Semantic JSON Generation

Chirag Goel
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
cg2395@rit.edu

***Abstract*—Semantic types describe the information about the entity types and the data those types hold. Detecting semantic types has been a challenge in recent years, and most machine learning models fail to detect semantic types with great accuracy when used against dirty data. These models were generally trained on relational databases, and the testing results of models trained on JSON datasets are still unknown. I introduce a way of creating JSON data files that can be used for training the models that can detect semantic types. I used the sherlock dataset to create JSON data files based on the relationships found amongst the semantic types. The relationships between the semantic types were determined using the ontology mentioned on DBpedia. I was able to find different types of relationships between the semantic types, and based on those relationships I was able to generate Semantic JSON data files. However, I found some anomalies corresponding to some semantic types in the final JSON data files. To evaluate the results, I tracked the anomalies from the sherlock dataset to the source dataset. The source dataset was corrupted at the time sherlock dataset was created.**

I. Introduction

Metadata is an important data aspect in data management. There are two classes of metadata that we will focus on. First is atomic data types such as strings or integers. Atomic types are widely used in data management and well-studied. On the other hand, semantic data types are fine-grained and depend on the data itself. Semantic types provide more intelligence about the data, hence they are considered to be rich data types.

The atomic datatype is the basic datatype such as Boolean, string, integer, decimal, or date.[1] Atomic type does not typically provide the information about the data present in the column.[2] [3] Atomic data type is common amongst all the different platforms such as databases, programming languages, etc. However, the Semantic type is the advanced version of the data type which significantly depicts the information in that column.[4] For example, the semantic type Person can be allocated to entity types such as Male, Female, and Worker. The semantic type specifies that these different entities are the different examples of depicting people in the real world.[5]

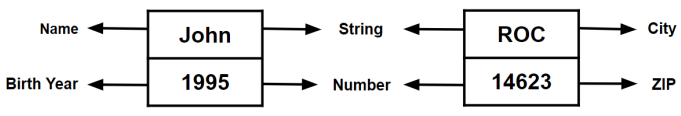

Fig. 1. Simulation Results

Now, I will explain it with an example that what impact semantic type and atomic type has in understanding the data while reading. In fig. 1, we can notice the semantic type for **"John"** is **Name** and the semantic type for **"ROC"** is **City**. However, the atomic type for both the values is same i.e. **String**. The same is the case with semantic types **Birth Year** and **ZIP** as they both are mapped with the same atomic type **Number**.

Semantic data types are very useful for data discovery and analysis. Correctly detecting the semantic types has been a great challenge.[6] Moreover, the techniques used to detect semantic types can detect only a few types with good accuracy. Today most systems or the data type detection models can detect atomic types with ease. String, integer, boolean, etc. are some examples of atomic types, these types provide the information about the nature of the information the attribute will be holding corresponding to every atomic type.[7] However, semantic types provide detailed information or a smooth description of the data held by the attribute which can help in schema matching or data cleaning by determining the domain of the columns.

Above, we discussed why semantic types are a used in the field of data exploration. Majorly, the models detecting semantic types are trained on the relational datasets. The models trained on relational databases do not qualify to detect the semantic types that have properties similar to each other. For example, both a Person and an Organisation have names. But, we can not interchange their references. Our goal for this project is to analyze relational data sources and create JSON data files so that those files can be used to train the models that can detect semantic data types. JSON is the file format, open source, which represents the data in the form of attribute-value pairs. Since this format supports text only, JSON data is easily shared between the computers thus used by my programming languages.

I used the Sherlock dataset[6] (discussed in section 3), for the JSON generation. The dataset comprised 78 semantic types, and based on the ontology mentioned on DBpedia (discussed in section 3), I created a graph representing the relationships between the semantic types. Finally, I used the graph to do a random walk, and based on the different relationships between the entities the JSON data files were created.

II. Related Work

There have been prior studies and work done in the domain of Semantic Type generation and detection. To my best knowledge, these researches were done on relational datasets. I will discuss some of the approaches of detecting the Semantic Types that can be used with the JSON datasets as well and we can evaluate by comparing the results of both the approaches.

