Universidade de Coimbra

Web Semântica

Relatório de Projeto

Versão 1.0

Pedro de Sousa Alves Janeiro

2012143629

pjaneiro@student.dei.uc.pt

Coimbra, 05 de janeiro de 2017

Introdução

Este projeto foi desenvolvido no âmbito da cadeira de Web Semântica, inserida no Mestrado em Engenharia Informática.

O projeto consiste numa plataforma de pesquisa de séries televisivas, usando para isso uma ontologia personalizada e conceitos de pesquisa semântica. Dadas as várias propostas existentes para o tema do projeto, este revelou-se como aquele com que o aluno se sentiria mais à vontade, por se inserir num meio com que convive diariamente.

Para popular a ontologia do projeto, foram usados dados recolhidos das APIs do Popcorn Time, Trakt.tv e OMDB. Na primeira, recolhem-se os nomes das séries mais populares, na segunda recolhem-se todos os detalhes relativos a cada uma dessas séries, incluíndo atores e equipa técnica, e na terceira recolhem-se imagens de cada série (poster).

Arquitetura do sistema

Para esta aplicação, nomeadamente na componente web, foi adotado o modelo MVP. Para tal, usou-se a framework Spark (http://sparkjava.com/) .

O modelo (Model) consiste na classe Series, que tem toda a informação relativa a uma série, e que é diretamente mapeada para um recurso da ontologia

As vistas (View) são os ficheiros web/index.ftl, web/about.ftl, web/search.ftl e web/series.ftl. A sua renderização é feita com o auxílio da biblioteca Freemarker (http://freemarker.org/). O ficheiro web/index.html corresponde à página inicial; o ficheiro web/about.ftl à página com informação sobre o projeto (incluindo dicas de utilização); o ficheiro search.ftl corresponde às várias vistas de apresentação de resultados (mediante inserção de termos de pesquisa); o ficheiro series.ftl corresponde à janela de visualização de uma série, com respetivas sugestões.

Os controladores (Controller) são representados pelas várias chamadas ao método *get* na função *main* da classe Web. Em cada uma destas chamadas, são obtidos os dados necessários da ontologia, e é definido o template a renderizar com esses dados.

Descrição da ontologia

A principal entidade deste projeto é a <series:Series>, subClass de <foaf:Project>, que representa uma das várias séries de televisão recolhidas com a API. Esta entidade tem vários atributos que a permitem caracterizar e individualizar, como o seu nome, sinopse e vários ids (IMDB, Trakt.tv, TVDB, …).

Foi também definida a entidade <series:Person>, equivalente a <foaf:Person> e que representa qualquer pessoa associada a uma série televisiva, entre elenco e equipa técnica. Para distinguir os vários cargos que uma pessoa pode ter numa série, foram criadas várias *ObjectProperties* paralelas, indicando precisamente esse cargo. De forma a estabelecer uma relação bidirecional com a <series:Series>, para cada *ObjectProperty* que relaciona a <series:Person> com a <series:Series>, existe uma *ObjectProperty* que relaciona a <series:Series> com a <series:Person>.

Outras entidades importantes definidas foram <series:Genre> e <series:Network>, esta descendente de <foaf:Organization>. Foi criada uma relação bidirecional entre cada uma destas entidades e a entidade <series:Series>, tal como se fez com a entidade <series:Person>.

Foram definidas outras entidades, e algumas *dataProperties*, que não foram usadas no produto final, como <series:Season>.

Descrição modular

Para a obtenção dos dados usados para popular a ontologia, foi usado um simples script python.

Lendo os dados (devolvidos em formato JSON) das primeiras páginas da API do PopcornTime (https://popcornwvnbg7jev.onion.to/shows/), armazenou essa informação num único ficheiro JSON, de forma a poder iterar sobre os vários dados como um só bloco e utilizar o id de cada série para outras operações.

