

DATA FOR HEROES II

OLF 2025

Catherine Devlin

2025-12-06

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

SUPERVILLAINS

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>



Nell Strongarm
The Astro-Thug



Henry Fnord
Mass-Producer
Of Mayhem



The Trilobite



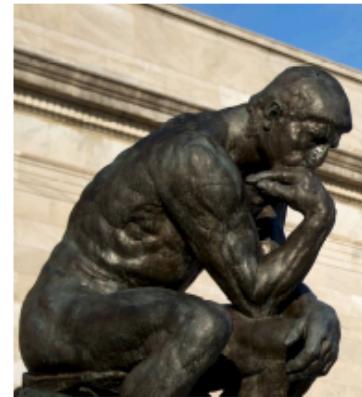
The Wrong Brothers
Fiends of Flight

SUPERHEROES

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>



Cardinal Virtue
The Redbird of
Righteousness



The Thinker



Captain Columbus
Protector of Ohio



Ohio Gosaimasu
Sandusky Samurai

INFORMATION?

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>



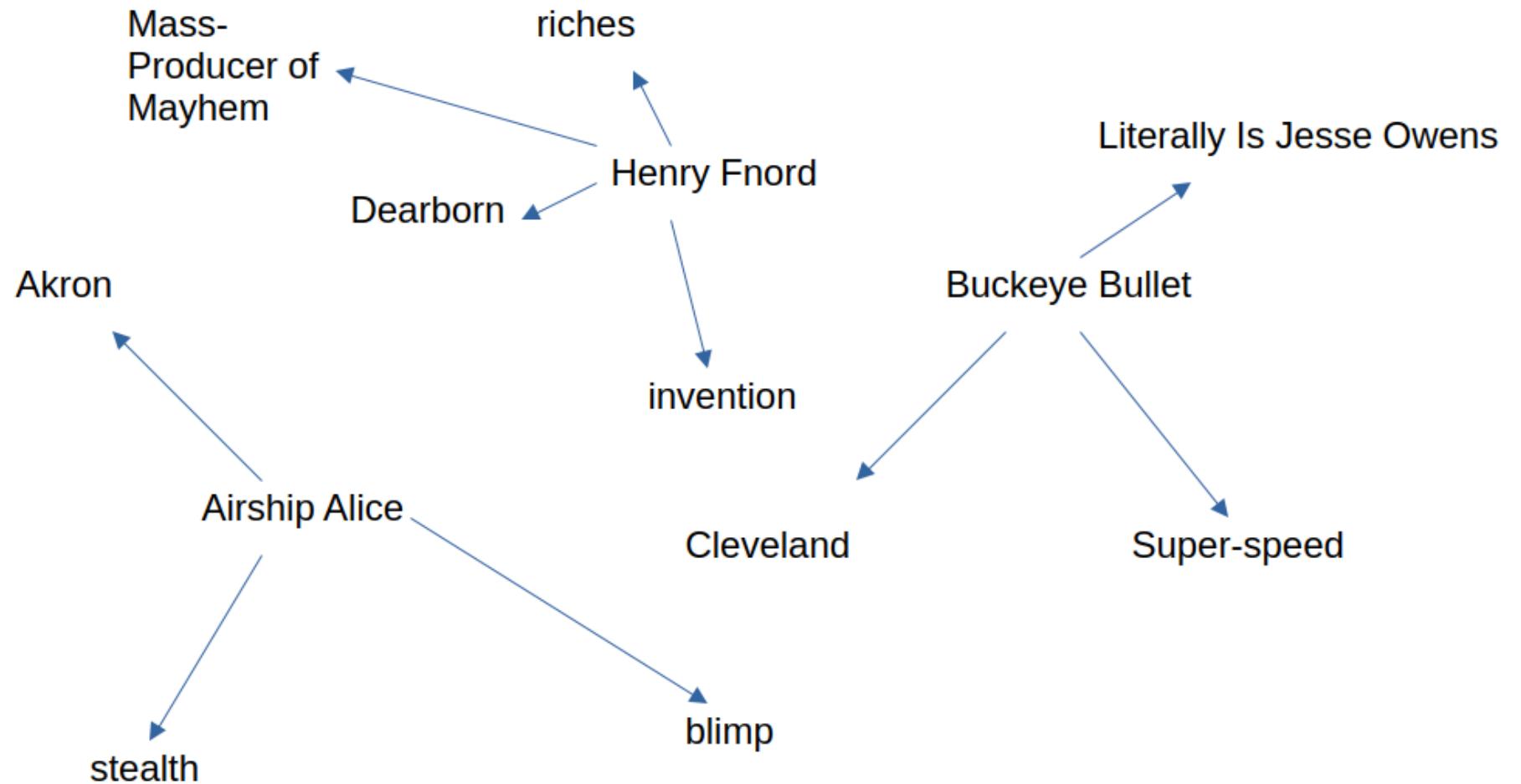
RELATIONSHIPS == INFORMATION



TABLES (RELATIONS)

Name	Tagline	Location	Power 1	Power 2
Buckeye Bullet	Literally Is Jesse Owens	Cleveland	Super-speed	
Henry Fnord	Mass- Producer of Mayhem	Dearborn	invention	riches
Airship Alice		Akron	blimp	stealth

