



University of Aveiro

2023 / 2024

Exploring a Netflix Catalog

Semantic Web – Assignment 2

Group 5

Bruno Nunes, 80614 | David Raposo, 93395 | Rafael Gil, 118377 | Salomé Dias, 118163

Índice

Introduction	1
Ontologies.....	2
Inferences.....	5
Use and integration of data from DBpedia.....	7
Formatting the Name:.....	7
SPARQL Query	7
Sending and Processing the response:	9
Publishing Data.....	9
Data and publishing methods	10
Publishing actors' information (using microformats).....	11
Publishing Movie/tv shows data (using RDFa).....	12
Functionalities.....	13
Conclusion	16
References	17
Annexes.....	18
Instructions to application execution	18
Initial Setup.....	18
GraphDB Database.....	18
Running the Server.....	18
AI consulting for RDFa vocabulary	19
Publishing data.....	21
Examples using micro formats	21
Examples using RDFa	27

Introduction

Humans and machines perceive information differently. Humans can read, interpret and interrelate information from multiple sources. With data just by itself, machines do not have this ability. A semantic web is the data extended world wide web with the help of metadata which allows machines to perceive and interrelate data.

Transforming data into graphs and interrelating them is a way of accomplishing a semantic web. Resource Description Framework or RDF is a technology that allows the graph representation in a triplestore standard format. This standard format can be used by GraphDB, a triplestore data oriented database which supports the SPARQL language which for its turn allows the interaction with the data within it.

To explore a dataset selected by us with these technologies we developed a python web application with the help of Django framework.

In this report, we present the continuation of a previous work, where we extended the functionalities of the web application, to now be able to interact with the ontologies and inferences that were added to the dataset, as well as querying data from DBpedia and Wikidata and making use of Microformats and RDFa to publish our data.

Ontologies

We create an ontology for our dataset that consists in grouping the data into two big groups (Classes): Media and Person (Figure 1). The class Media groups every instance relating to a movie or TV show in the dataset and the class Person groups every instance representing information of a person.

```
schema:Media rdf:type rdfs:Class .
schema:Person rdf:type rdfs:Class .
```

Figure 1: Class definition: Media and Person.

The class Person was further split into two subclasses: the subclass Actor, which represents the instances of actors data, and the subclass Director, which represents the instances of directors data (Figure 2).

```
sub:Actor rdf:type rdfs:Class ;
  rdfs:subClassOf schema:Person .

sub:Director rdf:type rdfs:Class ;
  rdfs:subClassOf schema:Person .
```

Figure 2: Person subclasses.

The class Media was split into five subclasses: the subclasses Kids, Teens and Adults, which group media destined for kids, teens and adults, respectively; and the subclasses Movie, which groups media of type movie, and TVShow, which groups media of type tv show (Figure 3).

```
sub:TVShow rdf:type rdfs:Class ;
  rdfs:subClassOf schema:Media .

sub:Movie rdf:type rdfs:Class ;
  rdfs:subClassOf schema:Media .

sub:Adults rdf:type rdfs:Class;
  rdfs:subClassOf schema:Media .

sub:Teens rdf:type rdfs:Class;
  rdfs:subClassOf schema:Media .

sub:Kids rdf:type rdfs:Class;
  rdfs:subClassOf schema:Media .
```

Figure 3: Media subclasses.

The subclasses Movie and TVShow were further divided into four subclasses each: the OldMovie and RecentMovie, which group media based on their release year, associating movies release prior to 2000 to the OldMovie subclass and those release on or after 2000 to the subclass RecentMovie; the other two sub-subclasses are LongMovie and ShortMovie,

which group media of type movie based on its duration, grouping those with a duration of 90 minutes or less in the subclass ShortMovie and the rest to the subclass LongMovie (Figure 4). The subclasses OldTVShow, RecentTVShow, LongTVShow and ShortTVShow are the equivalent the previously described classes, with all the same reasonings, but dedicated to group the media of type tv show.

```
# Sub Sub-Classes
sub:OldMovie rdf:type rdfs:Class;
|   rdfs:subClassOf sub:Movie .

sub:RecentMovie rdf:type rdfs:Class;
|   rdfs:subClassOf sub:Movie .

sub:LongMovie rdf:type rdfs:Class;
|   rdfs:subClassOf sub:Movie .

sub:ShortMovie rdf:type rdfs:Class;
|   rdfs:subClassOf sub:Movie .

sub:OldTVShow rdf:type rdfs:Class;
|   rdfs:subClassOf sub:TVShow .

sub:RecentTVShow rdf:type rdfs:Class;
|   rdfs:subClassOf sub:TVShow .

sub:LongTVShow rdf:type rdfs:Class;
|   rdfs:subClassOf sub:TVShow .

sub:ShortTVShow rdf:type rdfs:Class;
|   rdfs:subClassOf sub:TVShow .
```

Figure 4: Subclasses of TVshow and Movie classes.

The properties used were also defined (Figure 5), where some constraints were established through the definition of the domains and ranges of the properties. This is clear in the case of the property “cast”, where it was defined that the domain was an instance of the class Media, and all media has to have this property, and the range was an instance of the subclass Actor, because this property can only be used to represent actors that participated in said media. The same logic was followed for the property “director”. The rest of the properties were only restricted on their domains, to the class Media, as their range are Literals.

```
pred:rating rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range xsd:string .  
  
pred:real_name rdf:type rdf:Property ;  
  rdfs:range xsd:string .  
  
pred:director rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range sub:Director .  
  
pred:title rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range xsd:string .  
  
pred:cast rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range sub:Actor .  
  
pred:type rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range xsd:string .  
  
pred:date_added rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range xsd:date .  
  
pred:duration rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range xsd:duration .  
  
pred:release_year rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range xsd:gYear .  
  
pred:listed_in rdf:type rdf:Property ;  
  rdfs:domain schema:Media ;  
  rdfs:range xsd:string .
```

Figure 5: Property definition.

Having defined this, the final structure defined by these ontologies is as follows, inferred by Protégé, as shown in Figure 6.

