Projeto

Sistemas Baseados em Conhecimento (MAC0444)

Renato Lui Geh
NUSP: 8536030

1 Suposições

Neste projeto tivemos de assumir certas suposições com relação ao *dataset*. Para construir o OWL, usamos o arcabouço `imdbpy`.

1.1 Filmes

Ao analisar os dados do dataset do IMDb, notamos que haviam vários títulos que tinham uma definição diferente de filme. Séries e miniséries de TV são consideradas pelo IMDb como filmes. Similarmente *shows* de televisão, como a apresentação dos prêmios Oscar e Emmy também entram na mesma categoria de filmes. Apesar do IMDb considerar diferentes tipos de filmes, como `tv movie` e `movie`, essa diferenciação ainda não é uma boa classificação, já que haviam exceções a regra.

Para definirmos de forma mais rigorosa o que é um filme, tivemos que supor alguns conceitos sobre filmes. Em particular, consideramos um filme um título do IMDb que segue as seguintes regras:

1. Não tem "episódios";
2. Tem um conjunto de "diretores";
3. Tem ano de lançamento;
4. Tem um conjunto de "atores";
5. É do tipo `movie`.

Com estas fortes retrições, foi possível selecionar um conjunto consistente de filmes.

Na ontologia, chamamos de `Movie` o conceito de filme. Definimos um `Movie` como uma subclasse de `Project`, conceito da ontologia `FOAF-modified` dada. Todo `Movie` possui as propriedades de dado `movieTitle` do tipo `xsd:string` e `releaseYear` do tipo `xsd:positiveInteger`. Estas propriedades representam, respectivamente, o título do filme e o ano de lançamento.

Chamaremos de `projeto` o prefixo da ontologia que criamos, e `foaf-modified` a ontologia `FOAF-modified` dada no enunciado. A relação (i.e. propriedade de objeto) `foaf-modified:maker` de um `foaf-modified:Project` foi usado para denotar todos os colaboradores (i.e. atores e diretores) de um `projeto:Movie`.

1.2 Atores e diretores

Para os conceitos `Actor` e `Director`, usamos o conceito `foaf-modified:Person` como superclasse para ambos. As propriedades de dado de ambas são `familyName`, `firstName` e `gender`, todas da ontologia `foaf-modified`. Estas representam, respectivamente, o sobrenome, primeiro nome e gênero de um `Person`. Para `gender`, estabelecemos que os possíveis valores poderiam ser "`male`", "`female`" ou "`nil`".

Ao criar o `.owl` a partir do dataset, fizemos as seguintes suposições sobre as pessoas:

1. O primeiro nome de uma pessoa no IMDb é o seu `firstName`;
2. O último nome de uma pessoa no IMDb é o seu `familyName`;
3. O gênero de uma pessoa foi definido a partir de consultas no `imdbpy`.

Com relação a suposição 3, o dataset do IMDb não especificam gênero de uma pessoa. No entanto, o `imdbpy` gera uma tabela em SQL com os gêneros de cada ator, tendo estes valores `'f'`, `'m'` ou um valor nulo quando não especificado. Escolhemos seguir a mesma convenção neste projeto.

A relação `foaf-modified:made` mapeia um `Person` em um `Movie`. Seu inverso é `foaf-modified:maker`. Criamos duas relações subclasse de `made`: `actsIn` e `directs`, mapeando, respectivamente, um ator em um filme que ele atua e um diretor em um filme que ele dirige.

Para selecionarmos a lista de pessoas iniciais (i.e. Uma Thurman, Harvey Keitel, etc.), assumimos que a pessoa de interesse é o primeiro resultado da busca de seu nome concatenado com "`(I)`". Isso garante que o homônimo escolhido é sempre o primeiro em popularidade no IMDb.

1.3 Relações

Resumimos as relações existentes na tabela abaixo:

Nome	Domínio	Imagem	Superclasse
made	Person	Movie	-
maker	Movie	Person	-
actsIn	Actor	Movie	made
directs	Director	Movie	made

Tabela 1 Relações entre conceitos. As colunas indicam o nome da relação, o domínio e imagem, e a superclasse da relação. Como `made` e `maker` têm como superclasse a raíz comum de todas as relações, omitimos e no lugar indicamos com o caractere '-'.

1.4 Instâncias

Foram escolhidas as instâncias de `Movie`, `Actor` e `Director` como descrito no enunciado. Depois de aplicadas as restrições descritas nas subseções 1.1 e 1.2, geramos o `.owl` com a sintaxe Manchester.

Cada instância de `Movie` é do formato:

```
1  Individual: projeto:movieTitle
2
3    Types:
4      projeto:Movie
5
6    Facts:
7      projeto:movieTitle "Movie Title",
8      foaf-modified:maker projeto:director1,
9      ⋮
10     foaf-modified:maker projeto:directorN,
11     foaf-modified:maker projeto:actor1,
12     ⋮
13     foaf-modified:maker projeto:actorM,
14     projeto:releaseYear Y
```

Onde `Y` é um inteiro positivo. A lista de atores e diretores não são ordenadas como no exemplo. Para uma instância de `Actor` ou `Director` temos o seguinte formato:

```
Individual: projeto:personName

  Types:
    projeto:Actor,
    projeto:Director

  Facts:
    projeto:actsIn projeto:movie1,
    ⋮
    projeto:actsIn projeto:movieN,
    projeto:directs projeto:movie1,
    ⋮
    projeto:directs projeto:movieM,
    foaf-modified:familyName "Name",
    foaf-modified:firstName "Person",
    foaf-modified:gender ["male" ou "female" ou "nil"]
```

Para todas as instâncias, o nome da instância é baseado no seu título ou nome. Usamos o formato conhecido como `camelCase` para formatar os nomes das instâncias. Para atores que não são diretores, omitimos a vírgula anterior e `projeto:Diretor` em `Types:`. Fazemos o análogo para diretores que não atuam em nenhum filme.

1.5 Unicidade

Quando definimos os nomes dos atores e diretores, consideramos que se alguma pessoa p possui mesmo `firstName` e `familyName` que outra pessoa q, então $p = q$. Isso simplifica a ontologia, tratando homônimos como a mesma pessoa. Também supomos que todo filme que títulos diferentes. Apesar de não ser verdade, esta suposição simplifica bastante a ontologia.

Uma consequência das suposições anteriores é que não precisamos usar a propriedade `DifferentFrom` do OWL, evitando que o arquivo cresça exponencialmente, já que toda junção de primeiro com último nome será diferente, assim como todo título de filme será distinto.

2 Consultas

3 Resultados