GRAPH: NODES AND EDGES



FLEXIBILITY

Name	Power 1	Power 2	Favorite food	Zodiac	Height (cm)
Buckeye Bullet	speed				178
Airship Alice	blimp	stealth	Sub		
Henry Fnord	invention	riches		Leo	

CHAINED QUERY

“Somebody who knows somebody who knows somebody who knows Henry Fnord”

NEO4J

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

neo4j.com

Aura Login Partners Company Support

neo4j Products Use Cases Developers AI Systems Learn Pricing Contact Us Get Started Free

THE WORLD'S LEADING GRAPH INTELLIGENCE PLATFORM

Build Intelligent Apps and AI For Supply Chain Management

AI-ready data by design

[Start Building](#) [Learn More](#)

300k Developers Building | 80+ Fortune 100 Customers | 170+ Partner Ecosystem

alma^r Santander COMCAST BNP PARIBAS PERSONAL FINANCE UBS AIRBUS BT Group BMW Boston Scientific novo nordisk dun&bradstreet

CHELNICK, THE POSTGRESQL ELEPHANT

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>



OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

The screenshot shows the Apache AGE website at https://age.apache.org/. The header includes a navigation bar with links to Get Started, Apache AGE, Community, Contribution, Doc, Downloads (which is highlighted), and Apache AGE GitHub. The main title is "Apache AGE™ Graph Database for PostgreSQL". Below the title is a subtext: "Apache AGE™ is a PostgreSQL Graph database compatible with PostgreSQL's distributed assets and leverages graph data structures to analyze and use relationships and patterns in data." There are two buttons: "Get Started with Apache AGE" and "Download Apache AGE". A callout box on the left says "Apache AGE™, Graph database" and describes it as a PostgreSQL that provides graph database functionality. Another callout box on the right, titled "Apache AGE Architecture", details the two-stage query processing: 1. Parsers queries by a function call that uses parser following the openCypher specification; 2. Transforms a Cypher query into a Query tree that will be attached as a subquery node.

Apache AGE™
Graph Database for PostgreSQL

Apache AGE™ is a PostgreSQL Graph database compatible with PostgreSQL's distributed assets and leverages graph data structures to analyze and use relationships and patterns in data.

Get Started with Apache AGE Download Apache AGE

Apache AGE™, Graph database

Apache AGE™ is a PostgreSQL that provides graph database functionality.

The goal of Apache AGE™ is to provide graph data processing and

Apache AGE Architecture

/Cache Layer

Query Parsing

Query Transform

1. Parses queries by a function call that uses parser following the openCypher specification.
2. Transforms a Cypher query into a Query tree that will be attached as a subquery node.

GETTING INFORMATION

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

ASSUMED

supers.ods — LibreOffice Calc

File Edit View Insert Format Styles Sheet Data Tools Window Help

Liberation Sans 10 pt **I** *U* **A** A **B** **C** **D** **E** **F** **G** **H** **I** **J** **K** **L** **M** **N** **O** **P** **Q** **R** **S** **T** **U** **V** **W** **X** **Z** **Y** **Z** **Y** **X** **W** **V** **U** **T** **S** **R** **Q** **P** **O** **N** **M** **L** **K** **J** **I** **H** **G** **F** **E** **D** **C** **B** **A** **Z** **Y** **X** **W** **V** **U** **T** **S** **R** **Q** **P** **O** **N** **M**

E22 fx Σ = swimming

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
10	Blue Jacket	hero		Chillicothe	horsemanship	crack shot	oratory																			
11	Captain Columbus	hero	Protector of Ohio	Columbus	Super-strength	flight																				
12	Buckeye Bullet	hero		Cleveland	Super-speed																					
13	Annie Oakley	hero	Little Miss Sure Shot	Greenville	marksmanship	acrobatics	healing																			
14	The Stealer	villain		Pittsburgh	stealth																					
15	Ohio Gosaimasu	hero	Sandusky Samurai	Sandusky	swordplay	honor																				
16	The Hawk King	hero		Hocking Hills	flight	Super-strength																				
17	Airship Alice	villain		Akron	blimp	stealth																				
18	Motown Menace	villain		Detroit	funk	beat																				
19	Captain Cuyahoga	villain	Fire and Water Do Mix	Cleveland	water control	generate fire																				
20	Indian Jane	hero	Nazi-punching archaeologist	Bloomington	scholarship	Derring-do																				
21	Defiance Dogman	villain		Defiance	fangs	speed																				
22	Loveland Frogman			Loveland	swimming	tongue																				
23	Marvelous Mudhen	hero		Toledo	swimming																					
24																										

Sheet1 Sheet2 | Find Find All Formatted Display Match Case Average: ; Sum: 0 100%

ALWAYS A SEQUEL

2023: Info was gathered, needed organization

2025: Gather the info

AUTOMATE!

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

REQUESTS

The screenshot shows a web browser window with the Requests documentation at requests.readthedocs.io/en/latest/. The page features a large logo of a caduceus (a staff with two snakes entwined and wings at the top) above the word "Requests". Below the logo is a button to "Star" the project with a count of 53,528. A sidebar on the left contains "Useful Links" such as Quickstart, Advanced Usage, API Reference, Release History, Contributors Guide, Recommended Packages and Extensions, Requests @ GitHub, Requests @ PyPI, and Issue Tracker. A "Quick Search" bar is also present. The main content area has a title "Requests: HTTP for Humans™" and a sub-section "Behold, the power of Requests:" showing Python code examples for making HTTP requests. A "Fork me on GitHub" badge is in the top right corner. The bottom right corner of the page has a "latest" dropdown menu.