De seguida, para cada série, são feitos dois pedidos à API do Trakt.tv (http://docs.trakt.apiary.io/), o primeiro para recolher a descrição geral da série, com o seu título, sinopse, ano de lançamento, *rating*, entre outros, e o segundo para recolher a informação relativa a todas as pessoas envolvidas no desenvolvimento da série (entre elenco e equipa técnica).

Por último, é feito um pedido à API do OMDB (http://www.omdbapi.com/), de forma a recolher o url do poster da série.

Todos estes dados são armazenados num ficheiro JSON, que será lido pela aplicação na classe Populate. Quando se usa a aplicação pela primeira vez, será esta a classe a executar, pois irá criar todo o sistema da *triple-store* e popular a ontologia com os dados recolhidos do ficheiro JSON previamente gerado.

Na classe Main, é possível correr alguns testes iniciais da pesquisa na aplicação, através da linha de comandos. São testes pré-definidos, não permitindo input do utilizador.

Na class Web, é aplicada toda a lógica de negócio da aplicação, como referido na primeira secção deste relatório (modelo MVP). Inicialmente, quando é realizado um pedido à aplicação, a frase inserida é comparada com uma série de expressões regulares, de forma a tentar identificar pedidos mais complexos por parte dos utilizadores (por exemplo, ‘crime series by BBC with Benedict Cumberbatch’). Caso não seja estabelecida uma igualdade com nenhuma das expressões regulares detetadas, é feita uma pesquisa mais profunda, comparando a *query* com:

- nomes de séries (‘Modern Family’)
- valores de *ratings* de séries (8.0)
- nomes de distribuidoras (‘Netflix’)
- nomes de géneros (‘Comedy’)
- nomes de pessoas (‘Martin Freeman’)

Os dados são apresentados ao utilizador com informação relativa à pesquisa em que os dados foram encontrados (uma pequena label a dizer ‘Person’, por exemplo). Assim, se o utilizador pesquisar ‘Lena’, e clicar no botão que diz ‘Lena Headey - Person’, será feita uma nova pesquisa, desta vez apenas com as séries que de alguma forma contam com a participação de Lena Headey.

Todas as pesquisas são feitas com o uso de SPARQL. Na sua maioria, as pesquisas são relativamente simples. Em pesquisas mais complexas, é necessário intercalar o comando ‘UNION’ com o comando ‘OPTIONAL’, e filtros de expressões regulares. Por exemplo, para encontrar séries com a participação de uma pessoa, que pode participar como ‘Actor’, ou ‘Director’, ou outro, mas que nunca irá participar com todos os cargos, o que não deve levar a uma lista de resultados vazia (isto é, não devo identificar uma pessoa apenas se tiver TODOS os cargos na série, mas sim se tiver PELO MENOS UM cargo na série).

Estrutura do código

O código é relativamente modular, com funções com objetivos diferentes separadas umas das outras. No entanto, como muito do que se faz na aplicação é pesquisa, as várias funções de pesquisa têm muito código repetido, o que poderia ter sido evitado com um melhor estudo prévio e planeamento do desenvolvimento.

Conclusões e Trabalho Futuro

Apesar de o resultado final ser bastante satisfatório, também são evidentes algumas das suas falhas.

Para começar, as pesquisas são lentas, devido ao facto de não estarem bem estruturadas (muitas operações são repetidas várias vezes inutilmente, aumentando o tempo de execução das tarefas).

O sistema de recomendação é eficaz, mas certamente haverão melhores modelos para determinação das recomendações. Também requer várias operações que podiam ter sido melhor estudadas, tornando-se mais uma vez uma operação bastante lenta.

O mecanismo de pesquisa semântica também tem falhas, e apenas interpreta construções frásicas bastante simples.

Continuando o desenvolvimento do projeto, todas as pesquisas deviam ser revistas, de forma a reaproveitar o máximo possível de linhas de código, e simplificando ao máximo as operações para diminuir o tempo de execução. Deverão também ser revistas as expressões regulares usadas na pesquisa semântica e o motor de recomendações, não no sentido de o tornar mais eficaz, mas com o intuito de o tornar mais rápido, tal como as pesquisas.