Data Exchange

Data Engineering

Data Exchange

- XML (eXtensible Markup Language): A versatile and structured data format, commonly used in web services, configuration, and data interchange between languages with different data structures.

- HTML (HyperText Markup Language): The standard markup language for documents to be displayed in a web browser. While it's not typically used for data exchange, it's crucial when web scraping is required.

- JSON (JavaScript Object Notation): Lightweight and human-readable. Predominantly used in web APIs and config files because of its easy integration with most programming languages.

Why It Matters for Data Engineering:

- Interoperability: These formats facilitate communication between different systems, languages, and architectures, ensuring a smooth data exchange.

- Versatility: Different systems prefer different formats. A data engineer might need XML for a SOAP web service, JSON for a RESTful service, and HTML when scraping data from websites.

- Flexibility: They can represent complex hierarchical data structures, allowing for the representation of almost any data model.

- Ubiquity: Given their widespread use, understanding these formats is essential for integrating with a vast number of platforms and tools.

XML

XML stands for eXtensible Markup Language.

- It is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Key Features:

- Semi-Structured Data: Enables the organization of data into nested, hierarchically structured documents.

- Self-descriptive: Tags describe the data and the data types, making it more interpretable.

- Platform-Independent: Works on any system, facilitating data interchange between incompatible systems.

```xml
<?xml version="1.0"
encoding="UTF-8"?>
<products>
  <product>
    <name>Laptop</name>
    <price>1000</price>
    <brand>Dell</brand>
    <stock>20</stock>
  </product>
  <product>
    <name>Smartphone</name>
    <price>600</price>
    <brand>Apple</brand>
    <stock>50</stock>
  </product>
  <product>
    <name>Desk Chair</name>
    <price>150</price>
    <brand>IKEA</brand>
    <stock>100</stock>
  </product>
</products>
```

Understanding the Structure of XML

Components of an XML Element:

- Element Name: Identifies the type of data, wrapped in angle brackets (<elementName>).
- Attributes: Additional descriptors within the opening tag (attribute="value").
- Child Elements: Elements nested within parent elements, forming a tree structure.
- Text Content: The actual data enclosed between the opening and closing tags.

```xml
<product id="1">
  <name lang="en">Laptop</name>
  <price currency="USD">1000</price>
  <brand origin="USA">Dell</brand>
  <stock status="InStock">20</stock>
</product>
```

XML Path Language

XPath is language designed for querying XML data.

Navigational Path Expressions

- /: Selects from the root node.
  Example: /bookstore selects the root element named product.
- //: Selects nodes from the current node that match the selection, regardless of their position.
  Example: //book selects all book elements in the document.
- .: Selects the current node.
  Example: . can be used to refer to the current node being processed within a loop.
- ..: Selects the parent of the current node.
  Example: .. selects the parent element of the current node.

```
<bookstore>
    <book>
        <title lang="en">Learning XML</title>
        <price currency="USD">29.95</price>
    </book>
    <book>
        <title lang="fr">Lire XML</title>
        <price currency="USD">39.95</price>
    </book>
    <book>
        <title lang="es">Aprendiendo XML</title>
        <price currency="USD">34.95</price>
    </book>
</bookstore>
```

Predicates

- [ ]: Used to find a specific node or a node that contains a specific value.
  Example: /bookstore/book[1] selects the first book element that is a child of bookstore.
- @: Selects attributes.
  Example: /bookstore/book/@lang selects the lang attribute of the book element.

Logical and Comparison Operators

- |: Computes the union of two sequences.
  Example: /bookstore/book | //price selects all book elements under bookstore and all price elements in the document.
- =: Checks for equality.
  Example: /bookstore/book[price>30] selects all book elements under bookstore with a price of 30.
- !=: Checks for inequality.
  Example: //price[@currency != 'USD'] selects all price elements that have a currency attribute not equal to 'USD'.
- >, <, >=, <=: Comparison operators.
  Example: /bookstore/book[price > 30] selects all book elements under bookstore with a price greater than 30.

Functions

- text(): Selects the text content of a node.
  Example: //book/title/text() selects the text content of all title elements inside book.

- contains(): Checks if a node contains a specific string.
  Example: //book[contains(title, 'XML')] selects all book elements that have 'XML' within their title.