Requests: HTTP for Humans™

Release v2.32.5. ([Installation](#))

downloads/month 953M license Apache-2.0 wheel yes python 3.9 | 3.10 | 3.11 | 3.12 | 3.13 | 3.14

Requests is an elegant and simple HTTP library for Python, built for human beings.

Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type": "User"...'
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
```

See [similar code, sans Requests](#).

Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to [urllib3](#).

Beloved Features

Requests is ready for today's web.

- Keep-Alive & Connection Pooling
- International Domains and URLs
- Sessions with Cookie Persistence
- Browser-style SSL Verification
- Automatic Content Decoding
- Basic/Digest Authentication
- Elegant Key/Value Cookies
- Automatic Decompression
- Unicode Response Bodies
- HTTP(S) Proxy Support

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

GETTING A FILE: SUPERS.CSV

```
>>> import requests  
>>> resp =  
requests.get("http://localhost:8000/supers.csv")  
>>> resp.text  
'name,align,tagline,based_in,power_1,power_2,powe  
Trilobite,,Caped  
Cambrian,Sylvania,armor,antennae,\nPutin  
Bae,villain,Rasslin\x80\x99  
Russophile,Mansfield,politics,wrestling,selective  
blindness\nCardinal Virtue,hero,Redbird of  
Righteousness Cincinnati flight anarv
```

HTML TABLE

Wikipedia: [List of Superpowered individuals in Ohio](#)

```
<title>List of Superpowered individuals in Ohio  
- Wikipedia</title>  
<style>:root{--sf-img-1:  
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj  
-sf-img-2:  
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj  
-sf-img-3:  
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj  
-sf-img-4:  
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj  
-sf-img-5:<
```

PANDAS

Dataframe processing library

pandas - Python Data An... +

pandas.pydata.org

 pandas

About us ▾ Getting started Documentation Community ▾ Contribute

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

Getting started

- Install pandas
- Getting started
- Try pandas online

Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

Community

- About pandas
- Ask a question
- Ecosystem

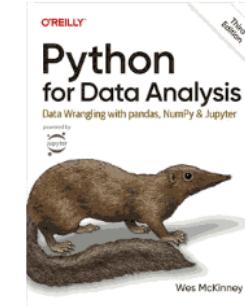
Latest version: 2.3.3

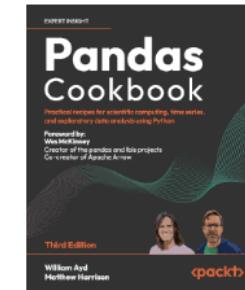
- What's new in 2.3.3
- Release date: Sep 29, 2025
- Documentation (web)
- Download source code

Follow us



Recommended books





With the support of:



NUMFOCUS
OPEN CODE = BETTER SCIENCE



NVIDIA.



TIDELIFT



bodo.ai

The full list of companies supporting *pandas* is available in the sponsors page.

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

DATA IMPORT FUNCTIONS

```
>>> import pandas as pd  
>>> pd.read_  
pd.read_clipboard(    pd.read_gbq(  
pd.read_parquet(      pd.read_sql_query(  
pd.read_csv(          pd.read_hdf(  
pd.read_pickle(       pd.read_sql_table(  
pd.read_excel(        pd.read_html(  
pd.read_sas(          pd.read_stata(  
pd.read_feather(     pd.read_json(  
pd.read_spss(         pd.read_table(  
pd.read_fwf(          pd.read_orc(  
)
```

read_html() READS ALL TABLES

```
>>> import pandas as pd  
  
>>> URL = 'http://localhost:8000/ohio-  
supers.html'  
>>> frames = pd.read_html(URL)  
>>> len(frames)  
3
```

PPPRINT

```
>>> import pprint  
>>> pprint.pprint(frames)
```

	Unnamed: 0	Name Alignment
Tagline	Based in	Power 1
Power 2		Power 3
0	0	The Trilobite
Caped Cambrian		Silvania
antennae		NaN
1	1	Putin Bae villain
Rasslin' Russophile		Mansfield

DUMP

```
>>> df.to_
df.to_clipboard(  df.to_excel(      df.to_hdf(
df.to_latex(       df.to_orc(      
df.to_pickle(       df.to_stata(    
df.to_xarray()
df.to_csv(          df.to_feather(    df.to_html(
df.to_markdown(    df.to_parquet(   df.to_xml(
df.to_records(     df.to_string(    df.to_json(
df.to_dict(         df.to_gbq(      df.to_sql(
df.to_numpy(        df.to_period(   import pandas as pd
df.to_timestamp()
```

MORE SPECIFIC JSONL

```
import sys

import jsonlines
import pandas as pd

URL = "http://localhost:8000/ohio-supers.html"

def json_rows(row):
    if row["Based in"]:
        yield {
            "entity_1": {
```

OUTPUT

```
$ uv run get-html.py
{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Place",
"name": "Silvania"}, "relationship":
"based_in"}
{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Power",
"name": "armor"}, "relationship": "has"}
{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Power",
"name": "antennae"}, "relationship": "has"}
```

FROM JSONL TO NEO4J

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

GETTING NEO4J

neo4j.com -> cloud product

<https://debian.neo4j.com/>