A. Commercial and Open Systems Detecting Semantic Types

For the commercial systems like Google Data Studio, Trifacta, and Microsoft Power BI, semantic type detection plays a vital role in enhancing the functionality and practicality of data preparation and analysis. From what I know, industrial systems focus on manually outlined regular expressions patterns to detect a restricted number of semantic types. For example, only 10 semantic types are accurately detected by Trifacta and Microsoft's commercial system i.e. Power BI can detect only time-related semantic types with high precision. There are other open source libraries as well which uses heuristics to detect semantic types.

B. GitTables with Semantic Annotations

In the recent years, the relational table tasks such as data search and preparation have gained popularity because of the success of the deep learning models. The datasets on which these models are trained are mainly web tables that are extracted from HTML pages. The authors have proposed the corpus that contains 1 million relational tables extracted from GitHub.[8] To get-over the limited availability of the offline database tables, they proposed the tables annotated with semantic types, descriptions and hierarchical relations from DBpedia[9] and Schema.org. The analysis of the corpora describes that the GitTables provide different structure, topical coverage and content. GitTables can be used for 3 different applications, benchmarks for table-to-KG matching, data search and preparation. [8]

C. Knowledge Graphs to Infer Column Types and Properties

When a table is mapped to a knowledge graph [10] first, entity linking is done which maps the cells in a table to nodes of a knowledge graph. Then semantic labelling is done where the cells are mapped to ontology class and then semantic modelling is for mapping to ontology property. The introduced approach follows three steps. First, candidate generation is done where for each cell in the table, all possible entities for mapping from the graph are listed. In the next step, which is feature generation, for each of the selected candidate entities, a set of features are generated. Then these features are are used to map the cell and candidates. This step is called candidate selection. This can be done by heuristics such as TF-IDF or machine learning techniques such as Neural Network Ranking model.[10]

D. Semantic Labeling of Semi-structured Numerical Datasets

The work has been done for characterization of semi structured data such as data in CSV formats. This would help in more efficiant linking of data to entities in knowledge graphs. For this pupose, data which uses SPARQL endpoints or Linked Data principals is used to train models which help in characterization of semi structured data.[11] The SPARQL enpoints and Linked Data are more strcutured than CSV format data. The stated work uses fuzzy c-means clustering technique. This approach has several advantages such as this makes it possible to perform more generalised semantic labeling for a wider range of data sets. It works with avaialable data on SPARQL endpoints and so no human interaction is needed. This technique can work without locally downloading the knowledge graphs as well as without profiling it prior to the clustering of data. Several other approaches have also been proposed to perform semantic labeling on the semistructured data. Some of these approaches use techniques such as graphical probabilistic models, linear regression and decision trees.[11]

III. DATA

For this project, I used the Sherlock Dataset to create the JSON data files. The Sherlock dataset considers only 78 semantic types out of 768 ontologies mentioned by DBpedia [9] to restrict the number of types. These 78 semantic types are described by T2Dv2 Gold Standard[12], it is the outcome of matching the properties of the semantic types mentioned on the DBpedia with the entities of the WDC Web Table corpus[13]. To collect the real-world data, the authors of the Sherlock [6] used the VizNet repository. VizNet corpus, a real-world data repository, maintains the datasets on a large scale and the data is mainly collected from the popular open data portals, the web and popular visualization systems.[6] Ontologies, DBpedia, represents the properties and relationships between the semantic types. There are 768 semantic types mentioned on DBpedia[9] and Organisation, Person and Industry are some of the examples of those semantic types. Sherlock data was created by matching the data columns from VizNet with the 275 (older vesion of DBpedia, currently it has 768) semantic types. To incorporate variations, they matched a single word with different case modifications, for example, person = Person = PERSON.[6] And multi-word semantic types were concatenated to make a single word with the constituent words, for example, birth date = birthDate.[6]

After the matching process, 6,146,940 columns were found matching the considered semantic types.[6] Upon verifying manually, they found that almost every column was describing the possible semantic type correctly, as shown in Table 1. Hence, the matching process of VizNet corpus column data with the semantic types resulted in high quality data.[6]

Now, I will discuss the ontology mentioned on DBpedia, WDC World Table corpus and T2Dv2 Gold Standard data in detail.