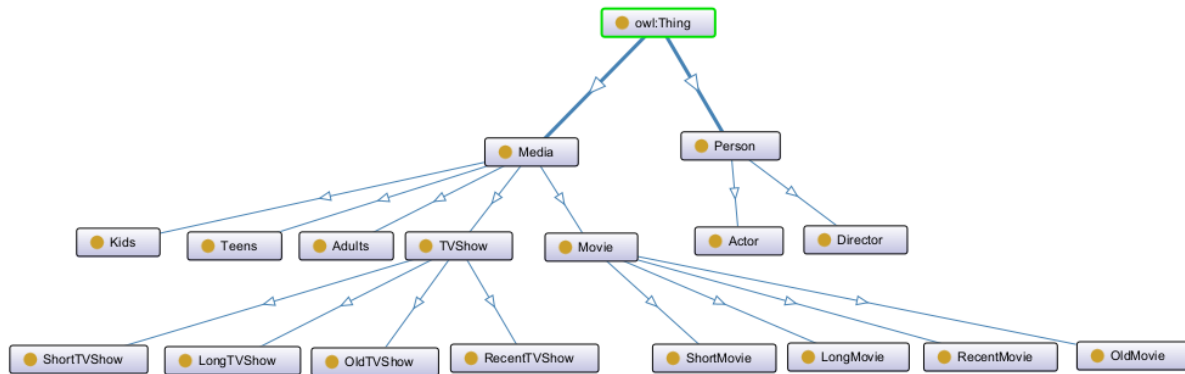


Figure 6: Structure visualization on Protégé.

Inferences

Even though the ontologies were defined, the reasoners could not make all the inferences necessary to associate the data to its corresponding groups. The only inferences made were the ones that associated the instances to the subclasses Actor and Director.

So, in order to perform the rest of the inferences, several SPIN rules were made, using SPARQL and GraphDB. The simple rule to make the inference for the subclass Movie is as follows (Figure 7).

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sub: <http://ws.org/netflix_info/sub/>
PREFIX schema: <https://schema.org/>
PREFIX pred: <http://ws.org/netflix_info/pred/>

INSERT {
  ?media a sub:Movie .
} WHERE {
  ?media a schema:Media .
  ?media pred:type ?type_id .
  ?type_id pred:real_name "Movie" .
}
  
```

Figure 7: SPIN rule - subclass Movie.

A very similar rule was defined for the subclass TVShow, with the adequate adaptations.

For the rest of the subclasses, which data was not inferred to, more complex SPIN rules were defined, as these had to depend on specific attributes, which have been described previously. For example, for the subclass Kids, which is based on the value of the property “rating”, the rule is as presented on Figure 9, which is very similar to the classes Teens and Adults.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sub: <http://ws.org/netflix_info/sub/>
PREFIX schema: <https://schema.org/>
PREFIX pred: <http://ws.org/netflix_info/pred/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

INSERT {
  ?movie a sub:Kids .
} WHERE {
  ?movie a schema:Media .
  ?movie pred:rating ?rating_id .
  ?rating_id pred:real_name ?rating .
  FILTER(?rating = "TV-Y" || ?rating = "TV-G" || ?rating = "TV-PG" || ?rating = "G")
}

```

Figure 9: SPIN rules - subclass Kids.

An example for class OldMovie in on Figure 8, which is similar to RecentMovie, OldTVShow and RecentTVShow, with their appropriate changes:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sub: <http://ws.org/netflix_info/sub/>
PREFIX schema: <https://schema.org/>
PREFIX pred: <http://ws.org/netflix_info/pred/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

INSERT {
  ?movie a sub:OldMovie .
} WHERE {
  ?movie a sub:Movie .
  ?movie pred:release_year ?release_year .
  ?release_year pred:real_name ?release .

  FILTER(xsd:integer(?release) < 2000)
}

```

Figure 8: SPIN rules - subclass OldMovie.

Example for class LongMovie, which is similar to ShortMovie, LongTVShow and ShortVShow, with their appropriate changes (Figure 10).

Doing these SPIN rules solved our problem, but ended up creating a new problem because these have to be executed manually in the database which is not practical and user-

friendly. So we had to come up with a way to execute these SPIN rules in a programmatically way by performing a simple action in the web application, which will be described ahead.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sub: <http://ws.org/netflix_info/sub/>
PREFIX schema: <https://schema.org/>
PREFIX pred: <http://ws.org/netflix_info/pred/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

INSERT {
  ?movie a sub:LongMovie .
} WHERE {
  ?movie a sub:Movie .
  ?movie pred:duration ?duration_id .
  ?duration_id pred:real_name ?duration .

  FILTER(xsd:integer(REPLACE(?duration, "[^0-9]", "")) > 90)
}
```

Figure 10: SPIN rules - subclass LongMovie.

Use and integration of data from DBpedia

To obtain comprehensive details on actors or directors, we used the *actors_directors_info* function. DBpedia is a structured data project that compiles information from Wikipedia. Using the robust query language for RDF (Resource Description Framework) data, SPARQL, this function retrieves pertinent information about an actor/director, including their biography, birthplace, date and time of birth, a photo, and their occupations.

Formatting the Name:

The input *person_name* (e.g., "Margot Robbie") is prepared in accordance with the DBpedia resource naming convention by substituting underscores for spaces.

```
formatted_name = person_name.replace(" ", "_")
person_uri = f"http://dbpedia.org/resource/{formatted_name}"
```

Figure 11: DBpedia - Name formatting.

SPARQL Query

A SPARQL query (Figure 12) is constructed to fetch various details about the person. The query includes:

- **abstract:** A brief biography.

- **birthDate:** The birth date.
- **birthPlace:** The birth place.
- **thumbnail:** A URL to the person's image.
- **occupations:** The person's occupations..

The `GROUP_CONCAT` function is used to combine multiple occupation values into a single string, separated by commas.

```
query = f"""
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?abstract ?birthDate ?birthPlace ?thumbnail (GROUP_CONCAT(?occupation; separator=", ") AS ?occupations) WHERE {{
  <{person_uri}> dbo:abstract ?abstract .
  OPTIONAL {{ <{person_uri}> dbp:birthDate ?birthDate . }}
  OPTIONAL {{ <{person_uri}> dbp:birthPlace ?birthPlace . }}
  OPTIONAL {{ <{person_uri}> dbo:thumbnail ?thumbnail . }}
  OPTIONAL {{ <{person_uri}> dbp:occupation ?occupation . }}
  FILTER (lang(?abstract) = 'en')
}}
LIMIT 1
"""
```

Figure 12: DBPedia - SPARQL Query.