```
wget -O -
https://debian.neo4j.com/neotechnology.gpg.key |
sudo apt-key add -
echo 'deb https://debian.neo4j.com stable
latest' | sudo tee
/etc/apt/sources.list.d/neo4j.list
sudo apt-get update
sudo apt-get install neo4j
```

Docker container available

STARTING

```
sudo -i -u neo4j neo4j-admin dbms set-initial-  
password 4justice  
sudo -i -u neo4j start  
open http://localhost:7474
```

SHOW FULL GRAPH

```
MATCH (n) RETURN n;
```

INSERTING DATA

- CYPHER queries
- .csv files
- [neo4j](#) Python library

WRITE & EXECUTE CYpher QUERIES

jsonl2neo4j.py

```
import sys
import jsonlines
from neo4j import GraphDatabase, RoutingControl

URI = "neo4j://localhost:7687"
AUTH = ("neo4j", "4justice")

def save(driver, row):
    relationship = row["relationship"].upper()
    type2 = row["entity_2"]["type"]
    qry = (
```

(THIS)-[:TO]->(THAT)

```
MERGE (entity_1:Person {name: "Captain  
Columbus"})  
MERGE (entity_2:Power {name: "Super-strength"})  
MERGE (entity_1)-[:HAS]->(entity_2)
```

FULL PIPELINE

```
uv run get-html.py > supers.jsonl  
head supers.jsonl  
uv run jsonl2neo4j.py supers.jsonl
```

VIEW

```
MATCH (n) RETURN n;
```

HARVEST DATA

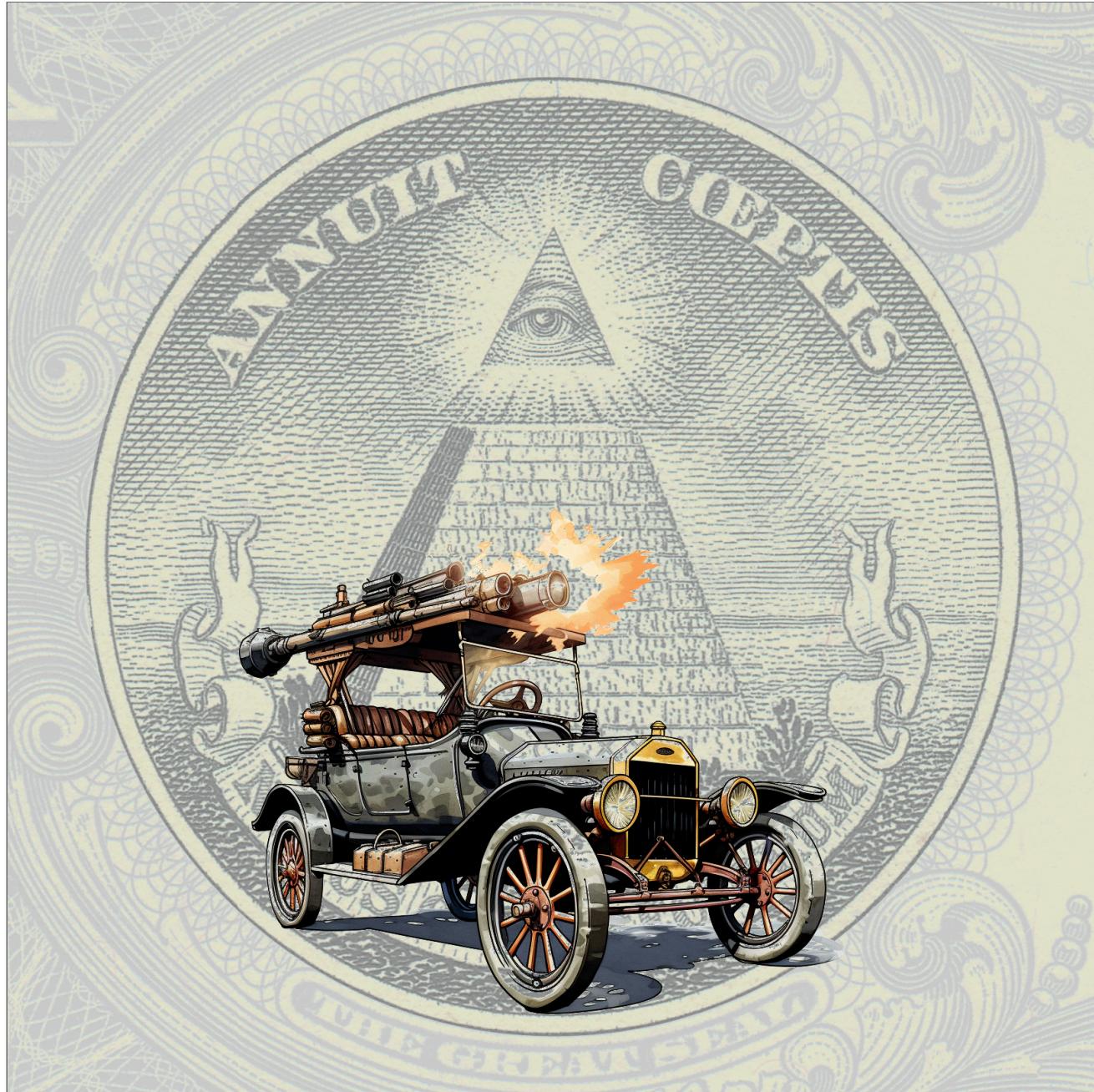
```
>>> pd.read_
pd.read_clipboard(    pd.read_gbq(
pd.read_parquet(      pd.read_sql_query(
pd.read_csv(          pd.read_hdf(
pd.read_pickle(        pd.read_sql_table(
pd.read_excel(         pd.read_html(
pd.read_sas(          pd.read_stata(
pd.read_feather(       pd.read_json(
pd.read_spss(          pd.read_table(
pd.read_fwf(          pd.read_orc(
pd.read_sql(          pd.read_xml(
```

UNCOMPILED DATA

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

INNOVATORS OF INIQUITY

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>



OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

Henry Fnord's Model T2: Judgement Day

WEB SCRAPING

Innovators of Iniquity

- Requests + BeautifulSoup
- Scrapy

GET SCRAPY