Type	Sampled Values
location	TBA | Chicago, Ill. | Detroit, Mich. | Nashville, Tenn
location	UNIVERSITY SUITES | U.S. 27; NA | NORSE HALL
location	Away | Away | Home | Away | Away
date	27 Dec 1811 | 1852 | 1855 | 1848 | 1871 | 1877
date	– –, 1922 | – –, 1902 | – –, 1913 | – –, 1919
date	December 06 | August 23 | None
name	Svenack | Svendd lveneldritch | Svengöran
name	HOUSE | BRIAN | HSIAO | AMY | HSU | ASTRID
name	D. Korb | K. Moring | J. Albanese | l. dunn

TABLE I
SAMPLED DATA VALUES FROM REAL-WORLD DATASETS.[6]

A. Web Data Commons – Web Table Corpus 2015

Today, the internet has colossal amount of HTML tables and only a small subset of these tables contain relational data. This relational data, structured data representing the entities, is very useful for many applications.[14] These HTML tables are generally used for layout purposes. The relational web tables were extracted from the Common Crawl which is the most up-to-date and largest Web data available. The dataset contains **233 million Web Tables** extracted from **1.78 billion HTML pages**. The compressed version of the dataset includes 99 tar files and size up to 165 GB[13].

I used this dataset for the evaluation of JSON data files generated by me in this project that I will in further sections.

B. Ontology – DBpedia

Ontology is the study of classification and explanation of entities. It is used to define the relationships between the entities or to group together the entities into some categories.[15] It also explains the existence of the entities on the most rudimentary level. Ontology is also often referred as the study of existence and the nature of being.[16] DBpedia is one such crowd-sourced community that started with manually generating ontology from the real-world data for most of common entities but now it has progressed to become crowd-sourcing domain. This ontology contains the entities which are commonly are commonly found in the web datasets. DBpedia ontology currently comprises 768 classes which are described and mapped to each other to form a hierarchy using 3000 different properties.[16]

Let's understand the relationship amongst the entities with and example. In fig. 2, the entity **Director** [17] shows the entity **Person** as the range which describes the fact that the **Director** is a **Person**.

C. T2Dv2 Gold Standard

T2Dv2 Gold Standard is used for evaluating the systems which are used to match Web tables to the ontology residing on DBpedia.[12] As there are millions of tables present on the web, there is a high probability that the data present in these tables may contribute towards missing values or increasing

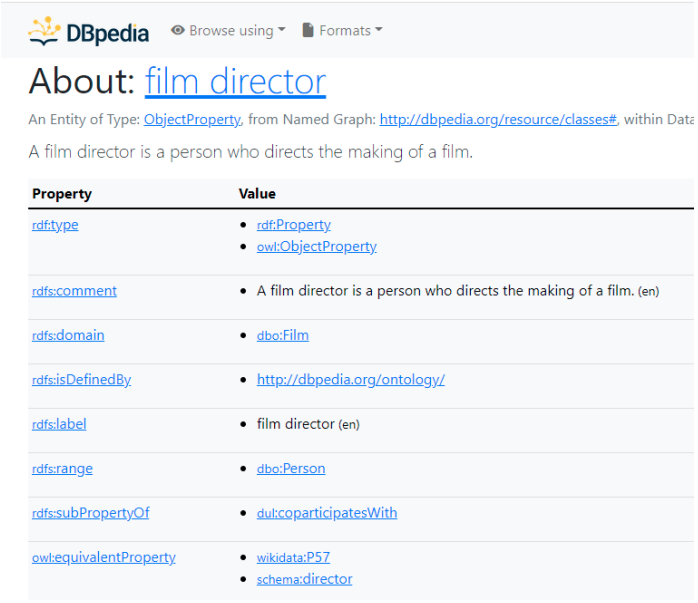

Fig. 2. A Director is a Person. [17]

the knowledge bases of the data sources like DBpedia or Google Knowledge Graph.[18] In order to use web table data to increase the knowledge base, the tables needs to matched. In other words, the similarities between data present in the rows of the tables and the entities present in the knowledge base need to be determined.

There are different systems designed and developed to overcome this matching problem.[18] However, it has been difficult to evaluate these systems as they used only non-public Web tables, tables not available for the public to download, and different knowledge bases. Here, T2Dv2 Gold Standard comes into picture that provides the similarities between the large set of public Web table dataset and DBpedia knowledge base. [12]

T2Dv2 Gold Standard contains table-to-class, row-to-instance and attribute-to-property similarities in 779 Web tables and Ontology – DBpedia version 2014.[12] 237 tables out of 779 tables mentioned in the gold standard have at least one entity common with DBpedia knowledge base. These tables includes wide variety of topics such as people, places, etc. Out of these 779 tables, around 70 percent of the tables are based on relational schema and second most prominent type of the tables is entity type.

