Mining scientific articles

with XML, Python, and allofplos

Elizabeth Seiver
plos.org (http://plos.org), @tweetotaler (http://twitter.com/tweetotaler)
PLOS and The Hacker Within
Wed November 29, 2017

Why Mine Scientific Articles?

- Science articles represent scientific knowledge
- XML is version of articles for machines, while PDF is for humans
- Tool for meta-research and meta-science
- Quickly identify sets of articles of interest
- Identify research literature trends over time (study findings, jargon usage, citation networks)

PLOS corpus of articles

- 220,000+ scientific articles from a wide array of research fields, focusing on the medical and life sciences
- Since 2003
- Open Access: free to read, free to re-use
- Creative Commons license (CC-BY, CC0)
- Many scientific articles are behind a paywall

Tutorial plan

- Goal: to enable research questions about science articles
- Will use JupyterHub and a sample corpus of 10,000 randomly selected PLOS articles
- Won't be discussing research techniques (data analysis, natural language processing)
- Assumes basic Python knowledge (lists, dictionaries, loops, conditionals, datetime)

Tutorial structure

1. How to parse XML using allofplos and lxml (http://lxml.de/tutorial.html)
2. Basic structure of XML documents and JATS standard (https://jats.nlm.nih.gov/)
3. Example projects with the PLOS test corpus
4. Hacking session: parse articles or contribute to the allofplos codebase

Exercises based on tutorial: https://github.com/eseiver/xml_tutorial (https://github.com/eseiver/xml_tutorial)

allofplos

- Python package for both downloading and parsing PLOS XML articles
- Turns PLOS XML articles into Python data structures
- Doesn't require knowledge of XML to use
- Focuses on article metadata (e.g., title, authors, date of publication)
- Work in progress, so aspects of it may change

allofplos basics

- Initialize an article object w/DOI or XML filename
  - DOI (Digital Object Identifier) is a unique identifier for an online document/article
  - All PLOS DOIs start with `"10.1371/journal."`, like `"10.1371/journal.pone.0185809"`
- allofplos XML files are named with last part of DOI, e.g. `"journal.pone.0185809.xml"`

```
from allofplos import Article # if have run `pip install allofplos`
# from article_class import Article  # if inside cloned GitHub directory

# first instantiation of Article class by DOI
article = Article('10.1371/journal.pone.0178690')
article.title
```

```
'Physician assessments of drug seeking behavior: A
mixed methods study'
```

```
# first instantiation of Article class by filename
article = Article.from_filename('allofplos_xml/journal.pone.0181748.xml')
article.title
```

```
'THPdb: Database of FDA-approved peptide and protein therapeutics'
```

```
# new article
article.doi = '10.1371/journal.pone.0183591'
article.title
```

```
'A checklist is associated with increased quality o
f reporting preclinical biomedical research: A syst
ematic review'
```

Notable properties

Try printing or returning some of these values

Basic metadata

```
article.doi
```

```
'10.1371/journal.pone.0183591'
```

```
article.journal
```

```
'PLOS ONE'
```

```
article.pubdate
```

```
datetime.datetime(2017, 9, 13, 0, 0)
```

```
article.title
```

```
'A checklist is associated with increased quality o
f reporting preclinical biomedical research: A syst
ematic review'
```

```
article.counts
```

```
{'fig-count': '3', 'page-count': '14', 'table-coun
t': '2'}
```

```
article.word_count
```

```
4954
```