```
$ uv init --bare -p 3.13
Initialized project `ccbus`
$ uv add scrapy
Using CPython 3.14.0rc1 interpreter at:
/home/catherine/.local/bin/python3.14
Creating virtual environment at: .venv
Resolved 39 packages in 760ms
Prepared 2 packages in 407ms
Installed 37 packages in 222ms
+ attrs==25.4.0
+ automat==25.4.16
```

NEW PROJECT

```
$ uv run scrapy startproject ccbus_scraper
New Scrapy project 'ccbus_scraper', using
template directory
'/home/catherine/ccbus/.venv/lib/python3.14/site-
packages/scrapy/templates/project', created in:
/home/catherine/ccbus/ccbus_scraper
```

You can start your first spider with:
cd ccbus_scraper
scrapy genspider example example.com

WHAT DO WE HAVE?

```
$ tree ccbus_scraper/
ccbus_scraper/
├── ccbus_scraper
│   ├── __init__.py
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       └── __init__.py
└── scrapy.cfg
```

EMPTY SPIDER

```
import scrapy

class IOISpider(scrapy.Spider):
    name = "ioi"
    start_urls = [
        "http://localhost:8000/pod.html",
    ]

    def parse(self, response):
        if response.status == 200:
            self.log(f"Scraped {response.url}")
```

RUN

```
/ccbus_scraper$ uv run scrapy crawl ioi
2025-11-27 15:17:16 [scrapy.utils.log] INFO:
Scrapy 2.13.4 started (bot: ccbus_scraper)
2025-11-27 15:17:16 [scrapy.utils.log] INFO:
Versions:
{'lxml': '6.0.2',
'libxml2': '2.14.6',
'cssselect': '1.3.0',
'parsel': '1.10.0',
'w3lib': '2.3.1',
'Twisted': '25.5.0'}
```

FIND ELEMENTS OF INTEREST

Developer Tools (Ctrl+Shift+I)

Right-click; Inspect

Right-click; Copy; CSS Path

CSS PATH

```
.media-card__title > a {  
    border-bottom-width: 0px;  
    --tw-text-opacity: 1;  
    color: rgb(33 83 128 / var(--tw-text-  
opacity));  
    text-decoration-line: none;  
}
```

COPY SELECTOR

```
#app > div.container.px-4.py-6.\32 xl\:px-0 >  
div.flex-grid.md\:flex-wrap.md\:-mx-6.py-6 >  
div:nth-child(1) > div > div.media-card__header  
> h2 > a
```

Less specific will do!

SCRAPY SHELL

```
~/ccbus/ccbus_scraper$ uv run scrapy shell  
http://localhost:8000/pod.html  
2025-11-27 15:47:19 [scrapy.utils.log] INFO:  
Scrapy 2.13.4 started (bot: ccbus_scraper)  
...  
[s] help()          Shell help (print this  
help)  
[s] view(response) View response in a  
browser  
>>>
```

EXPERIMENT TO FIND

```
>>> response.css("div.media-card__header")
[<Selector query="descendant-or-
self::div[@class and contains(concat(' ',
normalize-space(@class), ' '), ' media-
card__header ')]" data='<div class="media-
card__header">\n      ...>', <Selector
query="descendant-or-self::div[@class and
contains(concat(' ', normalize-space(@class), ' ',
' media-card__header ')]" data='<div
class="media-card__header">\n      ...>',
<Selector query="descendant-or-self::div[@class
```

:TEXT

```
>>> response.css("div.media-card__header  
a::text").getall()  
['\n                                Holiday Villainy  
Season 2025\n                                ', '\n  
Interview: The Wrong Brothers\n                                Interview: Putin  
, '\n                                Bae\n                                ', '\n  
Interview: The Trilobite\n                                ' ]
```

IOI.PY

```
import scrapy

class IOISpider(scrapy.Spider):
    name = "ioi"
    start_urls = [
        "http://localhost:8000/pod.html",
    ]

    def parse(self, response):
        for line in response.css("div.media-
card__header a::text").getall():
```

FEED EXPORTS

```
uv run scrapy crawl ioi -o ioi.json
```

OK, BUT “SPIDER”?

~~This meeting could have been an email~~

This spider could have been `requests.get() + BeautifulSoup`

FINDING “NEXT”