Sending and Processing the response:

The SPARQL query is sent to the DBpedia endpoint ('http://dbpedia.org/sparql') and a JSON format response is retrieved.

The necessary data is extracted from the response through processing. The function parses the JSON data if the response is successful (status code 200) and contains results.

Margot Robbie
Occupations: producer | Actress

Biography: Margot Elise Robbie (/ˈmɑːrɡoʊ ˈroʊbi/; born 2 July 1990) is an Australian actress and producer. Known for her work in both blockbusters and independent films, she has received several accolades, including nominations for two Academy Awards, three Golden Globe Awards, and five British Academy Film Awards. Time magazine named her one of the 100 most influential people in the world in 2017 and she was ranked as one of the world's highest-paid actresses by Forbes in 2019. Born and raised in Queensland, Robbie began her career in 2008 on the television series *Neighbours*, on which she was a regular until 2011. After moving to America, she led the television series *Pan Am* (2011–2012), and had her breakthrough in 2013 with the black comedy film *The Wolf of Wall Street*. She achieved wider recognition with starring roles as Jane Porter in *The Legend of Tarzan* (2016) and Harley Quinn in the DC superhero films *Suicide Squad* (2016), *Birds of Prey* (2020) and *The Suicide Squad* (2021). Robbie received critical acclaim and a nomination for the Academy Award for Best Actress for her portrayal of disgraced figure skater Tonya Harding in the biopic *I, Tonya* (2017). This acclaim continued with her roles as Queen Elizabeth I in the period drama *Mary Queen of Scots* (2018), Sharon Tate in the comedy-drama *Once Upon a Time in Hollywood* (2019), and a fictional Fox News employee in the drama *Bombshell* (2019); she received BAFTA Award nominations for all three and a nomination for the Academy Award for Best Supporting Actress for the lattermost. Robbie is married to filmmaker Tom Ackerley. They are co-founders of the production company LuckyChap Entertainment, under which they have produced several films, including *I, Tonya* and *Promising Young Woman* (2020), as well as the television series *Dollface* (2019–2022) and the miniseries *Maid* (2021).

Birth Date: 1990-07-02
Birth Place: Dalby, Queensland, Australia

Movies:

Suite Française	About Time	Peter Rabbit
Release: 2014	Release: 2013	Release: 2018
Description: Waiting for news of her prisoner-of-war husband, a French woman in Nazi-occupied France reluctantly falls for a German officer quartering in her home.	Description: When Tim learns that the men in his family can travel in time and change their own lives, he decides to go back and win the woman of his dreams.	Description: In this mix of live action and CGI, rascally Peter Rabbit faces a tough new foe in his ongoing battle for the veggies in Mr. McGregor's garden.
View More	View More	View More

Figure 13: Demonstration - Searching for the Actress 'Margot Robbie' in our Homepage.

Publishing Data

In order to comply with the semantic web purposes, the way we publish data on the web can make its contribution. There are currently several standards with the purpose of giving the machines the ability to figure out at some degree about the knowledge that humans publish on the web. Within these base rules are microformats and RDFa. Both allow the insertion of some metadata on HTML documents which helps machines to perform knowledge extraction from them.

On our application, we experimented with the presented standards when publishing actors information, explored previously from DBpedia, and tv shows data, while exploring the application base dataset.

Data and publishing methods

Since we desired to use both methods on this application, a decision was required to establish the best fitting relationship between the data and the standards in use.

First and foremost, we started by pointing out the data being published. As seen previously, while exploring DBpedia, the data published related with the actors comprises the actor's name, birthday, birthplace, photo, and also its occupations and biography. For the movies/tv shows, as it is conceptually a different class, a different set of data that characterizes it is published. On Figure 14 and Figure 15 are described the data published alongside the methods or metadata associated with its publication.

Actors Data Published		Microformat	property
Name		h-card	p-name
Birth date		h-card	dt-bday
Birthplace		h-adr	p-country-name
Photograph		h-card	u-photo
Biography		h-card	p-note
Occupation		h-card	p-role

Figure 14: Actors data and Microformat description.

Movie / TV show Data Published		RDFa Vocab	https://schema.org/
Title		Typeof: Movie	https://schema.org/Movie
Director			name
Cast			director
Country			actor
Release year			countryOfOrigin
Rating			datePublished
Duration			contentRating
Genres			duration
Description			genre
			description

Figure 15: Movie/TV show data and RDFa description.

For the actors data, microformats standards were the selected option for the publishing purposes. Microformats are translated in different formats in accordance with the related data. "For publishing people and organizations on the web"[1] h-card is often used and was our main format here.

Although data like the actor's name, birthday and photo were easily related with the h-card standard properties, other fields like biography and, in particular, birthplace were not. For the sake of simplicity, and not existing a directly related property on the standard, the "p-note" was chosen to relate the actor to its biography, since this property is attributed to "additional

notes” [1]. With regards to the birthplace, we opted to embed another format, h-adr, which is the “open format for publishing locations”. On h-card there are existing properties related to addresses as it is in h-adr however, on our point of view, those seem to be more related to the person or organization’s address. Taking this into account, to somehow distinguish the birthplace, the “p-country-name” from the embedded h-adr standard was used.

Is important to note here that, to more accurately define the birthplace data, we also considered other possibilities like the usage of experimental extensions [2] of the vocabulary. This is the method used to “experiment” new vocabulary into the microformat and potentially evolve it by expanding.

Concerning the movies/tv shows data, as seen in Figure 15, the *Resource Description Framework in Attributes* (RDFa) was used instead of microformats for two main reasons. The first one, because one of the objectives of this applications is experimenting with this two standards, and the second, because this format provides a more specific definition of attributes making it easier to apply to the particularities of the data being published.

Instead of defining a brand new vocabulary we opted for finding a robust vocabulary source. With this purpose we consulted Gemini (view annexes for more details - AI consulting for RDFa vocabulary), the AI tool from google, which made us converge to Schema.org, a rich and versatile vocabulary source, “founded by Google, Microsoft, Yahoo and Yandex” [3].

Within this vocabulary source, a “Movie” [4] type is defined with a large set of related properties. From this type we characterized all the published data. For the sake of simplicity and due to the similarity no other type like “TVSeries” [<https://www.schema.org/TVSeries>] were considered.