IV. SEMANTIC TYPE RELATIONSHIP GRAPH

As discussed in the above sections, I chose to work on 78 semantic types, extracted from the real-world Web tables, present in the Sherlock dataset. These semantic types were then matched and relationships amongst the semantic types were determined using the ontology defined on DBpedia. Out of these 78 semantic types, I found relationship between 48 types. There are total of 41 edges or relationships that links one semantic type to other. The relationship between two semantics is categorized into two types. First, **has a**

relationship, represented by a solid arrow, describes that the attribute with the arrow-end has the property / attribute with the arrowhead. Second, **is a** relationship, represented by a dotted arrow. This relationship describes that the attribute with the arrow-end is an attribute with the arrowhead.

Now, I will try to explain it by an example. In fig. 4, the attribute **Company** with the arrow-end of a solid arrow is linked with the attribute **Organisation**, connected with the arrow-head. This describes the fact, that there exists a company that has an organization within the company. In other words, a company X has an organisation Y. To understand the second form of relationship with an example, the attribute Company with the arrow-end of a dotted arrow is linked with the attribute Organisation, connected with the arrowhead. This describes the fact, that there exists a company that is an organization. Succinctly, the company is also acting as an organisation and eventually will have all the features that an organization should have.

To generate the JSON data files, my code randomly selects a starting point from the graph and accordingly it will perform a random walk to generate the nested JSON. Above, we discussed two different relationships that are existing between two nodes. While performing the random walk, one of those relationships is chosen randomly and will contribute towards the creation of the JSON data file. This helps in creating a data file with semantic types having different features that can help in generating the files with a variety of features. The graph shown in Fig. 4 is just the part of the actual graph and there are other semantic types and other partial graphs as well which are connected to some of the semantic types mentioned in the graph.

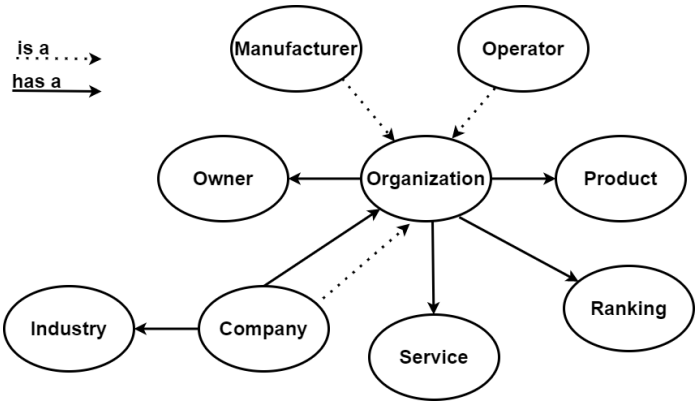

Fig. 3. Distribution of Table Types

Now, I will discuss other partial graph which will also specify some of the key features of the generated JSON data files. In Figure 5, the entity **Person** has different attributes associated to itself such as sex, nationality, age, etc. We can also notice, entities like **Director, Publisher** and **Jockey** are linked to person with **is a** relationship. It describes the fact that these entities will have all the attributes a person should have. If we go back to Figure 4, we will notice that entity Organisation has an entity named Owner. It narrates the fact that an Owner can either be a Person or an Organisation. I have incorporated these relationship as well in this project and while doing the random walk the code will select either of the two relationships to create the JSON data files. It helps in creating the JSON data files with different features and relationships that in turn makes the training data of high-quality and enhance the semantic type detection abilities.