People

```
contributor = article.contributors[0]
contributor.keys()
```

```
dict_keys(['contrib_initials', 'given_names', 'surn
ame', 'group_name', 'ids', 'rid_dict', 'contrib_typ
e', 'author_type', 'editor_type', 'email', 'affilia
tions', 'author_roles', 'footnotes'])
```

```
article.authors[0]
```

```
{'affiliations': ['Division of Pulmonary, Allergy,
and Critical Care Medicine, Department of Medicin
e, University of Pittsburgh, Pittsburgh, Pennsylva
nia, United States of America'],
 'author_roles': {'CASRAI CREDiT taxonomy': ['Conc
eptualization',
   'Data curation',
   'Formal analysis',
   'Funding acquisition',
   'Investigation',
   'Writing — original draft',
   'Writing — review & editing']},
 'author_type': 'corresponding',
 'contrib_initials': 'SH',
 'contrib_type': 'author',
 'editor_type': None,
 'email': ['shan.workmd@gmail.com'],
 'footnotes': ['Current address: Division of Pulmo
nary and Critical Care, Department of Medicine, No
rthwestern University, Chicago, Illinois, United S
tates of America'],
 'given_names': 'SeungHye',
```

```
'group_name': None,
'ids': [{'authenticated': 'true',
  'id': 'http://orcid.org/0000-0001-5625-6337',
  'id_type': 'orcid'}],
'rid_dict': {'aff': ['aff001'],
 'corresp': ['cor001'],
 'fn': ['currentaff001']},
'surname': 'Han'}
```

```
article.corr_author
```

```
[{'affiliations': ['Division of Pulmonary, Allerg
y, and Critical Care Medicine, Department of Medic
ine, University of Pittsburgh, Pittsburgh, Pennsyl
vania, United States of America'],
  'author_roles': {'CASRAI CREDiT taxonomy': ['Con
ceptualization',
    'Data curation',
    'Formal analysis',
    'Funding acquisition',
    'Investigation',
    'Writing – original draft',
    'Writing – review & editing']},
  'author_type': 'corresponding',
  'contrib_initials': 'SH',
  'contrib_type': 'author',
  'editor_type': None,
  'email': ['shan.workmd@gmail.com'],
  'footnotes': ['Current address: Division of Pulm
onary and Critical Care, Department of Medicine, N
orthwestern University, Chicago, Illinois, United
States of America'],
  'given_names': 'SeungHye',
```

```
'group_name': None,
'ids': [{'authenticated': 'true',
  'id': 'http://orcid.org/0000-0001-5625-6337',
  'id_type': 'orcid'}],
'rid_dict': {'aff': ['aff001'],
 'corresp': ['cor001'],
 'fn': ['currentaff001']},
'surname': 'Han'}]
```

```
article.editor[0]
```

```
{'affiliations': ['Fraunhofer Research Institution
of Marine Biotechnology, GERMANY'],
 'author_roles': {None: ['Editor']},
 'author_type': None,
 'contrib_initials': 'JB',
 'contrib_type': 'editor',
 'editor_type': None,
 'email': None,
 'footnotes': [],
 'given_names': 'Johannes',
 'group_name': None,
 'ids': [],
 'rid_dict': {'aff': ['edit1']},
 'surname': 'Boltze'}
```

Article type

```
article.type_  # JATS
```

```
'research-article'
```

```
article.plostype
```

```
'Research Article'
```

```
article.proof  # whether an uncorrected proof/early version or not
```

Local article file (more on this later)

```
article.filename
```

```
'/Users/Elizabeth/PLOS_Corpus_Project/allofplos/all
ofplos/allofplos_xml/journal.pone.0183591.xml'
```

```
article.local
```

```
True
```

```
article.tree
```

```
<lxml.etree._ElementTree at 0x10efa9dc8>
```

```
article.root
```

```
<Element article at 0x1118cf348>
```

```
article.xml
```

Other notable methods

```
article.get_dates()
```

```
{'accepted': datetime.datetime(2017, 8, 7, 0, 0),
 'collection': datetime.datetime(2017, 1, 1, 0, 0),
 'epub': datetime.datetime(2017, 9, 13, 0, 0),
 'received': datetime.datetime(2017, 3, 19, 0, 0)}
```

```
article.check_if_doi_resolves()
```

```
'works'
```

```
article
```

```
DOI: 10.1371/journal.pone.0183591
Title: A checklist is associated with increased qua
lity of reporting preclinical biomedical research:
A systematic review
```

Primer on XML elements

Why use lxml.etree for parsing XML documents?