Publishing actors’ information (using microformats)

The code implementation can be consulted on the home.html document on the application. Examples of the results of the implementation and parsing validation can be consulted in annexes, on section Publishing data. The examples are result of the search by “Will Smith”, on the search field of the main page of the application.

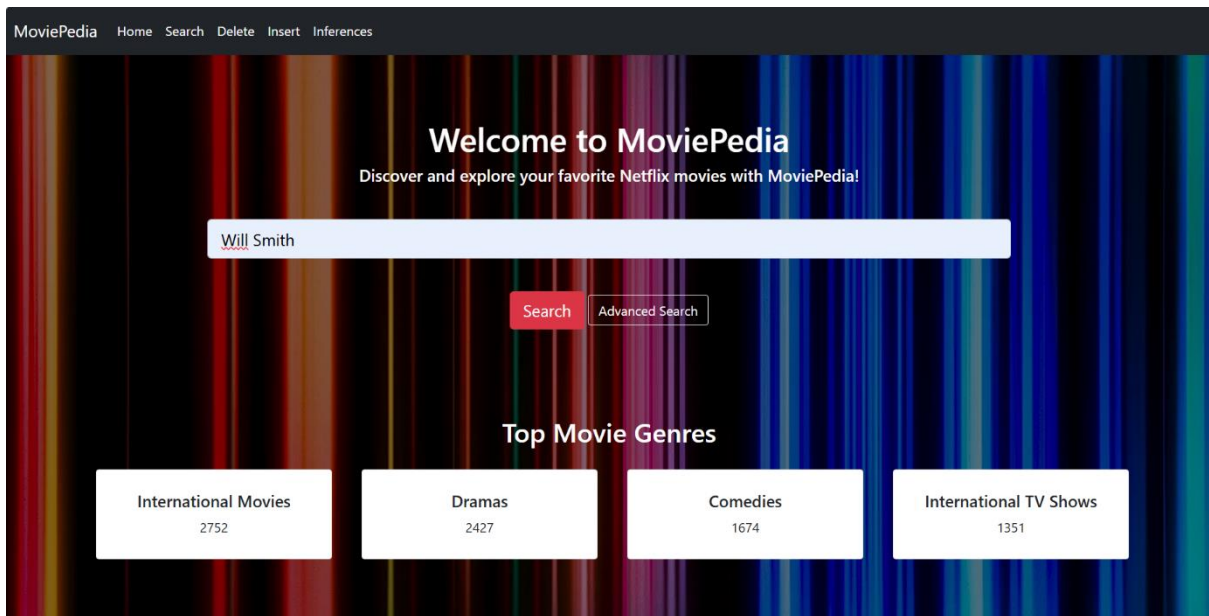


Figure 16: Demonstration of search performed for testing and validation of Microformats data publication. (view annexes for results)

The implementation of the microformats were validated with the Microformats Parser (PHP) v0.5.0 (<https://pin13.net/mf2/>).

Publishing Movie/tv shows data (using RDFa)

The code implementation can be consulted on the film_card.html document that is included in the rendering of the search.html page on the application. Examples of the results of the implementation and parsing validation can be consulted in annexes, on section Publishing data. The examples are result of the search “Men in Black II”, on the “search by a specific movie” field of the advanced search page of the application.

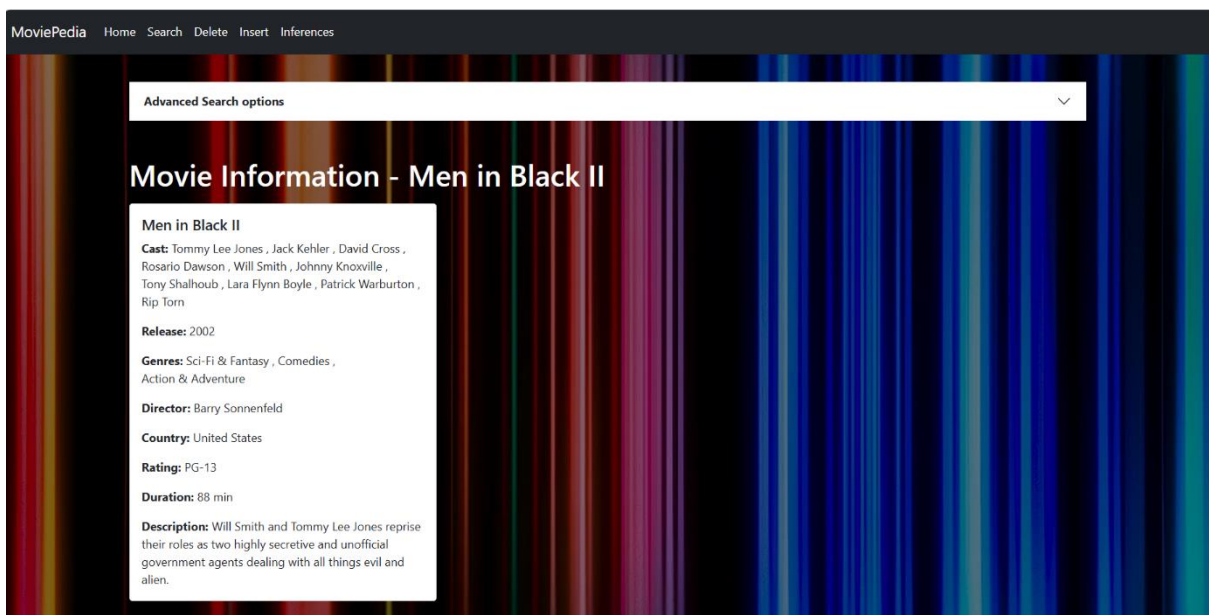


Figure 17: Demonstration of search performed for testing and validation of RDFa data publication. (view annexes for results)

The implementation of the RDFa format was validated with the tool provided on <https://rdfa.info/play/>. A short overview is provided on the following image.



Figure 18: Visualization of RDFa validation.

Functionalities

We extended the previous application functionalities, in order to integrate the topics discussed previously.

We started by extending the functionalities of the Home search bar, by giving it the ability to query DBpedia, when searching by instances of the class Person, returning information about the person searched, such as their occupation, bibliography, birth date and place.

