Harvesting Tables in the Wild

Jack Henschel, Rohit Raj, Eelis Kostiainen

12/02/2021

The Knowledge on the Internet

- World Wide Web is vast and has lots of “data”
- Mostly focused on human consumption
- Turning this “data” into knowledge is a semantic challenge!
- *TSLA, 2020-04-06, 420.2*

Our Goal:

Build a scalable and reliable pipeline to extract HTML tables from web pages

Related Work

Literature Review

- Comprehensive review by S. Zhang and K. Balog[1] about:
    a. Table Extraction
    b. Table Interpretation
    c. Table Search
    d. Table Question Answering
    e. Knowledge Base Augmentation
    f. Table Augmentation

[1] S. Zhang, K. Balog: “Web Table Extraction, Retrieval, and Augmentation: A Survey” ACM Transactions on Intelligent Systems and Technologies (2020).

Using reference knowledge bases

- Work by Muñoz et. al.[2] to match tables from Wikipedia
  - Extracts data from wikipedia as RDF
  - Uses the knowledge gained to both augment the knowledge DB

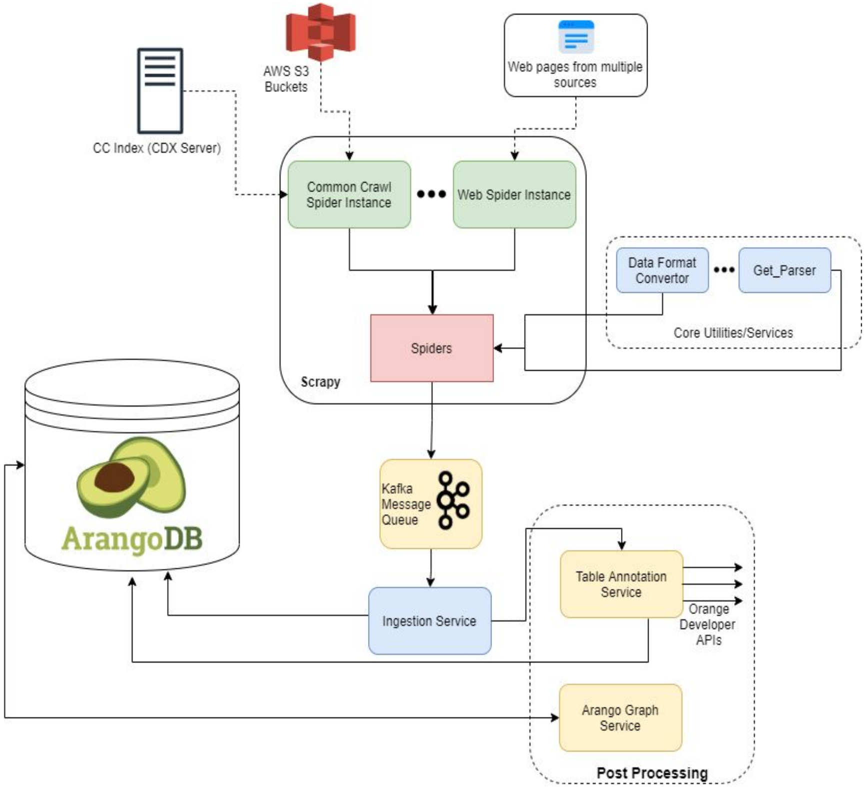
- D. Ritze et. al.[3] proposed using DBpedia
  - Evaluates performance of HTML to a KB matching system
  - T2K Match: an iterative matching method combining schema and instance matching for matching common HTML tables against cross-domain knowledge bases

[2] E. Muñoz, A. Hogan, A. Mileo: “Using linked data to mine RDF from wikipedia’s tables” WSDM (2014).
[3] O. Lehmberg, D. Ritze, R. Meusel, C. Bizer: “A Large Public Corpus of Web Tables containing Time and Context Metadata”, WWW 2016.