Example

```
import pandas as pd
from lxml import etree

# Parse the XML string
xml_data = '”our xml"


root = etree.fromstring(xml_data)

# Use XPath to extract the desired elements
titles = root.xpath('//book/title/text()')
prices = root.xpath('//book/price/text()')

# Create a DataFrame
df = pd.DataFrame({
    'Title': titles,
    'Price': prices
})
```

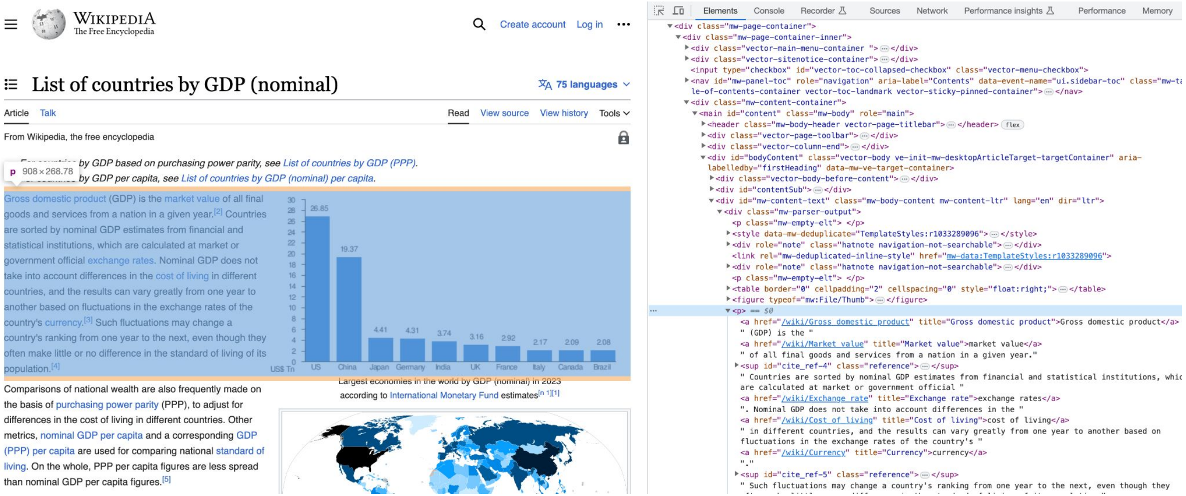
```
<data>
    <bookstore>
        <book>
            <title lang="en">Learning XML</title>
            <price currency="USD">29.95</price>
        </book>
        <book>
            <title lang="fr">Lire XML</title>
            <price currency="USD">39.95</price>
        </book>
    </bookstore>
</data>
```

	Title	Price
0	Learning XML	29.95
1	Lire XML	39.95

HTML (HyperText Markup Language)

Key Components:

- Elements: Represent structural units, e.g., <body>, <table> <div>, <p>, <a>
- Attributes: Provide additional information about elements, e.g., class="intro".



Extracting GDP data from Wikipedia

The table initially ranks each country or territory with their latest available estimates, and can be reranked by either of the sources

The links in the "Country/Territory" row of the following table link to the article on the GDP or the economy of the respective country or territory.

GDP (USD million) by country

	Country/Territory	UN region	IMF[1][13]		World Bank[14]		United Nations[15]	
			Estimate	Year	Estimate	Year	Estimate	Year
	World	**—**	**105,568,776**	**2023**	**100,562,011**	**2022**	**96,698,005**	**2021**
1	United States	Americas	26,854,599	2023	25,462,700	2022	23,315,081	2021
2	China	Asia	19,373,586	[n 1]2023	17,963,171	[n 3]2022	17,734,131	[n 1]2021
3	Japan	Asia	4,409,738	2023	4,231,141	2022	4,940,878	2021
4	Germany	Europe	4,308,854	2023	4,072,192	2022	4,259,935	2021
5	India	Asia	3,736,882	2023	3,385,090	2022	3,201,471	2021
6	United Kingdom	Europe	3,158,938	2023	3,070,668	2022	3,131,378	2021
7	France	Europe	2,923,489	2023	2,782,905	2022	2,957,880	2021
8	Italy	Europe	2,169,745	2023	2,010,432	2022	2,107,703	2021
9	Canada	Americas	2,089,672	2023	2,139,840	2022	1,988,336	2021
10	Brazil	Americas	2,081,235	2023	1,920,096	2022	1,608,981	2021
11	Russia	Europe	2,062,649	2023	2,240,422	2022	1,778,782	2021
12	South Korea	Asia	1,721,909	2023	1,665,246	2022	1,810,966	2021
13	Australia	Oceania	1,707,548	2023	1,675,419	2022	1,734,532	2021
14	Mexico	Americas	1,663,164	2023	1,414,187	2022	1,272,839	2021
15	Spain	Europe	1,492,432	2023	1,397,509	2022	1,427,381	2021

https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)

