Parsing and indexing PDF in Python

Benjamin Bertrand — 2016-11-16 21:59 — 2 Comments

I have a Doxie Go scanner and I scan all the documents I receive in paper. That's nice, but it creates another problem. All the resulting PDF files have to be named, organized and stored... Doing that manually is boring and time consuming. Of course that's something I want to automate!

I even bought Hazel a while ago. It's a nice software that monitors files in a folder and performs specific instructions based on the rules you defined. It works well but I felt a bit limited and I thought I could probably write something more tailored to my use case. And that would be more fun :-)

Parsing PDF in Python

A quick solution I found was to run pdftotext using subprocess. I looked at PDFMiner, a pure Python PDF parser but I found pdftotext output to be more accurate. On MacOS, you can install it using Homebrew:

```
$ brew install Caskroom/cask/pdftotext
```

Here is a simple Python function to do that:

In [1]:
```
import subprocess

def parse_pdf(filename):
    try:
        content = subprocess.check_output(["pdftotext", '-enc', 'UTF-8', filename, "-"])
    except subprocess.CalledProcessError as e:
        print('Skipping {} (pdftotext returned status {})'.format(filename, e.returncode))
        return None
    return content.decode('utf-8')
```

Let's try to parse a pdf file. We'll use `requests` to download a sample file.

In [2]:
```
import requests

url = 'http://www.cbu.edu.zm/downloads/pdf-sample.pdf'
response = requests.get(url)
with open('/tmp/pdf-sample.pdf', 'wb') as f:
    f.write(response.content)
```

Let's first look at the PDF:

```
In [3]:  from IPython.display import IFrame
         IFrame('http://www.cbu.edu.zm/downloads/pdf-sample.pdf', width=600, height=870)

Out[3]:
```

Nothing complex. It should be easy to parse.

```
In [4]:  content = parse_pdf('/tmp/pdf-sample.pdf')
         content

Out[4]:  "Adobe Acrobat PDF Files\nAdobe® Portable Document Format (PDF) is a universal file format that preserves all of the
         fonts, formatting, colours and graphics of any source document, regardless of the application and platform used to
         create it. Adobe PDF is an ideal format for electronic document distribution as it overcomes the problems commonly
         encountered with electronic file sharing. • Anyone, anywhere can open a PDF file. All you need is the free Adobe
         Acrobat Reader. Recipients of other file formats sometimes can't open files because they don't have the applications
         used to create the documents. PDF files always print correctly on any printing device. PDF files always display
         exactly as created, regardless of fonts, software, and operating systems. Fonts, and graphics are not lost due to
         platform, software, and version incompatibilities. The free Acrobat Reader is easy to download and can be freely
         distributed by anyone. Compact PDF files are smaller than their source files and download a page at a time for fast
         display on the Web.\n\n• •\n\n• •\n\n\x0c"
```

This works quite well. The layout is not respected but it's the text that matters. It would be easy to define some regex to define rules based on the PDF content.

This could be the first step in naming and organizing the scanned documents. But it would be nice to have an interface to easily search in all the files. I've already used MongoDB full text search in a webapp I wrote and it worked well for my use case. But I read about Elasticsearch and I always wanted to give it a try.

Elasticsearch Ingest Attachment Processor Plugin

I could just index the result from pdftotext, but I know there is a plugin that can parse PDF files.

The Mapper Attachments Type plugin is deprecated in 5.0.0. It has been replaced with the ingest-attachment plugin. So let's look at that.

Running Elasticsearch

To run Elasticsearch, the easiest is to use Docker. As the official image from Docker Hub comes with no plugin, we'll create our own image. See Elasticsearch Plugin Management with Docker for more information.

Here is our `Dockerfile`:

```
FROM elasticsearch:5

RUN /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-attachment
```

Create the `elasticsearch-ingest` docker image:

```
$ docker build -t elasticsearch-ingest .
```

We can now run elasticsearch with the ingest-attachment plugin:

```
$ docker run -d -p 9200:9200 elasticsearch-ingest
```

Python Elasticsearch Client

We'll use elasticsearch-py to interact with our Elasticsearch cluster.

In [5]:
```
from elasticsearch import Elasticsearch
es = Elasticsearch()
```

Let's first check that our elasticsearch cluster is alive by asking about its health:

In [6]:
```
es.cat.health()
```

Out[6]:
```
'1479333419 21:56:59 elasticsearch green 1 1 0 0 0 0 0 0 - 100.0%\n'
```

Nice! We can start playing with our ES cluster.