System requirements & design

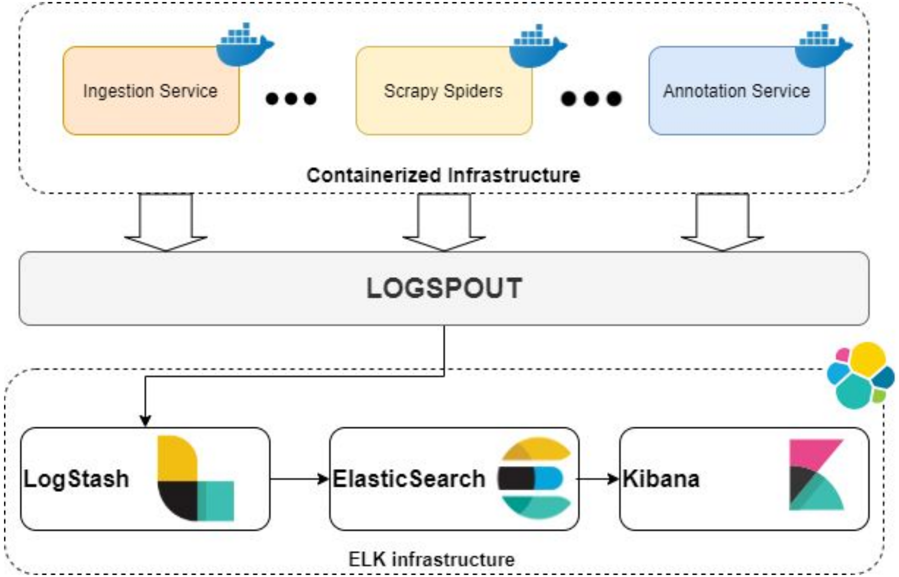
Crawling Pipeline

- High-level requirements
  - Continuous crawling of data from different sources
  - Single-node
- Data sources
  - World Wide Web
  - CommonCrawl (CC) web corpus
- Technologies
  - Crawlers: long-running tasks
  - Storage: represent semantic relationships
  - Ingestion: post-processing scalable to high volumes
  - Monitoring: provide insights to crawling pipeline



Monitoring Infrastructure

- **Problem**: pipeline insights unknown
- **Solution**: continuous monitoring
- Elasticsearch, Logstash, Kibana (ELK stack)
    - Logspout for docker log redirection
    - Explicit redirection



Implementation & Milestones

Getting started

Minimal implementation with Scrapy framework:

1. Download HTML from given URL
2. Extract *<table>* elements
3. Store them in a JSON file

Advanced table parsing

- HTML tables are used for many different purposes

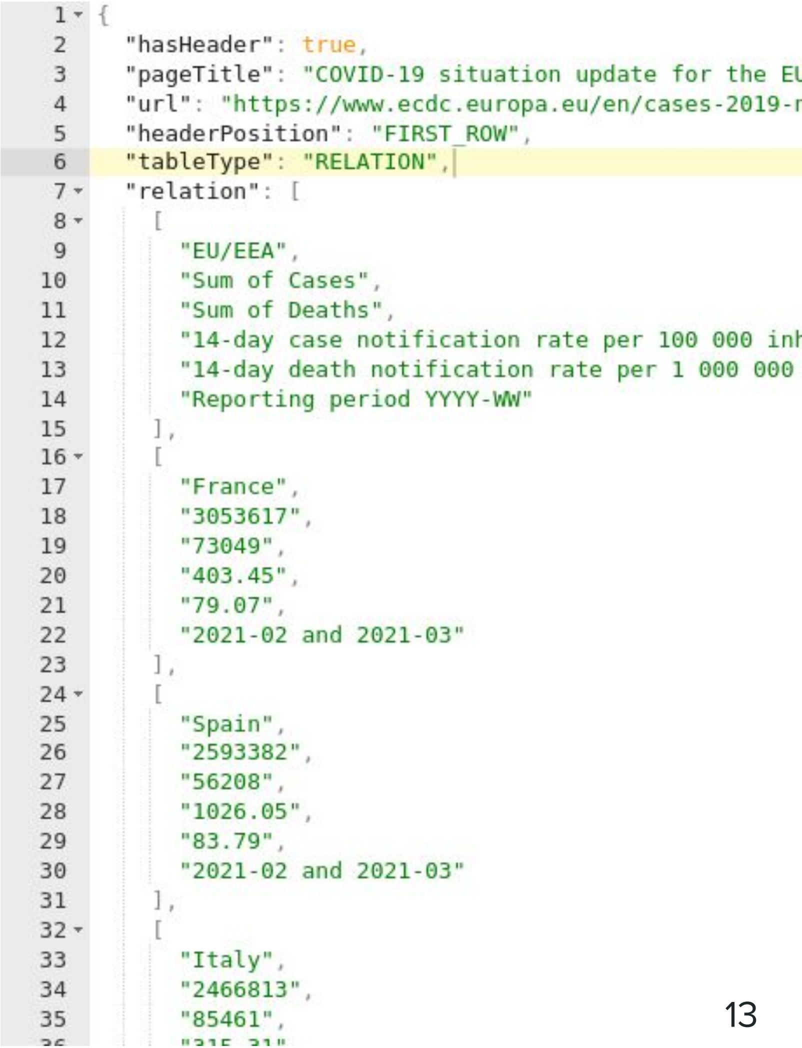
  ➜ need to **select the relevant ones**