- JATS standard for scientific article XML is consistent
- BeautifulSoup better for unreliable web documents
- lxml.etree and BeautifulSoup each have a module of the other
- The following XML element examples are derived from actual PLOS articles

Primer on XML elements

XML elements have four key properties in the lxml.etree library

1. element.tag
2. element.text
3. element.attrib
4. element.tail

Example basic element

```
<article-title>Why Most Published Research Findings
 Are False</article-title>
```

```
element.tag
```

```
'article-title'
```

```
element.text
```

```
'Why Most Published Research Findings Are False'
```

Basic element with attribute

```
<alt-title alt-title-type="running-head">Essay</alt
-title>
```

```
element.tag
```

```
'alt-title'
```

```
element.text
```

```
'Essay'
```

```
element.attrib
```

```
{'alt-title-type': 'running-head'}
```

```
# for any text that comes directly after closing tag and before a
nother tag
element.tail
```

An element attribute is a dictionary

Appears inside the element tag

```
XML(element)
```

```
<alt-title alt-title-type="running-head">Essay</alt
-title>
```

```
element.attrib
```

```
{'alt-title-type': 'running-head'}
```

```
element.attrib['alt-title-type']
```

```
'running-head'
```

```
element.attrib.get('alt-title-type')
```

```
'running-head'
```

XML elements can have sub-elements

```
<title-group>
  <article-title>Why Most Published Research Findin
gs Are False</article-title>
  <alt-title alt-title-type="running-head">Essay</a
lt-title>
</title-group>
```

```
# to find direct descendants; don't need to know their tags
element.getchildren()
```

```
[<Element article-title at 0x10ab16888>, <Element a
lt-title at 0x1087d06c8>]
```

```
new_element = element.getchildren()[0]
new_element.tag
```

```
'article-title'
```

```
# to find direct ancestor; don't need to know its tag
new_element.getparent()
```

```
<Element title-group at 0x10ab16588>
```

Finding sub-elements by name with xpath

Example element: Creative Commons License

```
<license xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:mml="http://www.w3.org/1998/Math/MathML" xli
nk:href="http://creativecommons.org/licenses/by/4.
0/" xlink:type="simple">
  <license-p>This is an open access article distrib
uted under the terms of the <ext-link ext-link-type
="uri" xlink:href="http://creativecommons.org/licen
ses/by/4.0/" xlink:type="simple">Creative Commons A
ttribution License</ext-link>, which permits unrest
ricted use, distribution, and reproduction in any m
edium, provided the original author and source are
 credited.</license-p>
</license>
```

Xpath returns a list of search results

```
# search direct descendants by name
license.xpath('./license-p')
```

```
[<Element license-p at 0x10ab47488>]
```

```
# search descendants of direct descendants
license.xpath('./license-p/ext-link')
```

```
[<Element ext-link at 0x10ab445c8>]
```

```
# search ALL descendants
license.xpath('.//ext-link')
```

```
[<Element ext-link at 0x10ab445c8>]
```

Warning 1: Multiple elements can have the same xpath location

Remember that it always returns a list

```
<contrib-group>
  <contrib contrib-type="author">
    <name name-style="western">
      <surname>dos Santos</surname>
      <given-names>Renato Vieira</given-names>
    </name>
  </contrib>
  <contrib contrib-type="author">
    <name name-style="western">
      <surname>da Silva</surname>
      <given-names>Linaena Mericy</given-names>
    </name>
  </contrib>
</contrib-group>
```

```
element.xpath('./contrib')
```

```
[<Element contrib at 0x10ab44608>, <Element contrib
at 0x10ab4d508>]
```