As described in the documentation, we first have to create a pipeline to use the Ingest Attachment Processor Plugin:

```
PUT _ingest/pipeline/attachment
{
  "description" : "Extract attachment information",
  "processors" : [
    {
      "attachment" : {
        "field" : "data"
      }
    }
  ]
}
```

OK, how do we do that using the Python client?

In [7]:
```
body = {
    "description" : "Extract attachment information",
    "processors" : [
        {
            "attachment" : {
                "field" : "data"
            }
        }
    ]
}
es.index(index='_ingest', doc_type='pipeline', id='attachment', body=body)
```

Out[7]:
```
{'acknowledged': True}
```

Now, we can send a document to our pipeline. Let's start by using the same example as in the documentation:

```
PUT my_index/my_type/my_id?pipeline=attachment
{
  "data": "e1xydGYxXGFuc2kNCkxvcmVtIGlwc3VtIGRvbG9yIHNpdCBhbWV0DQpccGFyIH0="
}
```

Using Python client, this gives:

In [8]:
```
result1 = es.index(index='my_index', doc_type='my_type', pipeline='attachment',
                   body={'data':
"e1xydGYxXGFuc2kNCkxvcmVtIGlwc3VtIGRvbG9yIHNpdCBhbWV0DQpccGFyIH0="})
result1
```

Out[8]:
```
{'_id': 'AVhvJKzVIvjFWZACJU_t',
 '_index': 'my_index',
 '_shards': {'failed': 0, 'successful': 1, 'total': 2},
 '_type': 'my_type',
 '_version': 1,
 'created': True,
 'result': 'created'}
```

Let's try to get the created document based on its id:

In [9]:
```
es.get(index='my_index', doc_type='my_type', id=result1['_id'])
```

Out[9]:
```
{'_id': 'AVhvJKzVIvjFWZACJU_t',
 '_index': 'my_index',
 '_source': {'attachment': {'content': 'Lorem ipsum dolor sit amet',
   'content_length': 28,
   'content_type': 'application/rtf',
   'language': 'ro'},
  'data': 'e1xydGYxXGFuc2kNCkxvcmVtIGlwc3VtIGRvbG9yIHNpdCBhbWV0DQpccGFyIH0='},
 '_type': 'my_type',
 '_version': 1,
 'found': True}
```

We can see that the binary data passed to the pipeline was a Rich Text Format file and that the content was extracted: *Lorem ipsum dolor sit amet*

Displaying the binary data is not very useful. It doesn't matter in this example as it's quite small. But it would be much bigger even on small files. We can exclude it using `_source_exclude`:

In [10]:
```
es.get(index='my_index', doc_type='my_type', id=result1['_id'], _source_exclude=['data'])
```

Out[10]:
```
{'_id': 'AVhvJKzVIvjFWZACJU_t',
 '_index': 'my_index',
 '_source': {'attachment': {'content': 'Lorem ipsum dolor sit amet',
   'content_length': 28,
   'content_type': 'application/rtf',
   'language': 'ro'}},
 '_type': 'my_type',
 '_version': 1,
 'found': True}
```

Indexing PDF files

Let's try to parse the same sample pdf as before.

```
In [11]:  url = 'http://www.cbu.edu.zm/downloads/pdf-sample.pdf'
          response = requests.get(url)
```

Note that we have to encode the content of the pdf before to pass it to ES. The source field must be a base64 encoded binary.

```
In [12]:  import base64

          data = base64.b64encode(response.content).decode('ascii')
```

```
In [13]:  result2 = es.index(index='my_index', doc_type='my_type', pipeline='attachment',
                            body={'data': data})
          result2
```

```
Out[13]: {'_id': 'AVhvJMC6IvjFWZACJU_u',
          '_index': 'my_index',
          '_shards': {'failed': 0, 'successful': 1, 'total': 2},
          '_type': 'my_type',
          '_version': 1,
          'created': True,
          'result': 'created'}
```

We can get the document based on its id:

```
In [14]:  doc = es.get(index='my_index', doc_type='my_type', id=result2['_id'], _source_exclude=['data'])
          doc
```

```
Out[14]: {'_id': 'AVhvJMC6IvjFWZACJU_u',
          '_index': 'my_index',
          '_source': {'attachment': {'author': 'cdaily',
            'content': "Adobe Acrobat PDF Files\n\nAdobe® Portable Document Format (PDF) is a universal file format that
preserves all\nof the fonts, formatting, colours and graphics of any source document, regardless of\nthe application
and platform used to create it.\n\nAdobe PDF is an ideal format for electronic document distribution as it overcomes
the\nproblems commonly encountered with electronic file sharing.\n\n•  Anyone, anywhere can open a PDF file. All you
need is the free Adobe Acrobat\nReader. Recipients of other file formats sometimes can't open files because
they\ndon't have the applications used to create the documents.\n\n•  PDF files always print correctly on any
printing device.\n\n•  PDF files always display exactly as created, regardless of fonts, software, and\noperating
systems. Fonts, and graphics are not lost due to platform, software, and\nversion incompatibilities.\n\n•  The free
Acrobat Reader is easy to download and can be freely distributed by\nanyone.\n\n•  Compact PDF files are smaller than
their source files and download a\npage at a time for fast display on the Web.",
            'content_length': 1073,
            'content_type': 'application/pdf',
            'date': '2000-06-28T23:21:08Z',
            'language': 'en',
            'title': 'This is a test PDF file'}},
          '_type': 'my_type',
          '_version': 1,
          'found': True}
```

