Extraction_with_Comment

March 28, 2019

1 Table of Contents

1.0.1 1. Environment Information

- show the python version and system for this computer
- help others who use this code to produce replicable result

```
In [1]: import IPython

        # Information of my Python version, computer system
        print(IPython.sys_info())
```

```
{'commit_hash': '523ed2fe5',
 'commit_source': 'installation',
 'default_encoding': 'UTF-8',
 'ipython_path': '/anaconda/envs/nlp/lib/python3.6/site-packages/IPython',
 'ipython_version': '7.2.0',
 'os_name': 'posix',
 'platform': 'Darwin-18.5.0-x86_64-i386-64bit',
 'sys_executable': '/anaconda/envs/nlp/bin/python',
 'sys_platform': 'darwin',
 'sys_version': '3.6.8 |Anaconda, Inc.| (default, Dec 29 2018, 19:04:46) \n'
                '[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]'}
```

1.0.2 2.Read PDF files

```
In [2]: # import pdfminer and to read PDF files
        from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
        from pdfminer.converter import TextConverter
        from pdfminer.layout import LAParams
        from pdfminer.pdfpage import PDFPage

        # import io (input and output); BytesIO encode string to byte object
        from io import BytesIO

        # extract all file name in a folder, for the convenience of reading PDF files
        import glob

        # latter use to store cleaned string into json file
        import simplejson as json

        # re (regular expression) to find string with certain patterns
        import re
```

A function to read PDF file:

```
pdf_file: the filename of PDF (including the path (i.e location) )
return: contentt of the PDF (string in Byte object,
remember we use BytesIO to encode our string result)
```

```
In [3]: def read_pdf(pdf_file):

            resource_mgr = PDFResourceManager()
            retstr = BytesIO()
            codec = 'utf-8'
            laparams = LAParams()
            device = TextConverter(resource_mgr, retstr, codec=codec, laparams=laparams)
            interpreter = PDFPageInterpreter(resource_mgr, device)
            fp = open(pdf_file, 'rb')
            maxpages = 0
            caching = True
            pagenos = set()

            for page in PDFPage.get_pages(fp, pagenos,
                                          maxpages=maxpages,
                                          caching=caching,
                                          check_extractable=True):
                interpreter.process_page(page)

            result_str = retstr.getvalue()

            fp.close()
            device.close()
            retstr.close()

            return result_str
```

Create a list contains all the PDF files we want to read

```
In [224]: case_list = glob.glob('./Cases/*.pdf')
```

We can print the PDF files list to inspect.
The list contains all the PDF files name in a folder

```
In [227]: case_list[0:10]
```

```
Out[227]: ['./Cases/052 - Nuance Communications Inc v ABBYY USA Software House Inc.pdf',
           './Cases/021 - Intellectual Ventures I LLC v Erie Indemnity Company.pdf',
           "./Cases/040 - Sioux Honey Ass'n v Hartford Fire Ins Co.pdf",
           './Cases/021 - Aspex Eyewear Inc v Zenni Optical Inc.pdf',
           './Cases/056 - Soverain Software LLC v Newegg Inc.pdf',
           './Cases/100 - In re Geller.pdf',
           './Cases/08 - Boeri v US.pdf',
           './Cases/080 - Harmonic Inc v Avid Technology Inc.pdf',
           './Cases/095 - Ethicon EndoSurgery Inc v Covidien LP.pdf',
           './Cases/039 - OIP Technologies Inc v Amazoncom Inc.pdf']
```

A for loop to read all the PDF files in `case_list`

```
In [228]: # a list to store the name of the pdf
          pdf_name = []
          # a list to store the content of the pdf
          pdf_content = []

          # a loop to read all the pdf and store their name and content to the respective list
          for case in case_list:
              pdf_name.append(case.replace('./Cases/', '').replace('.pdf', ''))
              pdf_content.append(read_pdf(case))
```

```
In [229]: # we can inspect the pdf names
          pdf_name[0:10]
```

```
Out[229]: ['./Cases/052 - Nuance Communications Inc v ABBYY USA Software House Inc',
           './Cases/021 - Intellectual Ventures I LLC v Erie Indemnity Company',
           "./Cases/040 - Sioux Honey Ass'n v Hartford Fire Ins Co",
           './Cases/021 - Aspex Eyewear Inc v Zenni Optical Inc',
           './Cases/056 - Soverain Software LLC v Newegg Inc',
           './Cases/100 - In re Geller',
           './Cases/08 - Boeri v US',
           './Cases/080 - Harmonic Inc v Avid Technology Inc',
           './Cases/095 - Ethicon EndoSurgery Inc v Covidien LP',
           './Cases/039 - OIP Technologies Inc v Amazoncom Inc']
```

```
In [230]: # the first 300 cahracters of'001 - Intellectual Ventures I LLC v Motorola Mobility LLC'
          pdf_content[0][:300]
```

```
Out[230]: b'Nuance Communications, Inc. v. ABBYY USA Software House, Inc., 813 F.3d 1368 (2016)\n117 U.S.P.Q.2d 1944\n\n813 F.3d 1368\n\nU
```