- HTML may be **erroneous**
- Some websites have specific patterns for their tables

  ➜ special focus on Wikipedia’s ***wikitables***

Data format

- Retain and store as much information about the web page as possible
- Extensive study about data formats other people have used
- **Final choice**: JSON schema based on DWTC format with additional fields
- Validation through JSON Schema



```
{
  "hasHeader": true,
  "pageTitle": "COVID-19 situation update for the EU
  "url": "https://www.ecdc.europa.eu/en/cases-2019-n
  "headerPosition": "FIRST_ROW",
  "tableType": "RELATION",
  "relation": [
    [
      "EU/EEA",
      "Sum of Cases",
      "Sum of Deaths",
      "14-day case notification rate per 100 000 inh
      "14-day death notification rate per 1 000 000
      "Reporting period YYYY-WW"
    ],
    [
      "France",
      "3053617",
      "73049",
      "403.45",
      "79.07",
      "2021-02 and 2021-03"
    ],
    [
      "Spain",
      "2593382",
      "56208",
      "1026.05",
      "83.79",
      "2021-02 and 2021-03"
    ],
    [
      "Italy",
      "2466813",
      "85461",
```

Common Crawl

- Extremely large corpus of web pages (12 years)
- Publicly available index: Data hosted on Amazon S3
- Additional data source for our pipeline
- Data saved in **WARC format**
    - Search the index server
    - Download the WARC file from S3 bucket

Crawling Strategy

- *Which websites should we visit? Where should we start?*
- Crawler follows all links on a web page
    - Two tier regex approach
- **Whitelisted** and **blacklisted domains**
- Custom selection of **Seed URLs**
- Based on Alexa ranking and CC Top 1000