HTTP Request

Key Components:

- Method: Describes the desired action (e.g., GET, POST, PUT, DELETE).
- URL: The address to which the request is directed.
- Headers: Contains metadata for the HTTP request, such as user-agent, content-type, and more.
- Body: Contains data to be sent to the server (common in POST and PUT methods).

```
GET /wiki/Main_Page HTTP/1.1
Host: en.wikipedia.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.121 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

HTTP Response

Key Components:

- Status Code: Indicates the result of the request (e.g., 200 for OK, 404 for Not Found).
- Headers: Contains metadata about the response, like content-type, server details, and more.
- Body: Contains the data returned from the server, such as a webpage's HTML or API data.

```python
import requests


 wiki_url = 'https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)'
 wiki_resp = requests.get(wiki_url)



 print(wiki_resp)
```

```
<Response [200]>
```

```python
import requests


wiki_url = 'https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)'
wiki_resp = requests.get(wiki_url)



print(wiki_resp)
```

```
<Response [200]>
```

```python
from lxml import html

wiki_tree = html.fromstring(wiki_resp.content)


print(wiki_tree)
```

```
<Element html at 0x7e84e1fc5620>
```

```python
tables = wiki_tree.xpath('//table')
for t in tables:
  print(t)
```

```
<Element table at 0x7e84e1fc62f0>
<Element table at 0x7e84e1fa8db0>
<Element table at 0x7e84e1fa8f40>
<Element table at 0x7e84e1fa9490>
<Element table at 0x7e84e1faafc0>
<Element table at 0x7e84e1faa750>
<Element table at 0x7e84e1fa89f0>
```

```
tables = wiki_tree.xpath('//table')
for t in tables:
  print(t)
```

```
<Element table at 0x7e84e1fc62f0>
<Element table at 0x7e84e1fa8db0>
<Element table at 0x7e84e1fa8f40>
<Element table at 0x7e84e1fa9490>
<Element table at 0x7e84e1faafc0>
<Element table at 0x7e84e1faa750>
<Element table at 0x7e84e1fa89f0>
```

```
wiki_tree.xpath('//table[2]/caption/text()')
```

```
['GDP (USD million) by country\n']
```

GDP (USD million) by country

	Country/Territory	UN region	IMF[1][13]		World Bank[14]		United Nations[15]	
			Estimate	Year	Estimate	Year	Estimate	Year
	a 84.05 × 16	—	**105,568,776**	**2023**	**100,562,011**	**2022**	**96,698,005**	**2021**
1	United States	Americas	26,854,599	2023	25,462,700	2022	23,315,081	2021
2	China	Asia	19,373,586	[n 1]2023	17,963,171	[n 3]2022	17,734,131	[n 1]2021
3	Japan	Asia	4,409,738	2023	4,231,141	2022	4,940,878	2021
4	Germany	Europe	4,308,854	2023	4,072,192	2022	4,259,935	2021
5	India	Asia	3,736,882	2023	3,385,090	2022	3,201,471	2021
6	United Kingdom	Europe	3,158,938	2023	3,070,668	2022	3,131,378	2021
7	France	Europe	2,923,489	2023	2,782,905	2022	2,957,880	2021
8	Italy	Europe	2,169,745	2023	2,010,432	2022	2,107,703	2021
9	Canada	Americas	2,089,672	2023	2,139,840	2022	1,988,336	2021
10	Brazil	Americas	2,081,235	2023	1,920,096	2022	1,608,981	2021

```
▼<tr class="static-row-header" style="font-weight:bold;">
   ::before
  ▶<td style="text-align:left">⋯</td>
   <td style="text-align:center">—</td>
   <td>105,568,776</td>
   <td>2023</td>
   <td>100,562,011</td>
   <td>2022</td>
   <td>96,698,005</td>
   <td>2021 </td>
 </tr>
▼<tr>
   ::before
  ▼<td style="text-align:left"> == $0
    ▶<span class="flagicon" style="display:inline-block;width:25px;text-align:left">⋯</span>
     " "
     <a href="/wiki/Economy_of_the_United_States" title="Economy of the United States">United
     States</a>
   </td>
  ▼<td style="text-align:center">
     <a href="/wiki/Americas" title="Americas">Americas</a>
   </td>
   <td>26,854,599</td>
   <td>2023</td>
```

