doesn't find it. Another reason coulbe be that the files are compressed? – Quonux Jul 8, 2012 at 16:15

@Quonux, And how would I go about testing whether or not this is the case? Is there an option to force PDFMiner to search the entire document for a Root Node? Concerning the possibility of compression, is there a way to check for this? What can be done if the files are compressed? – Louis Thibault Jul 8, 2012 at 16:48

@Quonux, I've added a stack trace from a similar attempt using pypdf. Does this help narrow down the cause? – Louis Thibault Jul 8, 2012 at 17:09

Maybe the parser expects an %%EOF label, but none is found... maybe you can fix it with: - open the "incorrect" file - write/append in binary mode "%%EOF\n" at the end of the file - close it - try to parse again – Quonux Jul 8, 2012 at 18:53

5 Answers

Sorted by: Highest score (default)

6

The solution in slate pdf is use 'rb' --> read binary mode.

Because slate pdf is depends on the PDFMiner and I have the same problem, this should solve your problem.

```
fp = open('C:\Users\USER\workspace\slate_minner\document1.pdf','rb')
doc = slate.PDF(fp)
print doc
```

2 I tried this option and got the exact same error code: `No /Root object! - Is this really a PDF?` – elPastor Jul 19, 2017 at 18:32

5

+100

interesting problem. i had performed some kind of research:

function which parsed pdf (from miners source code):

```
def set_parser(self, parser):
        "Set the document to use a given PDFParser object."
        if self._parser: return
        self._parser = parser
        # Retrieve the information of each header that was appended
        # (maybe multiple times) at the end of the document.
        self.xrefs = parser.read_xref()
        for xref in self.xrefs:
            trailer = xref.get_trailer()
            if not trailer: continue
            # If there's an encryption info, remember it.
            if 'Encrypt' in trailer:
                #assert not self.encryption
                self.encryption = (list_value(trailer['ID']),
                                   dict_value(trailer['Encrypt']))
            if 'Info' in trailer:
                self.info.append(dict_value(trailer['Info']))
            if 'Root' in trailer:
                #  Every PDF file must have exactly one /Root dictionary.
                self.catalog = dict_value(trailer['Root'])
                break
        else:
            raise PDFSyntaxError('No /Root object! - Is this really a PDF?')
        if self.catalog.get('Type') is not LITERAL_CATALOG:
            if STRICT:
                raise PDFSyntaxError('Catalog not found!')
        return
```

if you will be have problem with EOF another exception will be raised: '''another function from source'''

```
def load(self, parser, debug=0):
        while 1:
            try:
                (pos, line) = parser.nextline()
                if not line.strip(): continue
            except PSEOF:
                raise PDFNoValidXRef('Unexpected EOF - file corrupted?')
            if not line:
                raise PDFNoValidXRef('Premature eof: %r' % parser)
            if line.startswith('trailer'):
                parser.seek(pos)
```

```
                try:
                    (start, nobjs) = map(long, f)
                except ValueError:
                    raise PDFNoValidXRef('Invalid line: %r: line=%r' % (parser,
line))
                for objid in xrange(start, start+nobjs):
                    try:
                        (_, line) = parser.nextline()
                    except PSEOF:
                        raise PDFNoValidXRef('Unexpected EOF - file corrupted?')
                    f = line.strip().split(' ')
                    if len(f) != 3:
                        raise PDFNoValidXRef('Invalid XRef format: %r, line=%r' %
(parser, line))
                    (pos, genno, use) = f
                    if use != 'n': continue
                    self.offsets[objid] = (int(genno), long(pos))
            if 1 <= debug:
                print >>sys.stderr, 'xref objects:', self.offsets
            self.load_trailer(parser)
            return
```

from wiki(pdf specs): A PDF file consists primarily of objects, of which there are eight types:

```
Boolean values, representing true or false
Numbers
Strings
Names
Arrays, ordered collections of objects
Dictionaries, collections of objects indexed by Names
Streams, usually containing large amounts of data
The null object
```

Objects may be either direct (embedded in another object) or indirect. Indirect objects are numbered with an object number and a generation number. **An index table called the xref table gives the byte offset of each indirect object from the start of the file. This design allows for efficient random access to the objects in the file, and also allows for small changes to be made without rewriting the entire file (incremental update)**. Beginning with PDF version 1.5, indirect objects may also be located in special streams known as object streams. This technique reduces the size of files that have large numbers of small indirect objects and is especially useful for Tagged PDF.

i thk the problem is your "damaged pdf" have a few 'root elements' on the page.

```
Possible solution:
```

you can download sources and write `print function' in each places where xref objects

Share Improve this answer Follow edited Jul 12, 2012 at 7:25 answered Jul 11, 2012 at 17:55

Dmitry Zagorulkin
8,428 4 37 60

Dmitry, thanks for your response. If I understand correctly, you suspect that this is a bug in PDFMiner? I'm surprised because similar behavior is also observed in PyPDF. Or did you mean that the bug is in whichever software created the "broken" PDF? Concerning your solution, do you mean that I should add print lines in the PDFParser object methods whevever they manage an xref object? I'm a little unclear on exactly what I should be doing. Thanks! – Louis Thibault Jul 12, 2012 at 12:37

Good day. Just take a two files(normal and damaged) and try to analyse each in pdf analyser tool. i think that in damaged pdf will be invalid xref structure. After analysis try to repair pdf structure(w3.org/WAI/GL/WCAG20-TECHS/pdf.html) – Dmitry Zagorulkin Jul 12, 2012 at 12:58

pdflabs.com/tools/pdftk-the-pdf-toolkit great set of tools for performing this kind of actions. – Dmitry Zagorulkin Jul 12, 2012 at 13:03

Dimitry, Thanks for the links. I'll give pdftk a shot. Why bother fixing something if the work's already been done ;-) – Louis Thibault Jul 12, 2012 at 13:25

1 give me a feedback if you make pdf repair tool in python =) – Dmitry Zagorulkin Jul 12, 2012 at 13:53

1

I have had this same problem in Ubuntu. I have a very simple solution. Just print the pdf-file as a pdf. If you are in Ubuntu:

1. Open a pdf file using the (ubuntu) document viewer.
2. Goto File
3. Goto print
4. Choose print as file and check the mark "pdf"

If you want to make the process automatic, follow for instance this, i.e., use this script to print automatically all your pdf files. A linux script like this also works:

```
for f in *.pdfx
do
lowriter --headless --convert-to pdf "$f"
done
```

Note I called the original (problematic) pdf files as pdfx.

Share Improve this answer Follow edited Dec 13, 2018 at 2:10 answered Dec 13, 2018 at 2:00

0

However, I still got the error OP indicated. What I found is that I had bug in my code where the PDF was still open by another function.
So make sure you don't have the PDF open in memory elsewhere as well.

Share Improve this answer Follow

answered Apr 3, 2018 at 20:28

dasvootz
413 1 5 15

-1

An answer above is right. This error appears only in windows, and workaround is to replace `with open(path, 'rb')` to `fp = open(path,'rb')`

Share Improve this answer Follow

answered Feb 7, 2018 at 13:09

AeroSM
21 1 4