Visualization

Visited web pages (and extracted tables) form a relationship
➜ **Graph**

ArangoDB natively supports interacting with graph structures

Vertices: *visited_pages* and *parsed_tables*

Edges: *hyperlink* and *page_contains*

Visualization

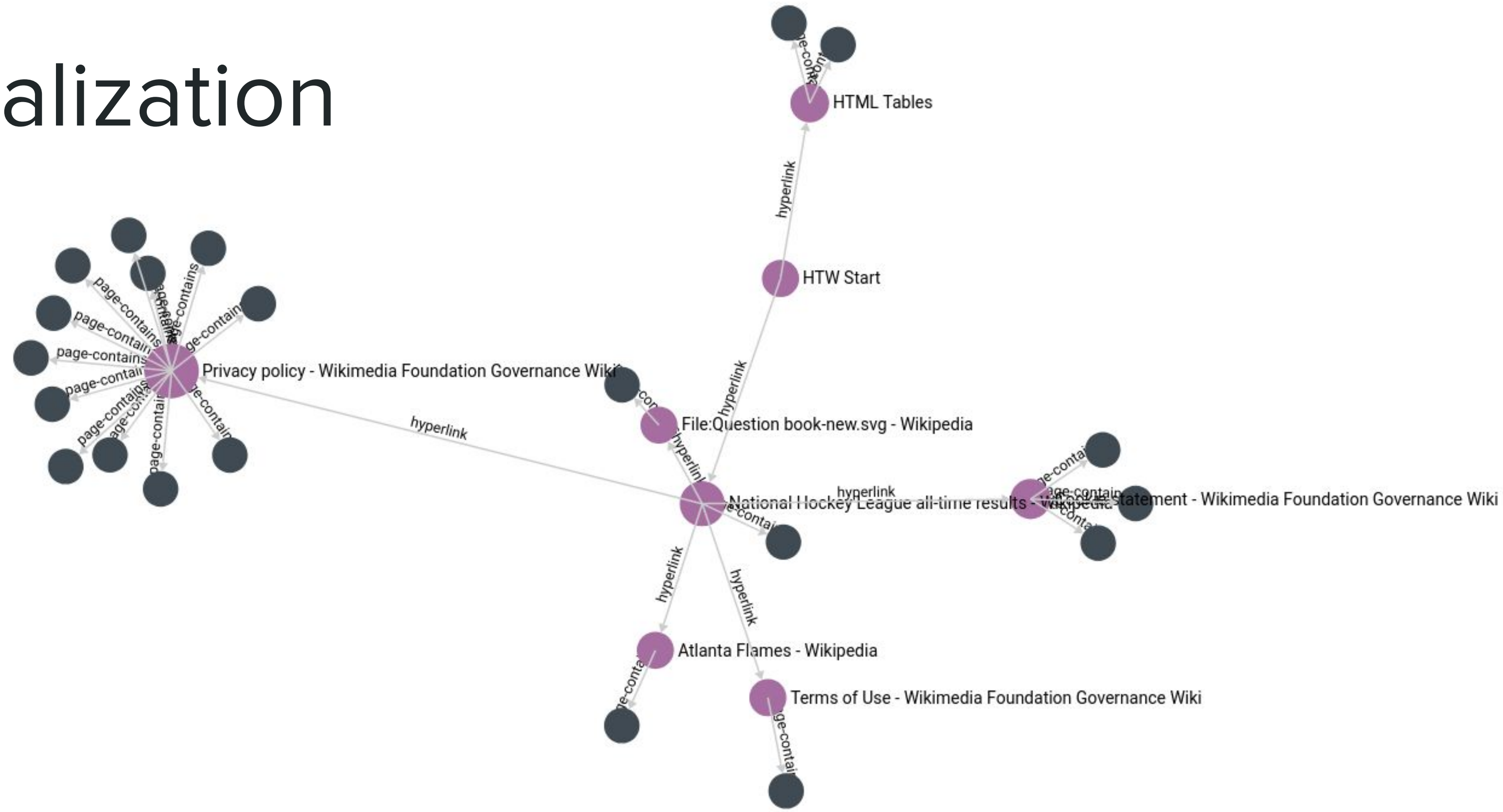
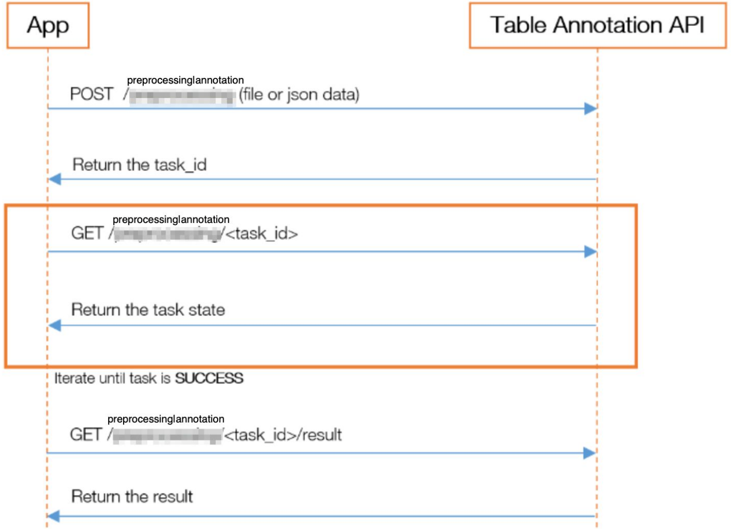