Warning 2: element.text doesn't always work

When in doubt, use `lxml.etree.tostring()`

```
<license xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:mml="http://www.w3.org/1998/Math/MathML" xli
nk:href="http://creativecommons.org/licenses/by/4.
0/" xlink:type="simple">
  <license-p>This is an open access article distrib
uted under the terms of the <ext-link ext-link-type
="uri" xlink:href="http://creativecommons.org/licen
ses/by/4.0/" xlink:type="simple">Creative Commons A
ttribution License</ext-link>, which permits unrest
ricted use, distribution, and reproduction in any m
edium, provided the original author and source are
 credited.</license-p>
</license>
```

```
print(license.text)
```

```
None
```

```
import lxml.etree as et
license_text = et.tostring(license, method='text', encoding='unic
ode')
print(license_text)
```

```
This is an open access article distributed under th
e terms of the Creative Commons Attribution Licens
e, which permits unrestricted use, distribution, an
d reproduction in any medium, provided the original
author and source are credited.
```

Quick quiz before we move on!

```
<contrib contrib-type="author" equal-contrib="yes">
    <name name-style="western">
        <surname>Chen</surname>
        <given-names>Ximing</given-names>
    </name>plossy
</contrib>
```

- element.tag?
- element.attrib?

```
element.tag: contrib
element.attrib: {'contrib-type': 'author', 'equal-c
ontrib': 'yes'}
```

Quick quiz before we move on!

```
<contrib contrib-type="author" equal-contrib="yes">
    <name name-style="western">
        <surname>Chen</surname>
        <given-names>Ximing</given-names>
    </name>plossy
</contrib>
```

```
new_element = element.xpath('./name')[0]
```

- new_element.tag?
- new_element.attrib?
- new_element.tail?

```
new_element.tag: name
new_element.attrib: {'name-style': 'western'}
new_element.tail: plossy
```

Using lxml.etree and allofplos Article class to parse XML files

```
from allofplos import Article
doi = '10.1371/journal.pone.0183591'
article = Article(doi)
article.filename
```

```
'/Users/Elizabeth/PLOS_Corpus_Project/allofplos/all
ofplos/allofplos_xml/journal.pone.0183591.xml'
```

```
# Is the article XML file locally stored?
article.local
```

```
True
```

```
tree = article.tree
tree.xpath('./body')
```

```
[<Element body at 0x10b176a48>]
```

```
xml_root = article.root
xml_root.xpath('.//license')
```

```
[<Element license at 0x10ab58488>]
```

```
license = xml_root.xpath('.//license')[0]
license.attrib
```

```
{'{http://www.w3.org/1999/xlink}href': 'http://crea
tivecommons.org/licenses/by/4.0/', '{http://www.w3.
org/1999/xlink}type': 'simple'}
```

Tying it all together!

Project example: which articles use PCR in their Methods section?

First, on finding elements with xpath searching tag AND attribute

- Body of article is divided into sections ('sec')
- Method section attribute of note: `{'sec-type': 'materials|methods'}` or `{'sec-type': 'methods'}`

```
methods_sections = xml_root.xpath("//sec[@sec-type='materials|methods']")
print(methods_sections)
```

```
[<Element sec at 0x10b176b48>]
```

```
from allofplos.samples.corpus_analysis import get_random_list_of_
dois
from allofplos.article_class import Article
import lxml.etree as et

# First get list of articles/DOIs
dois = get_random_list_of_dois(count=50)
pcr_list = []
# Initialize first article object
article = Article(dois[0])

for doi in dois:
    # Step 1: create new article object
    article.doi = doi
    xml_root = article.root
    # Step 2: find Method sections
    methods_sections = xml_root.xpath("//sec[@sec-type='materials
|methods']")
    if not methods_sections:
        methods_sections = xml_root.xpath("//sec[@sec-type='metho
ds']")
    for sec in methods_sections:
        # Step 3: turn the method sections into strings
        method_string = et.tostring(sec, method='text', encoding=
```

```
'unicode')
        # Step 4: add DOI if 'PCR' in string
        if 'PCR' in method_string:
            pcr_list.append(article.doi)
            break
        else:
            pass

print(pcr_list[0:5])
```

```
['10.1371/journal.pone.0128195', '10.1371/journal.p
one.0165464', '10.1371/journal.pone.0136574', '10.1
371/journal.pone.0072749', '10.1371/journal.pone.00
60101']
```