```
country_name = wiki_tree.xpath('//table[2]/tbody/tr/td[1]/a/text()')
region_name = wiki_tree.xpath('//table[2]/tbody/tr/td[2]/a/text()')
```

GDP (USD million) by country

	Country/Territory	UN region	IMF[1][13]		World Bank[14]		United Nations[15]	
			Estimate	Year	Estimate	Year	Estimate	Year
	a 84.05 × 16	—	105,568,776	2023	100,562,011	2022	96,698,005	2021
1	United States	Americas	26,854,599	2023	25,462,700	2022	23,315,081	2021
2	China	Asia	19,373,586	[n 1]2023	17,963,171	[n 3]2022	17,734,131	[n 1]2021
3	Japan	Asia	4,409,738	2023	4,231,141	2022	4,940,878	2021
4	Germany	Europe	4,308,854	2023	4,072,192	2022	4,259,935	2021
5	India	Asia	3,736,882	2023	3,385,090	2022	3,201,471	2021
6	United Kingdom	Europe	3,158,938	2023	3,070,668	2022	3,131,378	2021
7	France	Europe	2,923,489	2023	2,782,905	2022	2,957,880	2021
8	Italy	Europe	2,169,745	2023	2,010,432	2022	2,107,703	2021
9	Canada	Americas	2,089,672	2023	2,139,840	2022	1,988,336	2021
10	Brazil	Americas	2,081,235	2023	1,920,096	2022	1,608,981	2021

```
▼<tr class="static-row-header" style="font-weight:bold;">
   ::before
  ▶<td style="text-align:left">…</td>
   <td style="text-align:center">—</td>
   <td>105,568,776</td>
   <td>2023</td>
   <td>100,562,011</td>
   <td>2022</td>
   <td>96,698,005</td>
   <td>2021 </td>
 </tr>
▼<tr>
   ::before
  ▼<td style="text-align:left"> == $0
    ▶<span class="flagicon" style="display:inline-block;width:25px;text-align:left">…</span>
     " "
     <a href="/wiki/Economy_of_the_United_States" title="Economy of the United States">United
     States</a>
   </td>
  ▼<td style="text-align:center">
     <a href="/wiki/Americas" title="Americas">Americas</a>
   </td>
   <td>26,854,599</td>
   <td>2023</td>
```

```
country_name = wiki_tree.xpath('//table[2]/tbody/tr/td[1]/a/text()')
region_name = wiki_tree.xpath('//table[2]/tbody/tr/td[2]/a/text()')
```

```
world_bank_est = wiki_tree.xpath('//table[2]/tbody/tr/td[5]/text()')
```

```
country_name = wiki_tree.xpath('//table[2]/tbody/tr/td[1]/a/text()')
region_name = wiki_tree.xpath('//table[2]/tbody/tr/td[2]/a/text()')
```

```
world_bank_est = wiki_tree.xpath('//table[2]/tbody/tr/td[5]/text()')
```

```
worldbank_df = pd.DataFrame({'Country': country_name, 'Region': region_name,
'Estimate':  world_bank_est[1:]});
```

	Country	Region	Estimate
0	United States	Americas	25,462,700
1	China	Asia	17,963,171
2	Japan	Asia	4,231,141
3	Germany	Europe	4,072,192
4	India	Asia	3,385,090
...	...	...	...
208	Anguilla	Americas	303
209	Kiribati	Oceania	223
210	Nauru	Oceania	151
211	Montserrat	Americas	72
212	Tuvalu	Oceania	60

213 rows × 3 columns

```
worldbank_df[worldbank_df['Estimate'] == '-']
```

	Country	Region	Estimate
20	Taiwan	Asia	—
71	Venezuela	Americas	—
75	Turkmenistan	Asia	—
127	Yemen	Asia	—
160	South Sudan	Africa	—
171	Aruba	Americas	—
179	Bhutan	Asia	—
180	Eritrea	Africa	—
189	San Marino	Europe	—
203	Tonga	Oceania	—
207	Palau	Oceania	—