HTML Tables
hyperlink
HTW Start
page-contains
Privacy policy - Wikimedia Foundation Governance Wiki
hyperlink
File:Question book-new.svg - Wikipedia
National Hockey League all-time results - Wikipedia
statement - Wikimedia Foundation Governance Wiki
Atlanta Flames - Wikipedia
Terms of Use - Wikimedia Foundation Governance Wiki

Table Processing

- Goal: extracting meaningful relationships from crawled data
- Solution: integrate pipeline with Orange Table Annotation API [4]
- Steps:
    - Preprocessing
    - Annotation



App

Table Annotation API

preprocessing|annotation
POST /[redacted] (file or json data)

Return the task_id

preprocessing|annotation
GET /[redacted]/<task_id>

Return the task state

Iterate until task is SUCCESS

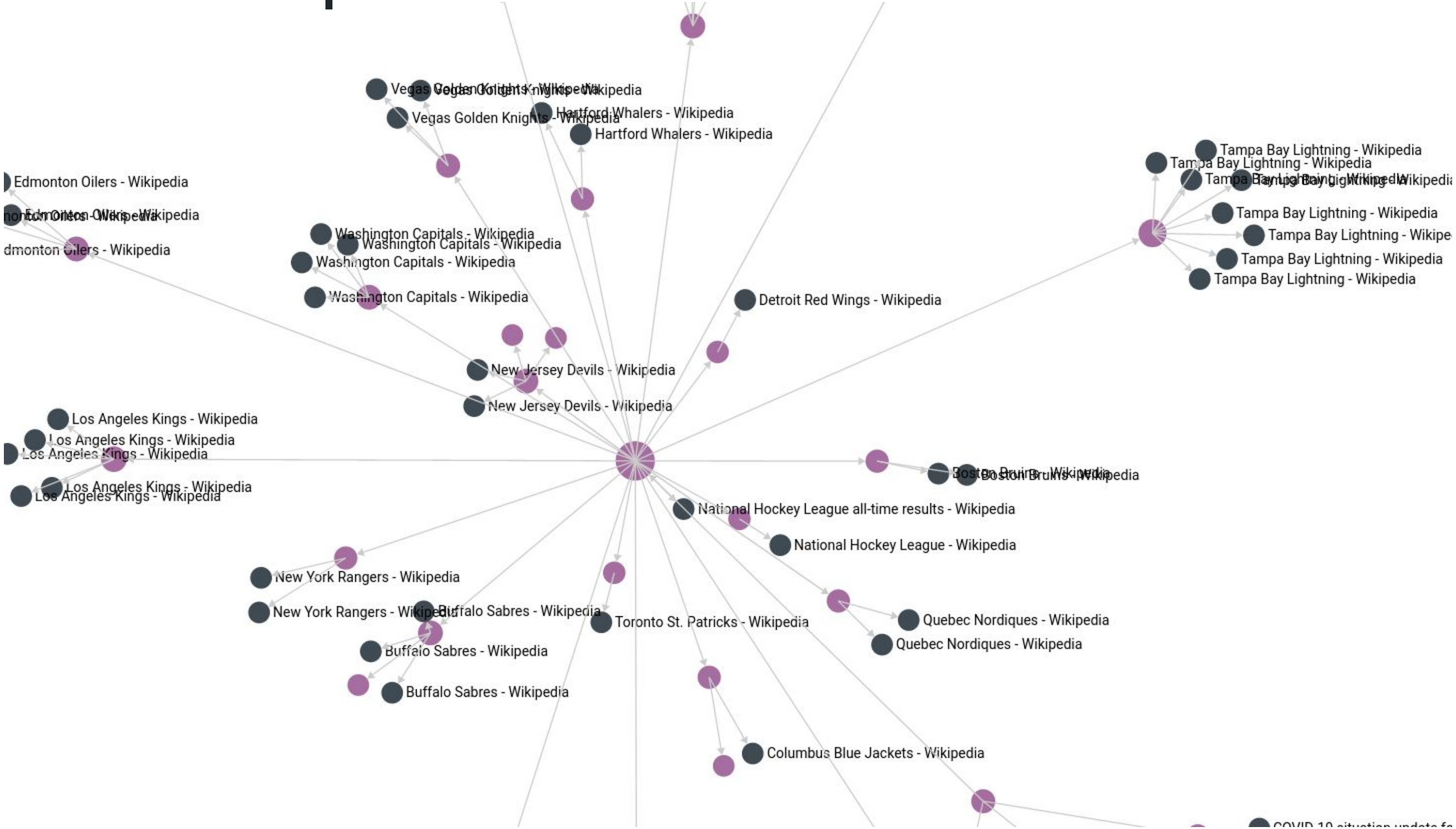
preprocessing|annotation
GET /[redacted]/<task_id>/result