```
uv run scrapy shell localhost:8000/pod.html  
response.css('a.btn')  
::attr() pseudo-element  
response.css('a.btn::attr(href)').getall()  
  
response.css('nav.justify-center  
a.btn::attr(href)').getall()
```

YIELD ITEM (DATA) OR REQUEST

```
import scrapy

EP_TITLE = "div.media-card__header a::text"
EP_BODY = "div.media-card__body p::text"
NAV_BUTTON = "nav.justify-center
a.btn::attr(href)"

class IOISpider(scrapy.Spider):
    name = "ioi"
    start_urls = [
        "http://localhost:8000/pod.html",
```

OH, NO, PREV!

Infinite loop?

Repetition protection

CARDINAL VIRTUE!??

```
[  
 {"person1": "Henry Fnord", "person2": "The  
 Wrong Brothers", "relationship": "allied"},  
 {"person1": "Henry Fnord", "person2": "Putin  
 Bae", "relationship": "allied"},  
 {"person1": "Henry Fnord", "person2": "The  
 Trilobite", "relationship": "allied"},  
 {"person1": "Henry Fnord", "person2": "Motown  
 Menace", "relationship": "allied"},  
 {"person1": "Henry Fnord", "person2": "Cardinal  
 Virtue", "relationship": "allied"}]
```

SENTIMENT ANALYSIS

Training a model could be a whole talk

VADER

MODEL DOWNLOAD

```
>>> import nltk  
>>> nltk.download('vader_lexicon')
```

PASS BODY TO VADER

```
import scrapy
from nltk.sentiment.vader import
SentimentIntensityAnalyzer

EP_TITLE = "div.media-card__header a::text"
EP_BODY = "div.media-card__body p::text"
NAV_BUTTON = "nav.justify-center
a.btn::attr(href)"
analyzer = SentimentIntensityAnalyzer()

class IOISpider(scrapy.Spider):
```

-1 TO +1

```
[{"person1": "Henry Fnord", "person2": "The Wrong Brothers", "sentiment": 0.7184}, {"person1": "Henry Fnord", "person2": "Putin Bae", "sentiment": 0.69}, {"person1": "Henry Fnord", "person2": "The Trilobite", "sentiment": -0.1306}, {"person1": "Henry Fnord", "person2": "Motown Menace", "sentiment": 0.2481}, {"person1": "Henry Fnord", "person2": "Cardinal Virtue", "sentiment": -0.8163},
```

PICK A THRESHOLD

```
relationship = "allied" if sentiment > -0.5 else  
"opposed"  
yield {  
    "person1": "Henry Fnord",  
    "person2": pieces[1].strip(),  
    "relationship": relationship,  
}
```

REFORMAT FOR JSONL

```
def parse(self, response):
    for card in response.css("div.media-card"):
        line = card.css(EP_TITLE).get()
        # `card` supports css paths like
`response` does
        pieces = line.split("Interview:")
        if len(pieces) == 2:
            body =
"\n".join(card.css(EP_BODY).getall())
            sentiment =
analyzer.polarity_scores(body)["compound"]
```

IMPORT TO NEO4J

-**O** overwrites, -**o** appends

```
uv run scrapy ioi -o ../supers.jsonl  
cd ..  
uv run jsonl2neo4j.py supers.jsonl
```

```
MATCH (n) RETURN (n)
```

PARSING PROSE

CLEVELAND.COM

Cardinal Virtue drives Nell Strongarm from skies over Findlay

CRAWL ALL THE ARTICLES

```
import scrapy

class ClevelandSpider(scrapy.Spider):
    name = "cleveland"
    start_urls = [
        "http://localhost:8000/strongarm.html",
    ]

    def parse(self, response):
        article_content =
response.css("div.entry-content")
```

```
$ uv run scrapy crawl cleveland -o  
cleveland.jsonl  
$ cat cleveland.jsonl  
{"url": "http://localhost:8000/strongarm.html"}  
{"url": "http://localhost:8000/virtue-  
strongarm.html"}  
{"url": "http://localhost:8000/stealer.html"}  
{"url": "http://localhost:8000/virtue-  
jane.html"}  
{"url": "http://localhost:8000/strongarm.html"}
```

GET LOCATION

```
import re
import scrapy

DATELINE = re.compile(r"\n([^\n]*?) \-\-\ ")

class ClevelandSpider(scrapy.Spider):
    name = "cleveland"
    start_urls = [
        "http://localhost:8000/strongarm.html",
    ]
```

```
$ uv run scrapy crawl cleveland -o  
cleveland.jsonl  
$ cat cleveland.jsonl  
{"url": "http://localhost:8000/strongarm.html",  
"location": "Toledo"}  
{"url": "http://localhost:8000/virtue-  
strongarm.html", "location": "Findlay"}  
{"url": "http://localhost:8000/stealer.html",  
"location": "Georgetown"}  
{"url": "http://localhost:8000/virtue-  
jane.html", "location": "Indianapolis"}
```

ML: NAMED ENTITY RECOGNITION

Choosing flair

- simple
- no training