Above is the content of the first pdf file ('001 - Intellectual Ventures I LLC v Motorola Mobility LLC')

You can compare the PDF file with the string above, basically, python process the PDF file into raw string format, e.g `\n` means a new line, `\xe2\x80\x93` is a maker of byte encoding

Note the first letter b". . . ..(content)" indicate this is a byte string (remember we use BytesIO to specify the string to be stored in a byte object, for the convenience of encoding.)

1.0.3 3. Extract Citation

There are multiple citation (references) in a legal court case, come with a format like `870 F.3d 1320`

- `870` is the volumn
- `F.3d` is the reporter
- `1320` is the page

`lexnlp` is a module deal with legal domain-specific text, we can use `lexnlp` to help us extract citations from the pdf content.
For more information about `lexnlp`, go to the documentation website: https://lexpredict-lexnlp.readthedocs.io/en/docs-0.1.6/index.html
`pandas` (i.e. Python Data Analysis Library) is a module to manipulate data tables. After extract information from the PDF contents, `pandas` is used to create data frame to store the information.

```
In [231]: # import the function in lexnlp module to extract citations
          import lexnlp.extract.en.citations

          # store extracted information into dataframe
          import pandas as pd
```

Before the extraction, we have to decode out byte string object to literal string. Remember that our PDF content come with `b`". . . ..(literal content)", since python read the PDF to byte object

```
In [232]: court_text = []
          for content in pdf_content:
              # decode every pdf file content, sepecify the decode style "utf-8"
              # p.s. "utf-8" is the most common encoding sytle today
              decoded_content = content.decode("utf-8")
              court_text.append(decoded_content)
```

```
In [233]: # Decoded the first 300 cahracters of
          # '001 - Intellectual Ventures I LLC v Motorola Mobility LLC'
          court_text[0][:300]
```

```
Out[233]: 'Nuance Communications, Inc. v. ABBYY USA Software House, Inc., 813 F.3d 1368 (2016)\n117 U.S.P.Q.2d 1944\n\n813 F.3d 1368\n\nUn
```

After decoding, we can see the `b` was removed (the decoding is more than removing the `b` behind the scenes, `b` is a notation for *byte* object)

```
In [234]: cite_list = []
          for text in court_text:
              text_citation =list(lexnlp.extract.en.citations.get_citations(text,
                                                                             return_source=False,
                                                                             as_dict=True))

              # cite_in_pdf is a list contains all the citations in a pdf file
              cite_in_pdf = []
              for cite in text_citation:
                  # sotre every citation in the pdf file to this container
                  cite_in_pdf.append(str(cite['volume']) + " " + cite['reporter'] +
                                     " " + str(cite['page']))

                  # remove duplicated citations in a file
                  cite_in_pdf = list(dict.fromkeys(cite_in_pdf))

              # cite_list is a list contatins lists of citations for pdf files in the folder
              cite_list.append(cite_in_pdf)
```

```
In [235]: # citations in the first pdf files
          cite_list[0]
```

```
Out[235]: ['813 F.3d 1368',
           '805 F.3d 1368',
           '467 F.3d 1355',
           '299 F.3d 1313',
           '639 F.3d 1303']
```

1.0.4 4. Case Number of the PDF file

```
In [236]: # A list contain the case number of the PDF file
          case_number = []
          for cites in cite_list:
              # Case number is the first citation number in the citation list
              case_number.append(cites[0])
```

```
In [263]: case_number[0:6]
```

```
Out[263]: ['813 F.3d 1368',
            '850 F.3d 1315',
            '672 F.3d 1041',
            '713 F.3d 1377',
            '728 F.3d 1332',
            '751 F.3d 1355']
```