Will Smith

Occupations: Actor | rapper | film producer

Biography: Willard Carroll Smith II (born September 25, 1968), also known by his stage name The Fresh Prince, is an American actor and rapper. He began his career starring as a fictionalized version of himself on the NBC sitcom The Fresh Prince of Bel-Air (1990–1996). Noted for his work in film, television, and music, Smith has received multiple accolades, including an Academy Award and four Grammy Awards. As of 2022, his films have grossed over \$9.3 billion globally, making him one of Hollywood's most bankable stars. Smith first gained recognition as part of a hip hop duo with DJ Jazzy Jeff, with whom he released five studio albums and the US Billboard Hot 100 top 20 singles "Parents Just Don't Understand", "A Nightmare on My Street", "Summertime", "Ring My Bell", and "Boom! Shake the Room" from 1984 to 1994. He released the solo albums Big Willie Style (1997), Willennium (1999), Born to Reign (2002), and Lost and Found (2005), which contained the US number-one singles "Gettin' Jiggy wit It" and "Wild Wild West". He has received four Grammy Awards for his rap performances. Smith achieved wider fame as a leading man in films such as the action film Bad Boys (1995), its sequels Bad Boys II (2003) and Bad Boys for Life (2020), and the sci-fi comedies Men in Black (1997), Men in Black II (2002), and Men in Black 3 (2012). After starring in the thrillers Independence Day (1996) and Enemy of the State (1998), he received Academy Award for Best Actor nominations for his portrayal as Muhammad Ali in Ali (2001), and as Chris Gardner in The Pursuit of Happyness (2006). He then starred in a range of commercially successful films, including I, Robot (2004), Shark Tale (2004), Hitch (2005), I Am Legend (2007), Hancock (2008), Seven Pounds (2008), Suicide Squad (2016) and Aladdin (2019). For his portrayal of Richard Williams in the biographical sports drama King Richard (2021), Smith won the Academy Award, BAFTA Award, Golden Globe Award, and Screen Actors Guild Award for Best Actor. At the 2022 Academy Awards ceremony, shortly before winning, Smith faced public backlash for slapping and shouting at Oscar presenter Chris Rock after Rock made a joke referencing Smith's wife, Jada Pinkett Smith. Smith subsequently resigned from the Academy and was banned from attending all Academy functions, including the Oscars, for ten years.

Birth Date: 1968-09-25

Birth Place: Philadelphia, Pennsylvania, U.S.




Figure 19: Functionalities – actor/director info.

We then added a new page dedicated to query Inferences.

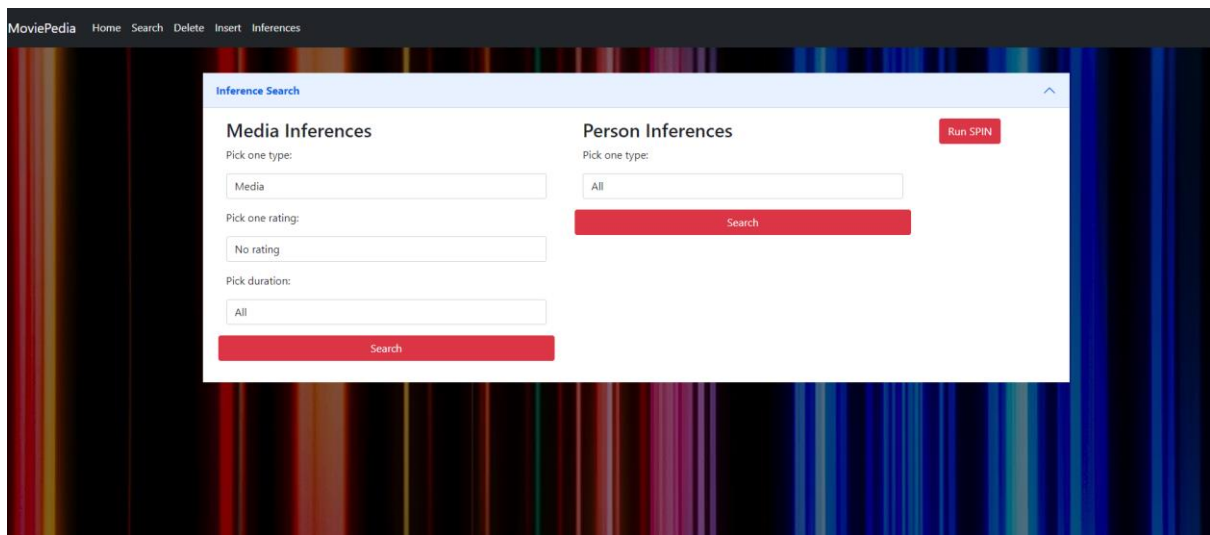


Figure 20: Functionalities – Query Inferences.

In this page, we can perform several search combinations, between the different classes established by the ontologies.

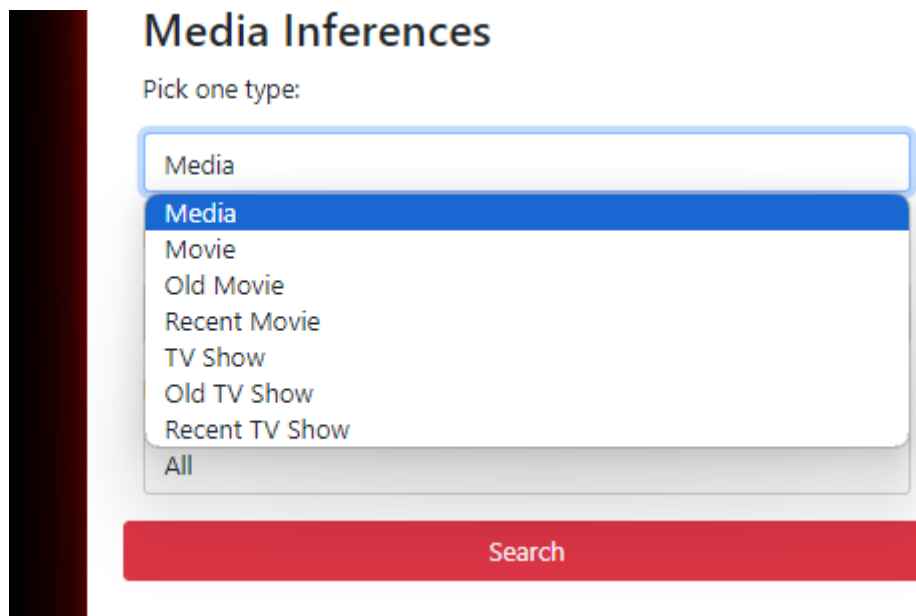


Figure 21: Functionalities – Query Inferences.