Or with a basic search:

In [15]:
```
es.search(index='my_index', doc_type='my_type', q='Adobe', _source_exclude=['data'])
```

Out[15]:
```
{'_shards': {'failed': 0, 'successful': 5, 'total': 5},
 'hits': {'hits': [{'_id': 'AVhvJMC6IvjFWZACJU_u',
    '_index': 'my_index',
    '_score': 0.45930308,
    '_source': {'attachment': {'author': 'cdaily',
      'content': "Adobe Acrobat PDF Files\n\nAdobe® Portable Document Format (PDF) is a universal file format that preserves all\nof the fonts, formatting, colours and graphics of any source document, regardless of\nthe application and platform used to create it.\n\nAdobe PDF is an ideal format for electronic document distribution as it overcomes the\nproblems commonly encountered with electronic file sharing.\n\n•  Anyone, anywhere can open a PDF file. All you need is the free Adobe Acrobat\nReader. Recipients of other file formats sometimes can't open files because they\ndon't have the applications used to create the documents.\n\n•  PDF files always print correctly on any printing device.\n\n•  PDF files always display exactly as created, regardless of fonts, software, and\noperating systems. Fonts, and graphics are not lost due to platform, software, and\nversion incompatibilities.\n\n•  The free Acrobat Reader is easy to download and can be freely distributed by\nanyone.\n\n•  Compact PDF files are smaller than their source files and download a\npage at a time for fast display on the Web.",
      'content_length': 1073,
      'content_type': 'application/pdf',
      'date': '2000-06-28T23:21:08Z',
      'language': 'en',
      'title': 'This is a test PDF file'}},
    '_type': 'my_type'}],
  'max_score': 0.45930308,
  'total': 1},
 'timed_out': False,
 'took': 75}
```

Of course Elasticsearch allows much more complex queries. But that's something for another time.

One interesting thing is that by printing the content, we can see that even the layout is quite acurate! Much better than the pdftotext output:

In [16]:
```
print(doc['_source']['attachment']['content'])
```

```
Adobe Acrobat PDF Files

Adobe® Portable Document Format (PDF) is a universal file format that preserves all
of the fonts, formatting, colours and graphics of any source document, regardless of
the application and platform used to create it.

Adobe PDF is an ideal format for electronic document distribution as it overcomes the
problems commonly encountered with electronic file sharing.

•  Anyone, anywhere can open a PDF file. All you need is the free Adobe Acrobat
Reader. Recipients of other file formats sometimes can't open files because they
don't have the applications used to create the documents.

•  PDF files always print correctly on any printing device.

•  PDF files always display exactly as created, regardless of fonts, software, and
operating systems. Fonts, and graphics are not lost due to platform, software, and
version incompatibilities.

•  The free Acrobat Reader is easy to download and can be freely distributed by
anyone.

•  Compact PDF files are smaller than their source files and download a
page at a time for fast display on the Web.
```

The ingest-attachment plugin uses the Apache text extraction library Tika. It's really powerful. It detects and extracts metadata and text from many file types.

Sending the file directly to Elasticsearch is nice, but in my use case, I'd like to process the file (change its title, move it to a specific location...) based on its content. I could of course update the document in ES after processing it.

It might be better in some case to decorelate the parsing and processing from the indexing. So let's check how to use Tika from Python.