1.0.5 5. Extract Regulations

```
In [238]: # import the function in lexnlp module to extract regulations
          import lexnlp.extract.en.regulations
```

```
In [239]: reg_list = []
          for text in court_text:
              text_regulation =list(lexnlp.extract.en.regulations.get_regulations(text,
                                                                                     return_source=False,
                                                                                    as_dict=True))

              # reg_in_pdf is a list contains all the regulations in a pdf file
              reg_in_pdf = []
              for reg in text_regulation:

                  # sotre every regulation in the pdf file to this container
                  reg_in_pdf.append(reg['regulation_code'])

                  # remove duplicated regulations in a file
                  reg_in_pdf = list(dict.fromkeys(reg_in_pdf))

              # reg_list is a list contatins lists of regulations for pdf files in the folder
              reg_list.append(reg_in_pdf)
```

```
In [240]: # regulations in the first pdf file
          reg_list[0]
```

```
Out[240]: ['28 USC § 1295']
```

1.0.6 6. Extract Courts

```
In [241]: # Load court configuration data automatically from LexPredict legal dictionaries
          from lexnlp.extract.en.dict_entities import get_entity_name, entity_config
          import lexnlp.extract.en.courts
          import pandas
          text = court_text[0]
          court_df = pandas.read_csv("https://raw.githubusercontent.com/LexPredict/lexpredict-legal-dictionary/1.0.5/en/legal/us_courts.cs
          # Create config objects

          court_config_data = []
          for _, row in court_df.iterrows():
              c = entity_config(row["Court ID"],row["Court Name"],row["Court Type"])
              court_config_data.append(c)

          court_in_pdf = []
          for entity in lexnlp.extract.en.courts.get_courts(text, court_config_data):
              court_in_pdf.append(entity[0][1:3])
              # remove duplicated regulations in a file
              court_in_pdf = list(dict.fromkeys(court_in_pdf))

In [242]: court_list = []
          for text in court_text:
              text_court =lexnlp.extract.en.courts.get_courts(text, court_config_data)

              # court_in_pdf is a list contains all the courts in a pdf file
              court_in_pdf = []
              for entity in text_court:

                  # sotre every court in the pdf file to this container
                  court_in_pdf.append(entity[0][1:3])

                  # remove duplicated courts in a file
                  court_in_pdf = list(dict.fromkeys(court_in_pdf))

              # reg_list is a list contatins lists of regulations for pdf files in the folder
              court_list.append(court_in_pdf)

In [262]: court_list[0:6]

Out[262]: [[('Federal Circuit', 'Appellate Court'),
            ('Northern District of California', 'Federal District Court'),
            ('Northern District of California', 'Bankruptcy Court')],
           [('Federal Circuit', 'Appellate Court'),
            ('Western District of Pennsylvania', 'Federal District Court'),
            ('Western District of Pennsylvania', 'Bankruptcy Court'),
            ('Third Circuit', 'Appellate Court')],
           [('Federal Circuit', 'Appellate Court'),
            ('United States Court of International Trade', 'Federal Court'),
            ('International Trade Commission', 'Federal Court')],
           [('Federal Circuit', 'Appellate Court'),
            ('Southern District of Florida', 'Federal District Court'),
            ('Southern District of Florida', 'Bankruptcy Court'),
            ('Eleventh Circuit', 'Appellate Court')],
           [('Federal Circuit', 'Appellate Court'),
            ('Eastern District of Texas', 'Federal District Court'),
            ('Eastern District of Texas', 'Bankruptcy Court')],
           [('Federal Circuit', 'Appellate Court'),
            ('Trademark Trial and Appeal Board', 'Federal Court')]]
```

1.0.7 7. Court for this PDF file

```
In [244]: # A list contain the courts appeared in the PDF file
          court_for_this_file = []
          for courts in court_list:
              # Court for this PDF file is the first court in the court list
              court_for_this_file.append(courts[0])
```

```
In [260]: court_for_this_file[0:10]
```

```
Out[260]: [('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court'),
           ('Federal Circuit', 'Appellate Court')]
```

1.0.8 8. Extract Date

```
In [246]: import lexnlp.extract.en.dates
In [254]: dates_for_pdf = []
          for text in court_text:
              date_list =list(lexnlp.extract.en.dates.get_dates(text))
              if len(date_list)>=1:
                  dates_for_pdf.append(date_list[0])
              else:
                  dates_for_pdf.append(None)
```

```
In [259]: dates_for_pdf[0:10]
```

```
Out[259]: [datetime.date(2016, 2, 22),
           datetime.date(2017, 3, 7),
           datetime.date(2012, 2, 7),
           datetime.date(2013, 4, 19),
           datetime.date(2035, 2, 1),
           datetime.date(2014, 5, 13),
           datetime.date(2013, 2, 1),
           datetime.date(2016, 3, 1),
           datetime.date(2016, 6, 22),
           datetime.date(2015, 6, 11)]
```