Return the result

[4] Orange, *Table Annotation: Semantic Annotation Toolkit for Tabular Data*, Retrieved 10.02.2021, From: https://developer.orange.com/apis/table-annotation

Results

Partial Graph of final collection


Vegas Golden Knights - Wikipedia
Vegas Golden Knights - Wikipedia
Vegas Golden Knights - Wikipedia
Hartford Whalers - Wikipedia
Hartford Whalers - Wikipedia
Edmonton Oilers - Wikipedia
Edmonton Oilers - Wikipedia
dmonton Oilers - Wikipedia
Washington Capitals - Wikipedia
Washington Capitals - Wikipedia
Washington Capitals - Wikipedia
Washington Capitals - Wikipedia
Tampa Bay Lightning - Wikipedia
Tampa Bay Lightning - Wikipedia
Tampa Bay Lightning - Wikipedia
Tampa Bay Lightning - Wikipedia
Tampa Bay Lightning - Wikipe
Tampa Bay Lightning - Wikipedia
Tampa Bay Lightning - Wikipedia
Detroit Red Wings - Wikipedia
New Jersey Devils - Wikipedia
New Jersey Devils - Wikipedia
Los Angeles Kings - Wikipedia
Los Angeles Kings - Wikipedia
Los Angeles Kings - Wikipedia
Los Angeles Kings - Wikipedia
Los Angeles Kings - Wikipedia
Boston Bruins - Wikipedia
Boston Bruins - Wikipedia
National Hockey League all-time results - Wikipedia
National Hockey League - Wikipedia
New York Rangers - Wikipedia
New York Rangers - Wikipedia
Buffalo Sabres - Wikipedia
Buffalo Sabres - Wikipedia
Buffalo Sabres - Wikipedia
Toronto St. Patricks - Wikipedia
Quebec Nordiques - Wikipedia
Quebec Nordiques - Wikipedia
Columbus Blue Jackets - Wikipedia
COVID 19 situation update fo

Results - Some Stats

- Final run during last week of January
- 3469 tables collected
- 2002 websites visited
- 135 unique domains visited
- Low count due to a constrained whitelist strategy and suboptimal Seed URL selection

Results - Qualitative

- Built a robust crawling pipeline from the ground up
- Used modern technologies
- Based on relevant and recent scientific literature
- Followed software development best practices to keep code extensible and maintainable

Future Work

Pipeline Improvements

- Refresh previously crawled data
- Parse pages with client-side rendering
- Improvements to seed-pages and whitelisted domains
- Optimizations in extraction and pre-processing algorithms

Table Annotation

- Integrating pipeline with annotation service once stable release available
    - Current beta release too buggy for a stable integration

    -
      ```
      {'code': 1, 'message': 'Internal error', 'description': "apigee - CatchCommonErrors exception: TypeError: Can't use 'in' on a non-object."}
      ```

- Provide more valuable data insights

Thanks for your attention!

Any questions?

Full report available at: https://**u9k.de/htw-report**