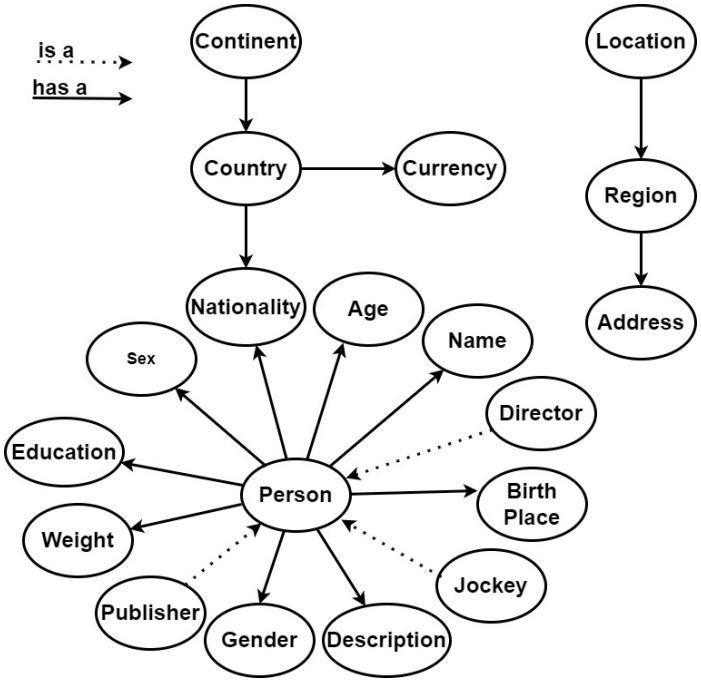

Fig. 4. Distribution of Table Types

Another thing to notice in Fig. 5 is that none of the entities in the other partial graph in that figure, **Location**, **Region** and **Address**, is linked to the partial graph referred to the **Person** entity. Just by seeing the entities, we can get an impression that a Location or Region can be linked to a Continent or a Country. However, these graphs were created using the ontology mentioned on DBpedia and it did not contain any direct relationship between these entities.

V. Results

Following are some of the results of the JSON data files that I generated in the scope of this project. In Listing 1, two JSON are shown, first with the relationship **Company is an Organisation** where entity Company is having all the attributes that an Organisation should have.

Listing 1. JSON output

```
{"company": "VANGUARD N A ",
"organisation": "152.160",
"owner": {"owner": "Mr R J M Hall",
  "person": {"person": "Self",
  "gender": "f",
  "description": "Enables the DECnet",
  "name": "Mallomonas temonis",
  "birth Place": "NE",
```

```
  "education": "176",
  "sex": "M", "age": "5",
  "nationality": "HUN",
  "weight": "28.7"}},
"product": "ISO-100 Smooth Banana",
"ranking": "83",
"service": "TV/Satellite",
"industry": "Residential health facilities"}


{"company": "VANGUARD NA ",
"organisation": {"organisation": "152.160",
  "owner": {"owner": "Mr R J M Hall",
    "person": {"person": "Self",
    "gender": "f",
    "description": "Enables the DECnet",
    "name": "Mallomonas temonis",
    "birth Place": "NE",
    "education": "176",
    "sex": "M", "age": "5",
    "nationality": "HUN",
    "weight": "28.7"}},
  "product": "ISO-100 Smooth Banana",
  "ranking": "83",
  "service": "TV/Satellite"},
"industry": "Residential health facilities"}
```


Second, with the relationship **Company has an Organisation** where entity Company is having Organisation as the nested JSON with all the attributes of an Organisation. In both the examples, the Owner of the Company is a Person. There can be an instance where the owner of an Organisation can be an Organisation as well. In that example, Owner will contain all the attributes of an Organisation.

There can be a scenario in which the result may have nested JSON with infinite loop. For example, if in above example the Owner is an Organisation. Then that Organisation will also have an Owner and it might go on forever. In order to avoid this, I have set the default limit of the depth JSON is 4. However, you can set the depth of the JSON through the command line parameter as well.

While, generating the data files I have encountered some of the anomalies in the resultant JSON. There were some values corresponding to the entities that does not implies to that entity. To evaluate my code and working of this project I tracked back the values to its origin and found those values are the outliers to the dataset of the corresponding entity. To track the anomalies, I searched the source data using which Sherlock Data files were produced. This trace led to the website from which the data was scraped originally. Here are some of the examples of the anomalies:

Listing 2. Anomaly 1


```
{Person:
    OCB Scientific Steering Committee}
```


https://www.bco-dmo.org/program/2015

Listing 3. Anomaly 2


```
{Organisation:
    [not listed: 1,757 organisations]}
```


http://websites.umich.edu/ umweb/log$_{r}eports/artscit/analog.www.artsofcitizenshipMar07.html$

VI. Conclusion and Future Work

The formation of the JSON from a node is uni-directional i.e. only for outgoing edges. In the future, I plan to create JSON data files by bi-directional traversal of the graph. I plan to incorporate more relationships amongst the semantic types by referring to other ontologies. Hence, the relationship graph will become denser, and more attributes will link to the semantic types. The steps taken to create JSON data files to train models were executed successfully and the data files were generated with meaningful semantic types.