```
nworldbank_df = worldbank_df[worldbank_df['Estimate'] == '-']

nworldbank_df['Estimate'] = nworldbank_df['Estimate'].apply(lambda x:
float(str(x).replace(',','')))
```

	Country	Region	Estimate
0	United States	Americas	25462700.0
1	China	Asia	17963171.0
2	Japan	Asia	4231141.0
3	Germany	Europe	4072192.0
4	India	Asia	3385090.0
...	...	...	...
208	Anguilla	Americas	303.0
209	Kiribati	Oceania	223.0
210	Nauru	Oceania	151.0
211	Montserrat	Americas	72.0
212	Tuvalu	Oceania	60.0

202 rows × 3 columns

```
region_gpd_stats = nworldbank_df.groupby('Region').agg({'Estimate': ['mean',
'sum' ,'max','min']}).reset_index()

region_gpd_stats.columns = ['Region', 'Mean', 'Sum', 'Max', 'Min']
```

	Region	Mean	Sum	Max	Min
0	Africa	55473.396226	2940090.0	477386.0	547.0
1	Americas	759319.136364	33410042.0	25462700.0	72.0
2	Asia	797008.978261	36662413.0	17963171.0	2022.0
3	Europe	540130.409091	23765738.0	4072192.0	3352.0
4	Oceania	131949.066667	1979236.0	1675419.0	60.0

REST: REpresentational State Transfer

- Statelessness: Every request from a client to a server must contain all information needed to understand and process the request.
- Client-Server: A separation of concerns. The client handles the user interface, while the server manages the backend and data.
- Cacheable: Responses can be cached, implying that some responses are reusable for identical requests in the future.

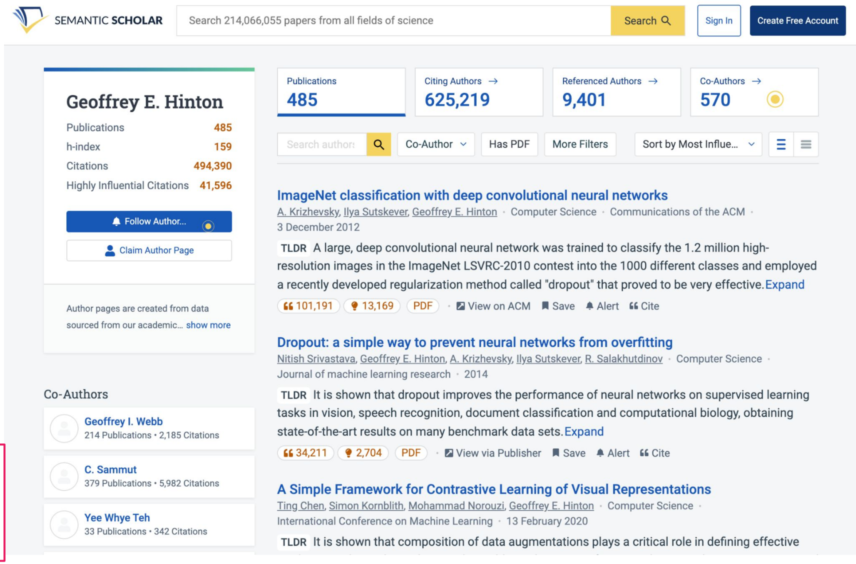
HTTP Methods (CRUD operations):

GET: Retrieve data.
POST: Add new data.
PUT/PATCH: Update existing data.
DELETE: Remove data.

Example: Finding Influential Papers

```
import requests
r =requests.get('https://api.semanticscholar.org/graph/v1/author/search',
    params={'query': "Geoffrey E. Hinton"})
r.content
```

{'total': 3,
 'offset': 0,
 'data': [{'authorId': '116210976',
'name': 'E. Geoffrey'},
  {'authorId': '50286838', 'name':
'James E. . Hinton'},
  {'authorId': '1695689', 'name':
'Geoffrey E. Hinton'}]
}




```
authorId = matched_authors['data'][2]['authorId'])
print(authorId)
```

'1695689'

```python
r = requests.get(
  'https://api.semanticscholar.org/graph/v1/author/' + authorId,
  params={'fields':'name,paperCount,citationCount,papers.title,papers.influentialCitationCount'})
paper_json = r.json()
```