Apache Tika

Tika-Python makes Apache Tika available as a Python library. It can even starts a Tika REST server in the background, but this requires Java 7+ to be installed. I prefer to run the server myself using the prebuilt docker image: docker-tikaserver. Like that I have control of what is running.

```
$ docker run --rm -p 9998:9998 logicalspark/docker-tikaserver
```

We can then set Tika-Python to use Client mode only:

In [17]:
```
import tika
tika.TikaClientOnly = True
from tika import parser
```

In [18]:
```
parsed = parser.from_file('/tmp/pdf-sample.pdf', 'http://localhost:9998/tika')
```

```
2016-11-16 22:57:14,233 [MainThread  ] [INFO ]  Starting new HTTP connection (1): localhost
```

In [19]:
```
parsed
```

Out[19]:
```
{'content': "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\nThis is a test PDF
file\n\n\nAdobe Acrobat PDF Files\n\nAdobe® Portable Document Format (PDF) is a universal file format that preserves
all\nof the fonts, formatting, colours and graphics of any source document, regardless of\nthe application and
platform used to create it.\n\nAdobe PDF is an ideal format for electronic document distribution as it overcomes
the\nproblems commonly encountered with electronic file sharing.\n\n•  Anyone, anywhere can open a PDF file. All you
need is the free Adobe Acrobat\nReader. Recipients of other file formats sometimes can't open files because
they\ndon't have the applications used to create the documents.\n\n•  PDF files always print correctly on any
printing device.\n\n•  PDF files always display exactly as created, regardless of fonts, software, and\noperating
systems. Fonts, and graphics are not lost due to platform, software, and\nversion incompatibilities.\n\n•  The free
Acrobat Reader is easy to download and can be freely distributed by\nanyone.\n\n•  Compact PDF files are smaller than
their source files and download a\npage at a time for fast display on the Web.\n\n\n",
 'metadata': {'Author': 'cdaily',
  'Content-Type': 'application/pdf',
  'Creation-Date': '2000-06-28T23:21:08Z',
  'Last-Modified': '2013-10-28T19:24:13Z',
  'Last-Save-Date': '2013-10-28T19:24:13Z',
  'X-Parsed-By': ['org.apache.tika.parser.DefaultParser',
   'org.apache.tika.parser.pdf.PDFParser'],
  'X-TIKA:parse_time_millis': '62',
  'access_permission:assemble_document': 'true',
  'access_permission:can_modify': 'true',
  'access_permission:can_print': 'true',
  'access_permission:can_print_degraded': 'true',
  'access_permission:extract_content': 'true',
  'access_permission:extract_for_accessibility': 'true',
  'access_permission:fill_in_form': 'true',
  'access_permission:modify_annotations': 'true',
  'created': 'Wed Jun 28 23:21:08 UTC 2000',
  'creator': 'cdaily',
  'date': '2013-10-28T19:24:13Z',
  'dc:creator': 'cdaily',
  'dc:format': 'application/pdf; version=1.3',
  'dc:title': 'This is a test PDF file',
  'dcterms:created': '2000-06-28T23:21:08Z',
  'dcterms:modified': '2013-10-28T19:24:13Z',
  'meta:author': 'cdaily',
  'meta:creation-date': '2000-06-28T23:21:08Z',
  'meta:save-date': '2013-10-28T19:24:13Z',
  'modified': '2013-10-28T19:24:13Z',
  'pdf:PDFVersion': '1.3',
  'pdf:docinfo:created': '2000-06-28T23:21:08Z',
  'pdf:docinfo:creator': 'cdaily',
  'pdf:docinfo:creator_tool': 'Microsoft Word 8.0',
  'pdf:docinfo:modified': '2013-10-28T19:24:13Z',
  'pdf:docinfo:producer': 'Acrobat Distiller 4.0 for Windows',
  'pdf:docinfo:title': 'This is a test PDF file',
  'pdf:encrypted': 'false',
  'producer': 'Acrobat Distiller 4.0 for Windows',
  'resourceName': 'pdf-sample.pdf',
  'title': 'This is a test PDF file',
  'xmp:CreatorTool': 'Microsoft Word 8.0',
  'xmpMM:DocumentID': 'uuid:0805e221-80a8-459e-a522-635ed5c1e2e6',
  'xmpTPg:NPages': '1'}}
```

In [20]:
```
print(parsed['content'].strip())
```

```
This is a test PDF file


Adobe Acrobat PDF Files

Adobe® Portable Document Format (PDF) is a universal file format that preserves all
of the fonts, formatting, colours and graphics of any source document, regardless of
the application and platform used to create it.

Adobe PDF is an ideal format for electronic document distribution as it overcomes the
problems commonly encountered with electronic file sharing.

•  Anyone, anywhere can open a PDF file. All you need is the free Adobe Acrobat
Reader. Recipients of other file formats sometimes can't open files because they
don't have the applications used to create the documents.

•  PDF files always print correctly on any printing device.

•  PDF files always display exactly as created, regardless of fonts, software, and
operating systems. Fonts, and graphics are not lost due to platform, software, and
version incompatibilities.

•  The free Acrobat Reader is easy to download and can be freely distributed by
anyone.

•  Compact PDF files are smaller than their source files and download a
page at a time for fast display on the Web.
```

