IMDb - Movie Recomendation

João Paulo Batista Ferreira, João Francisco da Silva Mateus

jpbat@student.dei.uc.pt, jfmsilva@student.dei.uc.pt

Departamento de Engenharia Informática, Universidade de Coimbra

January 21, 2015

Abstract

Neste projecto recolheu-se um conjunto vasto de informação proveniente do site "IMDb", criando depois um esquema de navegação e recomendação entre as várias páginas, nomeadamente entre séries, filmes, actores e directores. Para isto foram criados algoritmos tanto de pesquisa semântica como de recomendação.

Keywords:— imdb, filmes, series, actor, director, recomendação

1 Introduction

Filmes, séries e toda a industria existente por trás destes, é algo que existe na vida de todas as pessoas que apreciam a sétima arte. O cinema em particular é alvo de vários concursos todos anos a nível internacional, nomeadamente nos Óscares, ou no festival de Cannes. São várias as pessoas que dedicam a sua vida a apreciar todos os pequenos detalhes existentes em cada uma destas obras primas, no entanto nem todos têm o tempo, nem as condições necessárias para prestarem atenção aos mais pequenos promenores de cada filme, sendo que para isso necessitariam de rever o mesmo várias vezes.

A motivaçao que nos levou a este trabalho prende-se nesse mesmo promenor. Poder criar um sistema que mostre aos utilizadores as principais caracteristicas de cada filme, série ou actor, sem que para isso precise sequer de se lembrar do nome do mesmo filme, o ano em que este foi realizado.

2 System Architecture

Neste capítulo serão explicados em promenor os vários componentes do nosso sistema e o modo como estes interagem.

Analizando primeiramente toda a arquitectura, vemos que esta é composta por três modulos principais, sendo estes o crawler, o populador da ontologia e o website.

2.1 Data Crawler

Este foi o primeiro módulo a ser desenvolvido no projecto. A partir do site do "IMDb", foi feito o *crawl* da informação que se achou ser relevante das páginas de vários filmes, séries, e pessoas participantes nos mesmos.

Toda a informação é tratada dentro do mesmo módulo de modo a estar normalizada para tratamento nos módulos seguintes.

2.2 Ontology Populator

Este módulo foi criado já depois de toda a ontologia estar fixa, sendo que utiliza os ficheiros criados pelo *crawler* para popular a ontologia.

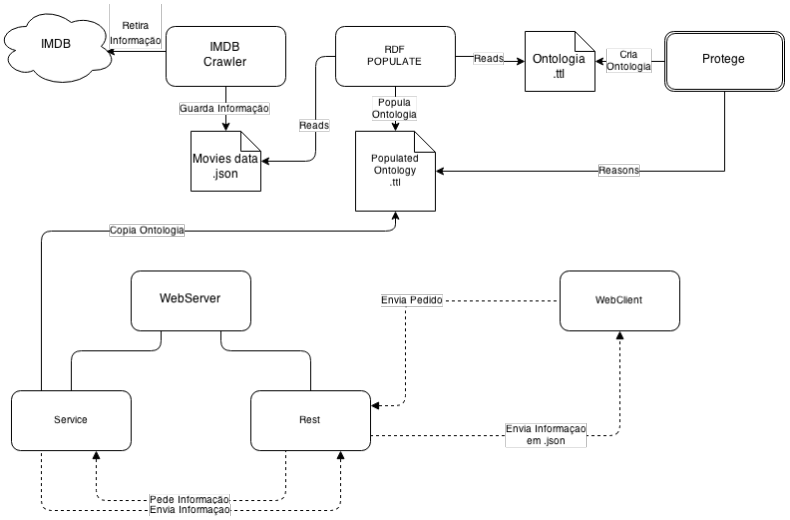

Figure 1: Diagrama da arquitectura completa.

Sobre a população feita é depois executado um *reasoner* de modo a ter a certeza que não existe qualquer tipo de inconsistência nos dados, bem como na ontologia em si.

2.3 Website

Após a obteção dos dados, e a populaçao da ontologia foi criado um *website* que mostra todas as informações ao utilizador.

Este foi desenvolvido utilizando uma arquitectura MV*, pelo que existe inerentemente a parte de *backend* e de *frontend*.

2.3.1 Backend

Nesta sub-módulo são executadas todas as operações relativas à comunicação com o *triple storage*, bem como a disponiblização dos dados para o *frontend*.

2.3.2 Frontend

Este sub-módulo é o responsável por apresentar os dados ao utilizador e comunicar com o *backend*.

3 Ontology Description

A nossa ontologia é composta por cinco classes, sendo que uma destas contém sublcasses.

3.1 Genre

Esta classe é apenas uma enumeração de todos os géneros que podem existir tanto em filmes como em séries.