```
{'authorId': '1695689',
 'name': 'Geoffrey E. Hinton',
 'paperCount': 484,
 'citationCount': 493491,
 'papers': [{'paperId': '40b736b8a628504a9463130705f8012d2e5e5560',
   'title': 'Robust and data-efficient generalization of self-supervised machine learning for d
   'influentialCitationCount': 0},
  {'paperId': '944ab9578916572cd3896b2452dc89ba8ff2e8a9',
   'title': 'CogSci 2020 Developing a Mind: Learning in Humans, Animals, and Machines',
   'influentialCitationCount': 0},
  {'paperId': 'd26f928defed8d10a374d587cfdfb7b256d30c82',
   'title': 'UvA-DARE (Digital Academic Repository) A model of prenatal acquisition of vowels',
   'influentialCitationCount': 0},
  {'paperId': '06761cb27e14aa55a6c3d98b949898aa26416698',
   'title': 'A Unified Sequence Interface for Vision Tasks',
   'influentialCitationCount': 9},
  {'paperId': '37ba9c33025fb31f25436010e12c65a0bafc0e1f',
   'title': 'Meta-Learning Fast Weight Language Models',
   'influentialCitationCount': 0},
  {'paperId': '39e2f96723e41b38d7bf1bef6825506d7b5394c8',
   'title': 'Neural Networks',
   'influentialCitationCount': 0},
  {'paperId': '614dde18483338069d482d7452900c28052aba83',
   'title': 'A Generalist Framework for Panoptic Segmentation of Images and Videos',
   'influentialCitationCount': 6},
  {'paperId': '75e3475cf49caf1dbbcad526b0132b455dc88dd5',
   'title': 'The Forward-Forward Algorithm: Some Preliminary Investigations',
   'influentialCitationCount': 11},
```


- `authorId` - S2 unique ID for this author
- `externalIds` - ORCID/DBLP IDs for this author, if known
- `url` - URL on the Semantic Scholar website
- `name` - Author's name
- `aliases` - List of names the author has used on publications over time, not intended to be displayed to users. WARNING: this list may be out of date or contain deadnames of authors who have changed their name. (see https://en.wikipedia.org/wiki/Deadnaming)
- `affiliations` - Author's affiliations - sourced from claimed authors who have set affiliation on their S2 author page.
- `homepage` - Author's homepage
- `paperCount` - Author's total publications count
- `citationCount` - Author's total citations count
- `hIndex` - See the S2 FAQ on h-index
- `citations`
  - `paperId` - Always included. A unique (string) identifier for this paper
  - `corpusId` - A second unique (numeric) identifier for this paper
  - `url` - URL on the Semantic Scholar website
  - `title` - Included if no fields are specified
  - `venue` - Normalized venue name
  - `publicationVenue` - Publication venue meta-data for the paper
  - `year` - Year of publication
  - `authors` - Up to 500 will be returned. Will include: `authorId` & `name`
  - To get more detailed information about an author's papers, use the `/author/{author_id}/papers` endpoint
  - Total number of citations will be truncated at 10,000 for the entire batch.
  - To fetch more citations per paper, reduce the number of papers in the batch with `limit=` or use the `/paper/{paper_id}/citations` endpoint.
- `references`
  - `paperId` - Always included. A unique (string) identifier for this paper
  - `corpusId` - A second unique (numeric) identifier for this paper
  - `url` - URL on the Semantic Scholar website
  - `title` - Included if no fields are specified
  - `venue` - Normalized venue name
  - `publicationVenue` - Publication venue meta-data for the paper
  - `year` - Year of publication
  - `authors` - Up to 500 will be returned. Will include: `authorId` & `name`
  - To get more detailed information about an author's papers, use the `/author/{author_id}/papers` endpoint
  - Same fields supported as for papers above
  - Total number of references will be truncated at 10,000 for the entire batch.
  - To fetch more references per paper, reduce the number of papers in the batch with `limit=` or use the `/paper/{paper_id}/references` endpoint.

pandas.json_normalize

```
pandas.json_normalize(data, record_path=None, meta=None,
meta_prefix=None, record_prefix=None, errors='raise', sep='.',
max_level=None)
```

Normalize semi-structured JSON data into a flat table.

Parameters:

data: dict or list of dicts

record_path: str or list of str, default None

Path in each object to list of records. If not passed, data will be assumed to be an array of records.

meta: list of paths (str or list of str), default None

Fields to use as metadata for each record in resulting table.