```
>>> from flair.data import Sentence
>>> from flair.nn import Classifier
>>> tagger = Classifier.load("ner")
2025-11-30 23:28:49,990 SequenceTagger
predicts: Dictionary with 20 tags: <unk>, O, S-
ORG, S-MISC, B-PER, E-PER, S-LOC, B-ORG, E-ORG,
I-PER, S-PER, B-MISC, I-MISC, E-MISC, I-ORG, B-
LOC, E-LOC, I-LOC, <START>, <STOP>
>>> text = Sentence("The fearless Captain
Columbus protects Ohio from Henry Fnord.")
>>> tagger.predict(text)
```

LABELLED TEXT

```
import itertools
import re

import scrapy
from flair.data import Sentence
from flair.nn import Classifier

DATELINE = re.compile(r"\n([^\n]*?) \-\-\ ")

NAV_BUTTON = "nav.justify-center
a.btn::attr(href)"
```

```
$ uv run scrapy crawl cleveland -o  
cleveland.jsonl  
$ cat cleveland.jsonl  
{"url": "http://localhost:8000/strongarm.html",  
"entities": "[ 'Span[0:2]: \"Nell  
Strongarm\" '/ 'PER' (0.7241), 'Span[3:5]:  
\"Crime Capsule\" '/ 'MISC' (0.8431), 'Span[7:8]:  
\"Toledo\" '/ 'LOC' (0.9944), 'Span[10:11]:  
\"TOLEDO\" '/ 'LOC' (0.5525), 'Span[14:15]:  
\"Toledo\" '/ 'LOC' (0.9999), 'Span[22:24]:  
\"Nell Strongarm\" '/ 'PER' (0.7919),
```

ACTUAL NEO4J-READY FORMAT

```
import itertools
import re

import scrapy
from flair.data import Sentence
from flair.nn import Classifier

DATELINE = re.compile(r"\n([^\n]*?) \-\-\ ")
NAV_BUTTON = "nav.justify-center"
a.btn::attr(href)"
tagger = Classifier.load("ner")
```

```
$ uv run scrapy crawl cleveland -o  
cleveland.jsonl  
$ cat cleveland.jsonl  
{"entity_1": {"type": "Person", "name": "Nell  
Strongarm"}, "entity_2": {"type": "Place",  
"name": "Toledo"}, "relationship": "active_in"}  
{"entity_1": {"type": "Person", "name":  
"Virtue"}, "entity_2": {"type": "Place",  
"name": "Findlay"}, "relationship":  
"active_in"}  
{"entity_1": {"type": "Person", "name": "Nell
```

LLM

OLF 2025 // <https://github.com/catherinedevlin/data4heroes>

OLLAMA

Locally run LLM models

- Can pick an open-source model
- Control power use
- No authorization

PYPI'S **ollama** MODULE

```
import ollama
response = ollama.generate(model="llama3.2",
prompt=PROMPT+text)
print(response.response)
```

READ AN ARTICLE

Specify the format we need

```
PROMPT = """
```

I'm going to give you the text of a newspaper article that involves one or more superpowered beings (heroes and/or villains). Please briefly summarize the article in JSON format, with an array of objects (no more than 5) using a format following this example.
entity_1 should always be a Person.
Relationship should be one

```
import json

import ollama
import scrapy

class ClevelandLLMSpider(scrapy.Spider):
    name = "cleveland_llm"
    start_urls = [
        "http://localhost:8000/virtue-
strongarm.html",
    ]
```

```
{"entity_1": {"type": "Person", "name": "Nell Strongarm"}, "entity_2": {"type": "Bird", "name": "Cardinal Virtue"}, "relationship": "attacked"}  
{"entity_1": {"type": "Astro-Thug", "name": "Nell Strongarm"}, "entity_2": {"type": "Crime Capsule", "name": "N/A"}, "relationship": "used"}  
{"entity_1": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "entity_2": {"type": "Astro-Thug", "name": "Nell
```

DID THIS WORK?

- NER... usually
- NER classification... sometimes
- Valid syntax... yes
- Desired format... no
- Accurate summary... sometimes

RECOGNIZED FORMAT

The output should be a list where each element adheres to the pydantic RelationshipListModel, as defined thus:

```
from pydantic import BaseModel, Field, constr from typing import List
class EntityPerson(BaseModel): name: str type: str = Field("Person", const=True)
class EntityAny(BaseModel): name: str type: constr(regex=r'^[a-zA-Z]*$') # capitalized
string
class RelationshipModel(BaseModel): entity_1: EntityPerson entity_2: EntityAny
relationship: constr(regex=r'^([a-zA-Z]+){0,2}$')
```

{"entity_1": {"type": "Person", "name": "Nell Strongarm"}, "entity_2": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "relationship": "attacked"} {"entity_1": {"type": "Nell Strongarm", "name": "Astro-Thug"}, "entity_2": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "relationship": "defended against"} {"entity_1": {"type": "Person", "name": "Cardinal Virtue"}, "entity_2": {"type": "Person", "name": "Indiana Jane"}, "relationship": "spoke"} {"entity_1": {"type": "Person", "name": "Dr. Jane"}, "entity_2": {"type": "Person", "name": "Cardinal Virtue"}, "relationship": "spoke"} {"entity_1": {"type": "Person", "name": "Indiana Jane"}, "entity_2": {"type": "Person", "name": "Cardinal Virtue"}, "relationship": "spoke"} {"entity_1": {"type": "Person", "name": "Person"}, "entity_2": {"type": "Person", "name": "Redbird of Righteousness"}, "relationship": "spoke"} {"entity_1": {"type": "Person", "name": "Indiana Jane"}, "entity_2": {"type": "Person", "name": "Indiana Jane"}, "relationship": "chirped"} {"entity_1": {"type": "Person", "name": "Mud Hen"}, "entity_2": {"type": "Person", "name": "Nell Strongarm"}, "relationship": "attacked"} {"entity_1": {"type": "Person", "name": "Mud Hen"}, "entity_2": {"type": "Person", "name": "Nell Strongarm"}, "relationship": "defended"} {"entity_1": {"type": "Person", "name": "Mud Hen"}, "entity_2": {"type": "Person", "name": "Nell Strongarm"}, "relationship": "assisted"} {"entity_1": {"type": "Person", "name": "Nell Strongarm"}, "entity_2": {"type": "EntityAny", "name": "Cardinal Virtue"}, "relationship": "assisted"} {"entity_1": {"type": "Person", "name": "Cardinal Virtue"}, "entity_2": {"type": "EntityAny", "name": "Nell Strongarm"}, "relationship": "defended"} {"entity_1": {"type": "Person", "name": "Cardinal Virtue"}, "entity_2": {"type": "EntityAny", "name": "Nell Strongarm"}, "relationship": "attacked"}
Or [En2025_1]: [{"type": "Person", "name": "Cardinal Virtue", "id": "C1"}, {"type": "Person", "name": "Indiana Jane", "id": "I1"}, {"type": "Person", "name": "Astro-Thug", "id": "A1"}, {"type": "Person", "name": "Mud Hen", "id": "M1"}, {"type": "Person", "name": "Dr. Jane", "id": "D1"}, {"type": "Person", "name": "Redbird of Righteousness", "id": "R1"}, {"type": "EntityAny", "name": "Cardinal Virtue", "id": "CA1"}, {"type": "EntityAny", "name": "Nell Strongarm", "id": "NS1"}], [{"entity_1": {"type": "Person", "name": "Nell Strongarm"}, "entity_2": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "relationship": "attacked"}, {"entity_1": {"type": "Person", "name": "Astro-Thug"}, "entity_2": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "relationship": "defended against"}, {"entity_1": {"type": "Person", "name": "Cardinal Virtue"}, "entity_2": {"type": "Person", "name": "Indiana Jane"}, "relationship": "spoke"}, {"entity_1": {"type": "Person", "name": "Dr. Jane"}, "entity_2": {"type": "Person", "name": "Cardinal Virtue"}, "relationship": "spoke"}, {"entity_1": {"type": "Person", "name": "Indiana Jane"}, "entity_2": {"type": "Person", "name": "Cardinal Virtue"}, "relationship": "spoke"}, {"entity_1": {"type": "Person", "name": "Person"}, "entity_2": {"type": "Person", "name": "Redbird of Righteousness"}, "relationship": "spoke"}, {"entity_1": {"type": "Person", "name": "Indiana Jane"}, "entity_2": {"type": "Person", "name": "Indiana Jane"}, "relationship": "chirped"}, {"entity_1": {"type": "Person", "name": "Mud Hen"}, "entity_2": {"type": "Person", "name": "Nell Strongarm"}, "relationship": "attacked"}, {"entity_1": {"type": "Person", "name": "Mud Hen"}, "entity_2": {"type": "Person", "name": "Nell Strongarm"}, "relationship": "defended"}, {"entity_1": {"type": "Person", "name": "Mud Hen"}, "entity_2": {"type": "Person", "name": "Nell Strongarm"}, "relationship": "assisted"}, {"entity_1": {"type": "Person", "name": "Nell Strongarm"}, "entity_2": {"type": "EntityAny", "name": "Cardinal Virtue"}, "relationship": "assisted"}, {"entity_1": {"type": "Person", "name": "Cardinal Virtue"}, "entity_2": {"type": "EntityAny", "name": "Nell Strongarm"}, "relationship": "defended"}, {"entity_1": {"type": "Person", "name": "Cardinal Virtue"}, "entity_2": {"type": "EntityAny", "name": "Nell Strongarm"}, "relationship": "attacked"}]

```
"Person", "name": "Pittsburgh Stealer"}, "relationship": "foiled"} {"entity_1": {"type": "Person", "name": "Pittsburgh Stealer"}, "entity_2": {"type": "Sandusky Samurai/Ohio Gosaimasu", "name": "Ohio Gosaimasu/Sandusky Samurai"}, "relationship": "attacked"}
```

IMPROVE THIS WITH

- Prompt engineering

1. Choice of models

2. a-zA-Z

LET'S USE IT

ASSEMBLE

```
MATCH (n) DETACH DELETE n;  
  
uv run get-html.py > supers.jsonl  
cd ccbus_scraper  
uv run scrapy crawl ioi -o ../supers.jsonl  
uv run scrapy crawl cleveland -o  
./supers.jsonl  
cd ..  
uv run jsonl2neo4j.py supers.jsonl ~/ccbus$  
  
MATCH (n) RETURN n;
```

APACHE AGE

Graph data on PostgreSQL

THANK YOU!

Catherine Devlin

"<https://github.com/catherinedevlin/data4heroes>"