Not sure why we get the title of the PDF inside the content. Anyway the text is extracted properly and we even get a lot of metadata:

In [21]:
```
parsed['metadata']
```

Out[21]:
```
{'Author': 'cdaily',
 'Content-Type': 'application/pdf',
 'Creation-Date': '2000-06-28T23:21:08Z',
 'Last-Modified': '2013-10-28T19:24:13Z',
 'Last-Save-Date': '2013-10-28T19:24:13Z',
 'X-Parsed-By': ['org.apache.tika.parser.DefaultParser',
  'org.apache.tika.parser.pdf.PDFParser'],
 'X-TIKA:parse_time_millis': '62',
 'access_permission:assemble_document': 'true',
 'access_permission:can_modify': 'true',
 'access_permission:can_print': 'true',
 'access_permission:can_print_degraded': 'true',
 'access_permission:extract_content': 'true',
 'access_permission:extract_for_accessibility': 'true',
 'access_permission:fill_in_form': 'true',
 'access_permission:modify_annotations': 'true',
 'created': 'Wed Jun 28 23:21:08 UTC 2000',
 'creator': 'cdaily',
 'date': '2013-10-28T19:24:13Z',
 'dc:creator': 'cdaily',
 'dc:format': 'application/pdf; version=1.3',
 'dc:title': 'This is a test PDF file',
 'dcterms:created': '2000-06-28T23:21:08Z',
 'dcterms:modified': '2013-10-28T19:24:13Z',
 'meta:author': 'cdaily',
 'meta:creation-date': '2000-06-28T23:21:08Z',
 'meta:save-date': '2013-10-28T19:24:13Z',
 'modified': '2013-10-28T19:24:13Z',
 'pdf:PDFVersion': '1.3',
 'pdf:docinfo:created': '2000-06-28T23:21:08Z',
 'pdf:docinfo:creator': 'cdaily',
 'pdf:docinfo:creator_tool': 'Microsoft Word 8.0',
 'pdf:docinfo:modified': '2013-10-28T19:24:13Z',
 'pdf:docinfo:producer': 'Acrobat Distiller 4.0 for Windows',
 'pdf:docinfo:title': 'This is a test PDF file',
 'pdf:encrypted': 'false',
 'producer': 'Acrobat Distiller 4.0 for Windows',
 'resourceName': 'pdf-sample.pdf',
 'title': 'This is a test PDF file',
 'xmp:CreatorTool': 'Microsoft Word 8.0',
 'xmpMM:DocumentID': 'uuid:0805e221-80a8-459e-a522-635ed5c1e2e6',
 'xmpTPg:NPages': '1'}
```

Conclusion

We saw different methods to extract text from PDF in Python. Depending on what you want to do, one might suit you better. And this was of course not exhaustive.

If you want to index PDFs, Elasticsearch might be all you need. The ingest-attachment plugin uses Apache Tika which is very powerful.

And thanks to Tika-Python, it's very easy to use Tika directly from Python. You can let the library starts the server or use Docker to start your own.

elasticsearch python tika

Previous post

Next post

Comments

ALSO ON TCHUT-TCHUT BLOG

Experimenting with asyncio on a ...
8 years ago · 4 comments
In a previous post, I described how I built a LEGO Macintosh Classic ...

Control your accessories from ...
8 years ago · 3 comments
While reading more about Home Assistant, I discovered it was ...

My LEGO Macintosh Classic with ...
8 years ago · 7 comments
UPDATED 2019-11-24 Beginning of April I read an inspiring blog post from ...

Logging to a Tkinter ScrolledText Widget
7 years ago · 6 comments
I've been programming in Python for almost 10 years. I did many CLI tools, ...

OpenVPN sou
routing
8 years ago · 11 cc
I already spoke a
installing OpenV
Raspberry Pi in a

2 Comments

1 Login ▾

G Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

♡ 2 Share

Best Newest Oldest

disqus_ofwfLzMnJV
7 years ago edited
Hey, did you know why my indexed documents are not persisting after reboot or restart the docker image?

14 0 Reply Share ›

B Benjamin Bertrand Mod ➦ disqus_ofwfLzMnJV
7 years ago edited
Data in a docker container is not persistent. Check https://docs.docker.com/sto...
If you want to save the data between restart, you should either use:
- a volume: https://docs.docker.com/sto...
- or bind mount: https://docs.docker.com/sto...

0 0 Reply Share ›

Subscribe Privacy Do Not Sell My Data

Contents © 2024 Benjamin Bertrand - Powered by Nikola