```
pd.json_normalize(paper_json)
```

	authorId	name	paperCount	citationCount	papers
0	1695689	Geoffrey E. Hinton	484	493491	[{'paperId': '40b736b8a628504a9463130705f8012d...

```
pd.json_normalize(paper_json)
```

	authorId	name	paperCount	citationCount	papers
0	1695689	Geoffrey E. Hinton	484	493491	[{'paperId': '40b736b8a628504a9463130705f8012d...

```
pd.json_normalize(paper_json, record_path="papers")
```

	paperId	title	influentialCitationCount
0	40b736b8a628504a9463130705f8012d2e5e5560	Robust and data-efficient generalization of se...	0
1	944ab9578916572cd3896b2452dc89ba8ff2e8a9	CogSci 2020 Developing a Mind: Learning in Hum...	0
2	d26f928defed8d10a374d587cfdfb7b256d30c82	UvA-DARE (Digital Academic Repository) A model...	0
3	06761cb27e14aa55a6c3d98b949898aa26416698	A Unified Sequence Interface for Vision Tasks	9
4	37ba9c33025fb31f25436010e12c65a0bafc0e1f	Meta-Learning Fast Weight Language Models	0
...	...	...	...
480	2b114f4d05494fceb22473fcd29d940e9aa52bf4	Rectified Linear Units Improve Restricted Boltz...	60
481	3e5d838aef4aea3ff22a744f292cfb4094909e35	COOPERATIVE : COMPUTATION	0
482	5f75f39caf0ad58afd117ec14266658a7e504781	Using mixtures of deformable models to capture...	0
483	88d214e9fb00e58a7b504c88be0af4b4d07c624f	Maximizing Mutual Information	0
484	f1fa5a8addd84be5a0e0382772c807d1ebcf0476	7. Conclusion and Future Work Novel Objective ...	0

485 rows × 3 columns

```
pd.json_normalize(paper_json, record_path="papers", meta=['authorId'])
```

	paperId	title	influentialCitationCount	authorId
0	40b736b8a628504a9463130705f8012d2e5e5560	Robust and data-efficient generalization of se...	0	1695689
1	944ab9578916572cd3896b2452dc89ba8ff2e8a9	CogSci 2020 Developing a Mind: Learning in Hum...	0	1695689
2	d26f928defed8d10a374d587cfdfb7b256d30c82	UvA-DARE (Digital Academic Repository) A model...	0	1695689
3	06761cb27e14aa55a6c3d98b949898aa26416698	A Unified Sequence Interface for Vision Tasks	9	1695689
4	37ba9c33025fb31f25436010e12c65a0bafc0e1f	Meta-Learning Fast Weight Language Models	0	1695689
...	...	...	...	...
480	2b114f4d05494fceb22473fcd29d940e9aa52bf4	Rectified Linear Units Improve Restricted Boltz...	60	1695689
481	3e5d838aef4aea3ff22a744f292cfb4094909e35	COOPERATIVE : COMPUTATION	0	1695689
482	5f75f39caf0ad58afd117ec14266658a7e504781	Using mixtures of deformable models to capture...	0	1695689
483	88d214e9fb00e58a7b504c88be0af4b4d07c624f	Maximizing Mutual Information	0	1695689
484	f1fa5a8addd84be5a0e0382772c807d1ebcf0476	7. Conclusion and Future Work Novel Objective ...	0	1695689

485 rows × 4 columns

```
hinton_papers = pd.json_normalize(paper_json, record_path="papers", meta=['authorId'])
hinton_papers.sort_values(by=['influentialCitationCount'],ascending=False).head()
```

	paperId	title	influentialCitationCount	authorId
106	abd1c342495432171beb7ca8fd9551ef13cbd0ff	ImageNet classification with deep convolutiona...	13169	1695689
77	34f25a8704614163c4095b3ee2fc969b60de4698	Dropout: a simple way to prevent neural networ...	2704	1695689
24	34733eaf66007516347a40ad5d9bbe1cc9dacb6b	A Simple Framework for Contrastive Learning of...	2648	1695689
74	a4cec122a08216fe8a3bc19b22e78fbaea096256	Deep Learning	2608	1695689
71	0c908739fbff75f03469d13d4a1a07de3414ee19	Distilling the Knowledge in a Neural Network	1847	1695689