Inference Search

Media Inferences

Pick one type:

Media

Pick one rating:

No rating

No rating

For kids

For teens

For adults

Search

Figure 22: Functionalities – Query Inferences.

Inference Search

Media Inferences

Pick one type:

Media

Pick one rating:

No rating

Pick duration:

All

All

Movies

Long duration (>90m)

Short duration (<=90m)

TV Shows

Long duration (>= 2 seasons)

Short duration (1 season)

Figure 23: Functionalities – Query Inferences.

By selecting the values of these fields and performing the search, we can test and obtain results from the inference data. However, this can only be done after executing the SPIN rules, and in order to not have to do that manually, we added a button to this page that, when clicked, will execute every SPIN rule necessary to infer the data.

Run SPIN

Figure 24: Functionalities – SPIN Rules.

Conclusion

By implementing the integrations previously described, we successfully extended the features of our application. By using microformats and RDFa, we managed to make our data machine readable, enhancing interoperability. Also the use of ontologies allowed to formulate a useful categorization of the data, improving the querying of the data. Additionally, by integrating DBpedia querying, we also improved the usability of our application, by adding extra useful information for the user.

Overall, we managed to demonstrate the usefulness of semantic web technologies in enhancing data accessibility and usability in the web.

References

- [1] “h-card,” [Online]. Available: <https://microformats.org/wiki/h-card>. [Acedido em Maio 2024].
- [2] “microformats2,” [Online]. Available: <https://microformats.org/wiki/microformats2#extensions>. [Acedido em Maio 2024].
- [3] “Welcome to Schema.org,” [Online]. Available: <https://www.schema.org/>. [Acedido em Junho 2024].
- [4] “Movie,” [Online]. Available: <https://www.schema.org/Movie>. [Acedido em Junho 2024].

Annexes

Instructions to application execution

Initial Setup

1. **Create a virtual environment** - In the terminal, run the following commands:

```
python3 -m venv .venv
```

```
source .venv/bin/activate
```

2. **Install requirements** - In the terminal, use the following command to install the dependencies listed in the requirements.txt file:

```
pip install -r requirements.txt
```

GraphDB Database

1. **Download GraphDB:** Download GraphDB from <https://graphdb.ontotext.com/>.
2. **Create a repository:** In the GraphDB dashboard, go to "Configuration -> Repositories -> Create New Repository" and create a new repository, naming it "MoviePedia".
3. **Import data:** In the GraphDB dashboard, go to "Import -> Load RDF Files" and upload the netfli_titles.nt file, located in the "Data" folder, then click the "Import" button.

Running the Server

1. **Start the server** - In the terminal, run the following command to start the server:

```
python manage.py runserver
```

AI consulting for RDFa vocabulary

Using Gemini, from google, on may 2024:

Prompt:

"Hi! im currentlry developing a web semantic based project that consists on a web page to explore a netflix film catalog. I am trying to chose the best fitting rdfa vocabulary to publish films data on the html page. what do you suggest?"

Response:

"There are several RDFa vocabularies suitable for publishing film data on your web page. Here are some top options:

1. Schema.org:

- Pros:
 - Widely adopted and supported by major search engines like Google, Bing, and Yandex.
 - Comprehensive vocabulary covering various aspects of films, including title, director, actors, release date, genre, rating, and more.
 - Integrates well with other Schema.org vocabularies for related entities like people and organizations.
- Cons:
 - Can be slightly verbose and complex for simple use cases.

2. Open Graph Protocol (OGP):

- Pros:
 - Specifically designed for social media sharing, making it ideal for promoting your film catalog on platforms like Facebook and Twitter.
 - Provides essential information like title, description, image, and URL for rich previews.
 - Easier to implement compared to Schema.org.
- Cons:
 - Limited scope compared to Schema.org, focusing primarily on social media sharing.

3. Dublin Core (DC):

- Pros:
 - Widely used standard for metadata across various domains, including film.
 - Offers basic elements like title, creator, description, and date.
 - Can be combined with other vocabularies for more detailed information.
- Cons:
 - Less specific to film data compared to Schema.org or OGP.

Choosing the right vocabulary:

The best choice depends on your specific needs and priorities:

- For broad search engine visibility and rich data representation, Schema.org is the preferred option.
- If your primary focus is social media sharing, OGP is a good choice.

- For basic metadata and compatibility with other systems, consider DC.

You can also combine these vocabularies to leverage their strengths. For example, use Schema.org for comprehensive film data and OGP for social media previews.

Here are some additional resources that might be helpful:

- **Schema.org Film:** <https://schema.org/Movie>
- **Open Graph Protocol:** <https://www.popwebdesign.net/what-is-ogp.html>
- **Dublin Core:** <https://dublincore.org/>

I hope this helps!"

Publishing data

Examples using micro formats

Example 1: using h-adr for birthplace

HTML snippet published:

```
<div class="h-card">
  <h2 class="text-white">
    <span class="p-name">Will Smith</span>
  </h2>

  <div class="actor-info">
    <p style="color: white;">
      <strong>Occupations:</strong>

      <span class="p-role"> Actor</span>
      |

      <span class="p-role"> rapper</span>
      |

      <span class="p-role"> film producer</span>

    </p>
    <p style="color: white; font-size:medium">
      <strong>Biography:</strong>
      <span class="p-note">Willard Carroll Smith II (born September 25, 1968),
[...], including the Oscars, for ten years.</span>
    </p>
    <p style="color: white;">
      <strong>Birth Date:</strong>
      <span class="dt-bday">1968-09-25</span>
    </p>
```

```

        <!-- <p style="color: white;"><strong>Birth Place:</strong><a
rel="bdayplace" href="http://ws.org//Will Smith//birthPlace">Philadelphia, Pennsylvania,
U.S.</a></p> -->

        <!-- <p style="color: white;">
        <strong>Birth Place: </strong>
        <span class="p-x-bdayplace">Philadelphia, Pennsylvania, U.S.</span>
        </p> -->
        <p style="color: white;">
        <strong>Birth Place:</strong>
        <span class="bdayplace">
        <span class="h-adr p-country-name">Philadelphia, Pennsylvania,
U.S.</span>
        </span>
        </p>
        <p></p>
        </div>

</div>