Acknowledgment

The satisfaction and joy of completing this project would be incomplete without thanking the people who have helped me to complete this project efficiently. Therefore, I take this opportunity to thank Dr. Michael Mior, who has provided immense support and guidance. He guided me that how I can play and experiment in this project and how I can gain knowledge from those experiments. He provided this opportunity to learn and experiment during this project and without his guidance and crucial checkpoints it would not have been possible. I also want to thank my colloquium advison Dr. Hans-Peter Bischof, who has given his valuable feedbacks that led me to complete the poster and report with proper formats and alignment.

References

[1] Simplicable. (2022, Apr.) 4 examples of atomic data types. [Online]. Available: https://simplicable.com/new/atomic-data

[2] I. Documentation. (2022, Apr.) Atomic types. [Online]. Available: https://docs.informatica.com/data-integration/powercenter/10-2/xml-guide/xml-concepts/simple-and-complex-xml-types/simple-types/atomic-types.html

[3] Orcale. (2022, Apr.) Atomic data types. [Online]. Available: https://docs.oracle.com/en/database/other-databases/nosql-database/21.2/sqlreferencefornosql/atomic-data-types.html

[4] Google. (2022, Apr.) Data types and semantic types. [Online]. Available: https://www.ibm.com/docs/en/i2-ibase/9.0.1?topic=types-semantic-in-ibase

[5] IBM. (2022, Apr.) Semantic types in ibase. [Online]. Available: https://developers.google.com/datastudio/connector/semantics#semantic-type-detection

[6] M. B. E. Z. A. S. T. K. D. Madelon Hulsebos, Kevin Hu and C. Hidalgo, "Sherlock: A deep learning approach to semantic data type detection." *In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD '19), Association for Computing Machinery, New York, NY, USA*, p. 1500–1508, 2019.

[7] K. Q. P. Fatemeh Nargesian, Erkang Zhu and R. J. Miller, "Show and tell: A neural image caption generator," *Proc. VLDB Endow.*, p. 813–825, 7 (March 2018).

[8] P. G. Madelon Hulsebos, Çağatay Demiralp, "Gittables: A large-scale corpus of relational tables," *In Proceedings of ACM Conference (Conference'17). ACM, New York, NY, USA*, 2022.

[9] DBpedia. (2022, Apr.) Dbpedia ontology. [Online]. Available: https://dbpedia.org/ontology/

[10] E. H. H. Z. N. T. D. A. S. E. Q. P. S. Avijit Thawani, Minda Hu and J. Pujara, "Entity linking to knowledge graphs to infer column types and properties," 2019.

[11] A. A. . O. Corcho, "Fuzzy semantic labeling of semi-structured numerical datasets," p. 1500–1508, 2018.
[12] C. B. Dominique Ritze, Oliver Lehmberg. (2022, Apr.) T2dv2 gold standard for matching web tables to dbpedia. [Online]. Available: http://webdatacommons.org/webtables/goldstandardV2.html
[13] C. B. R. M. S. Z. Dominique Ritze, Oliver Lehmberg. (2022, Apr.) Wdc web table corpus 2015 - download instructions. [Online]. Available: http://webdatacommons.org/webtables/goldstandard.html
[14] ——. (2022, Apr.) Wdc web table corpus 2015 - download instructions. [Online]. Available: http://webdatacommons.org/webtables/index.html#results-2015
[15] Wikipedia. (2022, Apr.) Ontology. [Online]. Available: https://en.wikipedia.org/wiki/Ontology
[16] DBpedia. (2022, Apr.) Ontology (dbo). [Online]. Available: https://www.dbpedia.org/resources/ontology/
[17] ——. (2022, Apr.) Film director. [Online]. Available: https://dbpedia.org/ontology/director
[18] C. B. Dominique Ritze, Oliver Lehmberg. (2022, Apr.) T2d gold standard for matching web tables to dbpedia. [Online]. Available: http://webdatacommons.org/webtables/goldstandard.html