1.0.9 9. Case No `xxxx-xxxx`

```
In [266]: case_num_xxx = []
          pattern = re.compile('[0-9]{4}[-|-][0-9]{3,4}', flags=re.DOTALL)
          for text in court_text:
              num = re.findall(pattern,text)
              if len(num)>=1:
                  case_num_xxx.append(num[0])
              else:
                  case_num_xxx.append(None)
```

```
In [269]: len(case_num_xxx)
```

```
Out[269]: 159
```

1.0.10 synopsis (and other titled part)

```
In [160]: synopsis = []
          pattern = re.compile('\nBackground:\s(.*?)\n\n', flags=re.DOTALL)
          for text in court_text:
              background = re.search(pattern,text).group(1)
              synopsis.append(background)
```

1.1 10. Wrap up to Data Frame

So for we have - `pdf_name`, - `cite_list`, - `case_number`, - `reg_list`, - `court_list`, - `court_for_this_file`, - `dates_for_pdf`, - `case_num_xxx`

```
In [277]: # all the lists contains extracted information
          varlist =[pdf_name,
          cite_list,
          case_number,
          reg_list,
          court_list,
          court_for_this_file,
          dates_for_pdf,
          case_num_xxx]
```

```
In [278]: # check the length of the lists
          for var in varlist:
              print(len(var))
```

```
159
159
159
159
159
159
159
159
```

A data frame contains the meta data of the court case in the PDF file

```
In [279]: metadata = pd.DataFrame(
              {'file_name': pdf_name,
               'case_number': case_number,
               'court':court_for_this_file,
               'dates':dates_for_pdf,
               'num_x_x':case_num_xxx
              })
```

```
In [280]: metadata.head()
```

```
Out[280]:                                            file_name    case_number  \
          0  052 - Nuance Communications Inc v ABBYY USA So...   813 F.3d 1368
          1  021 - Intellectual Ventures I LLC v Erie Indem...   850 F.3d 1315
          2     040 - Sioux Honey Ass'n v Hartford Fire Ins Co   672 F.3d 1041
          3        021 - Aspex Eyewear Inc v Zenni Optical Inc   713 F.3d 1377
          4           056 - Soverain Software LLC v Newegg Inc   728 F.3d 1332

                                          court       dates    num_x_x
          0  (Federal Circuit, Appellate Court)  2016-02-22  2014-1629
          1  (Federal Circuit, Appellate Court)  2017-03-07  2016-1128
          2  (Federal Circuit, Appellate Court)  2012-02-07  2011-1040
          3  (Federal Circuit, Appellate Court)  2013-04-19  2012-1318
          4  (Federal Circuit, Appellate Court)  2035-02-01  2011-1009
```

Save the data frame to CSV file

```
In [286]: metadata.to_csv("meta_file.csv", sep='\t', encoding='utf-8')
```

A data frame contains the reference list of the court case in the PDF file

```
In [282]: reference_extract = pd.DataFrame(
             {'file_name': pdf_name,
              'citations': cite_list,
              'regulations': reg_list,
              'courts':court_list,
             })
```

```
In [283]: reference_extract.head()
```

```
Out[283]:                                            file_name  \
          0  052 - Nuance Communications Inc v ABBYY USA So...
          1  021 - Intellectual Ventures I LLC v Erie Indem...
          2     040 - Sioux Honey Ass'n v Hartford Fire Ins Co
```

```
3        021 - Aspex Eyewear Inc v Zenni Optical Inc
4           056 - Soverain Software LLC v Newegg Inc

                                           citations  \
0  [813 F.3d 1368, 805 F.3d 1368, 467 F.3d 1355, ...
1  [850 F.3d 1315, 444 F.3d 1337, 718 P.2d 920, 5...
2  [672 F.3d 1041, 200 F.3d 1369, 517 F.3d 1319, ...
3  [713 F.3d 1377, 439 U.S. 322, 342 F.3d 1320, 6...
4  [728 F.3d 1332, 705 F.3d 1333, 2013 WL 2631445...

                                         regulations  \
0                                     [28 USC § 1295]
1        [35 USC § 101, 28 USC § 1295, 35 USC § 261]
2  [19 USC § 1675c, 19 USC § 1675, 28 USC § 1295,...
3                                                  []
4                                                  []

                                              courts
0  [(Federal Circuit, Appellate Court), (Northern...
1  [(Federal Circuit, Appellate Court), (Western ...
2  [(Federal Circuit, Appellate Court), (United S...
3  [(Federal Circuit, Appellate Court), (Southern...
4  [(Federal Circuit, Appellate Court), (Eastern ...
```

```
In [285]: reference_extract.to_csv("reference_extract.csv", sep='\t', encoding='utf-8')
```