```

Parsing result (using <https://pin13.net/mf2>) in JSON:

```

{
  "items": [
    {
      "type": [
        "h-card"
      ],
      "properties": {
        "name": [
          "Will Smith"
        ],

```



```

    "role": [
      "Actor",
      "rapper",
      "film producer"
    ],
    "note": [
      "Willard Carroll Smith II (born September 25, 1968), [...], including the
Oscars, for ten years."
    ],
    "photo": [
      {
        "value":
"http://commons.wikimedia.org/wiki/Special:FilePath/TechCrunch_Disrupt_2019_(4883443
4641)_(cropped).jpg?width=300",
        "alt": "Will Smith"
      }
    ],
    "bday": [
      "1968-09-25"
    ],
    "country-name": [
      {
        "type": [
          "h-adr"
        ],
        "properties": {
          "name": [
            "Philadelphia, Pennsylvania, U.S."
          ]
        },
        "value": "Philadelphia, Pennsylvania, U.S."
      }
    ]
  }

```

```

    }
  ],
  "rels": {},
  "rel-urls": {},
  "debug": {
    "package": "https://packagist.org/packages/mf2/mf2",
    "source": "https://github.com/indieweb/php-mf2",
    "version": "v0.5.0",
    "note": [
      "This output was generated from the php-mf2 library available at",
      "https://github.com/indieweb/php-mf2",
      "Please file any issues with the parser at https://github.com/indieweb/php-",
      "mf2/issues",
      "Using the Masterminds HTML5 parser"
    ]
  }
}

```

Example 2: using experimental extensions

HTML snippet published:

```

<div class="h-card">
  <h2 class="text-white">
    <span class="p-name">Will Smith</span>
  </h2>

  <div class="actor-info">
    <p style="color: white;">
      <strong>Occupations:</strong>

    <span class="p-role"> Actor</span>
    |
  </div>

```

```

    <span class="p-role"> rapper</span>
    |

    <span class="p-role"> film producer</span>

</p>

<p style="color: white; font-size:medium">
<strong>Biography:</strong>
    <span class="p-note">Willard Carroll Smith II (born September 25, 1968),
[...], including the Oscars, for ten years.</span>
</p>
<p style="color: white;">
<strong>Birth Date:</strong>
<span class="dt-bday">1968-09-25</span>
</p>
<!-- <p style="color: white;"><strong>Birth Place:</strong><a
rel="bdayplace" href="http://ws.org//Will Smith//birthPlace">Philadelphia, Pennsylvania,
U.S.</a></p> -->
<p style="color: white;">
<strong>Birth Place: </strong>
<span class="p-x-bdayplace">Philadelphia, Pennsylvania, U.S.</span>
</p>
<!-- <p style="color: white;">
<strong>Birth Place:</strong>
<span class="bdayplace">
<span class="h-adr p-country-name">Philadelphia, Pennsylvania,
U.S.</span>
</span>
</p> -->
<p></p>
</div>

```

</div>

Parsing result (using <https://pin13.net/mf2>) in JSON:

```
{
  "items": [
    {
      "type": [
        "h-card"
      ],
      "properties": {
        "name": [
          "Will Smith"
        ],
        "role": [
          "Actor",
          "rapper",
          "film producer"
        ],
        "note": [
          "Willard Carroll Smith II (born September 25, 1968), [...], including the Oscars, for ten years."
        ],
        "x-bdayplace": [
          "Philadelphia, Pennsylvania, U.S."
        ],
        "photo": [
          {
            "value":
"http://commons.wikimedia.org/wiki/Special:FilePath/TechCrunch_Disrupt_2019_(48834434641)_(cropped).jpg?width=300",
            "alt": "Will Smith"
          }
        ],
      ]
    }
  ]
}
```

26

```

        "bday": [
            "1968-09-25"
        ]
    }
}

],
"rels": {},
"rel-urls": {},
"debug": {
    "package": "https://packagist.org/packages/mf2/mf2",
    "source": "https://github.com/indieweb/php-mf2",
    "version": "v0.5.0",
    "note": [
        "This output was generated from the php-mf2 library available at",
        "https://github.com/indieweb/php-mf2",
        "Please file any issues with the parser at https://github.com/indieweb/php-",
        "mf2/issues",
        "Using the Masterminds HTML5 parser"
    ]
}
}

```

Examples using RDFa

HTML snippet published:

```

<div class="card-body" vocab="https://schema.org/" typeof="Movie">
  <!-- title -->
  <h5 class="card-title" property="name">Men in Black II</h5>
  <!-- cast -->
  <form class="mb-3" action="/search_alternative/" method="post">
    <p class="card-text"><strong>Cast:</strong>
      <input
        type="hidden"
        name="csrfmiddlewaretoken"
        value="cbmBf4AxyxVCBdN2wNLXICobUejayeV0MrFihZSynazrtRwHQMfIdMrgblO6HuNW"
      >

```

```

        <button    type="submit"    class="button_text"    name="cast_search"
value="Tommy Lee Jones">
        <span property="actor" typeof="person">
        <span property="name">Tommy Lee Jones</span>
        </span>
        </button>
        ,

        <button type="submit" class="button_text" name="cast_search" value="Jack
Kehler">
        <span property="actor" typeof="person">
        <span property="name">Jack Kehler</span>
        </span>
        </button>
        ,

        <button    type="submit"    class="button_text"    name="cast_search"
value="David Cross">
        <span property="actor" typeof="person">
        <span property="name">David Cross</span>
        </span>
        </button>
        ,

        <button    type="submit"    class="button_text"    name="cast_search"
value="Rosario Dawson">
        <span property="actor" typeof="person">
        <span property="name">Rosario Dawson</span>
        </span>
        </button>
        ,

        <button type="submit" class="button_text" name="cast_search" value="Will
Smith">

```

```

    <span property="actor" typeof="person">
    <span property="name">Will Smith</span>
    </span>
    </button>
    ,

    <button    type="submit"    class="button_text"    name="cast_search"
value="Johnny Knoxville">
    <span property="actor" typeof="person">
    <span property="name">Johnny Knoxville</span>
    </span>
    </button>
    ,

    <button    type="submit"    class="button_text"    name="cast_search"
value="Tony Shalhoub">
    <span property="actor" typeof="person">
    <span property="name">Tony Shalhoub</span>
    </span>
    </button>
    ,

    <button type="submit" class="button_text" name="cast_search" value="Lara
Flynn Boyle">
    <span property="actor" typeof="person">
    <span property="name">Lara Flynn Boyle</span>
    </span>
    </button>
    ,

    <button    type="submit"    class="button_text"    name="cast_search"
value="Patrick Warburton">
    <span property="actor" typeof="person">
    <span property="name">Patrick Warburton</span>

```

```

        </span>
        </button>
        ,

        <button type="submit" class="button_text" name="cast_search" value="Rip
Torn">

        <span property="actor" typeof="person">
        <span property="name">Rip Torn</span>
        </span>
        </button>

    </p>
</form>
<!-- release year -->
<form action="/search_alternative/" class="mb-3" method="post">
    <input
        type="hidden"
        name="csrfmiddlewaretoken"
value="cbmBf4AxyxVCBdN2wNLXICobUejayeV0MrFihZSynazrtRwHQMfldMrgblO6HuNW
">
    <p class="card-text">
        <strong>Release:</strong>
        <button type="submit" class="button_text" name="date_search"
value="2002">
        <span property="datePublished">2002</span>
        </button>
    </p>
</form>
<!-- genres -->
<form class="mb-3" action="/search_alternative/" method="post">
    <input
        type="hidden"
        name="csrfmiddlewaretoken"
value="cbmBf4AxyxVCBdN2wNLXICobUejayeV0MrFihZSynazrtRwHQMfldMrgblO6HuNW
">
    <p class="card-text"><strong>Genres:</strong>

```



```

        <button    type="submit"    class="button_text"    name="genre_search"
value="Sci-Fi & Fantasy">
        <span property="genre">Sci-Fi & Fantasy</span>
    </button>
    ,

        <button    type="submit"    class="button_text"    name="genre_search"
value="Comedies">
        <span property="genre">Comedies</span>
    </button>
    ,

        <button    type="submit"    class="button_text"    name="genre_search"
value="Action & Adventure">
        <span property="genre">Action & Adventure</span>
    </button>

</p>
</form>
<!-- director -->
<form action="/search_alternative/" class="mb-3" method="post">
    <input                type="hidden"                name="csrfmiddlewaretoken"
value="cbmBf4AxyxVCBdN2wNLXICobUejayeV0MrFihZSynazrtRwHQMfldMrgblO6HuNW
">
    <p class="card-text">
        <strong>Director:</strong>
        <button    type="submit"    class="button_text"    name="director_search"
value="Barry Sonnenfeld" property="director" typeof="person">
        <span property="name">Barry Sonnenfeld</span>
    </button>

    </p>
</form>
<!-- country -->
<p class="card-text">

```

```

<strong>Country:</strong>
<span property="countryOfOrigin" typeof="country">
  <span property="name">United States</span>
</span>
</p>
<!-- Rating -->
<p class="card-text">
  <strong>Rating:</strong>
  <span property="contentRating">PG-13</span>
</p>
<!-- Duration -->
<p class="card-text">
  <strong>Duration:</strong>
  <span property="duration">88 min</span>
</p>
<!-- description -->
<p class="card-text">
  <strong>Description:</strong>
  <span property="description">Will Smith and Tommy Lee Jones reprise their roles
as two highly secretive and unofficial government agents dealing with all things evil and
alien.</span>
</p>
</div>

```

Parsing result using (<https://rdfa.info/play/>) in JSON:

```

@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<https://rdfa.info/play/>
  rdfa:usesVocabulary <https://schema.org/> .
_:1
  rdf:type <https://schema.org/Movie>;
  <https://schema.org/name> "Men in Black II";

```

```

<https://schema.org/actor> _:2;
<https://schema.org/actor> _:3;
<https://schema.org/actor> _:4;
<https://schema.org/actor> _:5;
<https://schema.org/actor> _:6;
<https://schema.org/actor> _:7;
<https://schema.org/actor> _:8;
<https://schema.org/actor> _:9;
<https://schema.org/actor> _:10;
<https://schema.org/actor> _:11;
<https://schema.org/datePublished> "2002";
<https://schema.org/genre> "Sci-Fi & Fantasy";
<https://schema.org/genre> "Comedies";
<https://schema.org/genre> "Action & Adventure";
<https://schema.org/director> _:12;
<https://schema.org/countryOfOrigin> _:13;
<https://schema.org/contentRating> "PG-13";
<https://schema.org/duration> "88 min";
<https://schema.org/description> "Will Smith and Tommy Lee Jones reprise their
roles as two highly secretive and unofficial government agents dealing with all things evil
and alien." .
_:2
  rdf:type <https://schema.org/person>;
  <https://schema.org/name> "Tommy Lee Jones" .
_:3
  rdf:type <https://schema.org/person>;
  <https://schema.org/name> "Jack Kehler" .
_:4
  rdf:type <https://schema.org/person>;
  <https://schema.org/name> "David Cross" .
_:5
  rdf:type <https://schema.org/person>;
  <https://schema.org/name> "Rosario Dawson" .

```

_:6

rdf:type <https://schema.org/person>;
<https://schema.org/name> "Will Smith" .

_:7

rdf:type <https://schema.org/person>;
<https://schema.org/name> "Johnny Knoxville" .

_:8

rdf:type <https://schema.org/person>;
<https://schema.org/name> "Tony Shalhoub" .

_:9

rdf:type <https://schema.org/person>;
<https://schema.org/name> "Lara Flynn Boyle" .

_:10

rdf:type <https://schema.org/person>;
<https://schema.org/name> "Patrick Warburton" .

_:11

rdf:type <https://schema.org/person>;
<https://schema.org/name> "Rip Torn" .

_:12

rdf:type <https://schema.org/person>;
<https://schema.org/name> "Barry Sonnenfeld" .

_:13

rdf:type <https://schema.org/country>;
<https://schema.org/name> "United States" .