3.2 Media

Esta classe é provavelmente a classe com mais importância da nossa ontologia, uma vez que é nela que estão inseridos todos os filmes e todas as séries, sendo que cada um destes é uma subclasse

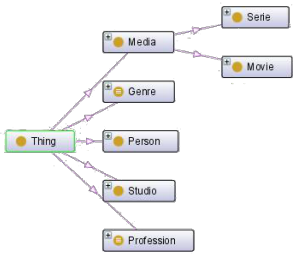

Figure 2: Diagrama da ontologia.

da classe media.
Atributos:

- hasGenre
- hasMediaClassification
- hasMediaDescription
- hasMediaDuration
- hasMediaId
- hasMediaLaunchDate
- hasMediaName
- hasMediaPoster
- hasStudio

Sendo que existem ainda atributos que apenas pretencem a uma das subclasses:
Movie:

- hasDirector

Serie:

- hasSerieStart
- hasSerieEnd
- hasSerieSeason

3.3 Person

Nesta classe são guardados todas as pessoas que participam em qualquer um dos filmes / séries que existem na ontologia, sendo que o seu papel pode variar entre director e actor.
Atributos:

- hasPersonBirth
- hasPersonBirthPlace

- hasPersonId
- hasPersonMiniBio
- hasPersonName
- hasPersonPicture
- hasProfession
- isKnownFor

3.4 Profession

Esta classe é também uma enumeração de todas as ocupações que as pessoas podem ter num determinado filme / série.

3.5 Studio

Esta última classe tem como objectivo guardar a informação disponibilizada sobre o conjunto de estúdios extraídos pelo crawler. Atributos:

- hasStudioId
- hasStudioName

4 Module Description

No capítulo anterior foram apresentados de um modo abstracto os diferentes módulos existentes no projecto, sendo que neste os mesmos serão explicados mais ao detalhe.

Todos os módulos podem ser corridos separadamente, sendo que existe uma depêndencia do módulo anterior, ou seja, o *Ontology Populator* necessita dos dados do *Data Crawler* e o website necessita da população gerada pelo *Ontology Populator*.

4.1 Data Crawler

Tal como foi referido acima este módulo foi desenvolvido em *Java*, utilizando *sreen scrapping*. Através do *package Jsoup*, foram pedidas várias páginas de filmes, sendo que em cada uma delas, através de *css selectors*, foram extraidas as informações necessárias para preencher os campos referidos na ontologia.

Tal como havia já sido supra referido, os dados são tratados de modo a que haja uma normalização dos mesmos. Para que este processo não atrasasse o *crawl* das páginas optou-se por ter um conjunto de *threads* que corriam paralelamente às *threads* que executavam o *crawler*. Por fim todos os dados são gravados localmente em ficheiro com formato *JSON*, para que pudessem facilmente ser utilizados por outra parte do sistema.

De modo a que a extracção de dados pudesse ser feita de um modo mais célere, este módulo foi criado de maneira a poder ser executado em várias máquinas diferentes simultâneamente, sem que haja um grande *overlapping* de dados.

4.2 Ontology Populator

Após terem sido já obtidos os dados do IMDb, foi criado um módulo em *Python*, que utilizando a biblioteca *rdflib* cria um grafo no qual insere todas as informações guardadas pelo *crawler*.

Neste módulo tornou-se necessário ter alguns cuidados no que diz respeito à redundância de informação, uma vez que se tornava possível o mesmo nó aparecer em ficheiros diferentes.

Após esta inserção correu-se sobre os dados o *reasoner FaCT++*, de modo a que se tivesse a certeza que não existia qualquer tipo de inconsistência de dados. Este *reasoner* foi também utilizado para criar inferências sobre os nossos dados. Estas permitem que se faça algumas ligações que não existiriam à partida na ontologia, nomeadamente as relações entre classes e subclasses.

4.3 Website

Todo o website foi desenvolvido utilizando *Backbone.js*, sendo que o serviço é disponibilizado através de um servidor *Tomcat*.

4.4 Frontend

Tal como já tinha sido referido foi utilizado um esquema MV*, já que esse é o esquema aconselhado pelos *developers* do próprio *Backbone*.

Existe um *main router* que redirecciona o utilizador para a página correcta, e informa o sistema de qual a *view* que deve estar a ser mostrada em cada momento.

Este sistema possui também templates web que são preenchidos com a informação fornecida pelo *backend*, bem como modelos para os dados que vai receber.

4.5 Backend

Esta parte do módulo foi criada utilizando *Java* para comunicar com o *triple storage*.

Este módulo possui dois submódulos bastante importantes, a parte dos serviços, resposável por toda a comunicação com o próprio sistema e a parte *REST*, que foi criada para que nós pudessemos testar a funcionalidade dos serviços.

No primeiro é criada a ligação à *triple storage*, e são também criadas as *queries* de *SPARQL* que são necessárias para responder aos pedidos quer do servidor *REST* quer do *frontend*.

No segundo foi criada uma *API REST* que permitiu testar, ao longo de todo o processo de desenvolvimento, tanto a validade dos serviços desenvolvidos, como observar quais os dados que eram devolvidos.

5 Code Structure

Em cada um dos módulos todo o código foi desenvolvido de modo a estar bem organizado.

5.1 Data Crawler

Este módulo começa por iterar sobre uma lista de anos, disponibilizada pelo imdb, na qual cada elemento é um filme lançado nesse mesmo ano. Em cada filme são guardados os ids relativos tanto ao realizador como aos actores bem como aos estúdios a que o filme pretence.

Após ser feito o *crawl* de um ano completo de filmes, é feito o *crawl* a todos os ids guardados até então.

No fim todos os dados são normalizados para que se apresentem no mesmo formato e gravados em ficheiros *JSON*.

5.2 Ontology Populator

Tal como foi acima referido este módulo foi desenvolvido em *Python*.

Este começa por fazer load de cada um dos ficheiros gerados pelo *crawler*, em cada um destes são primeiro adicionadas as pessoas ao grafo, sendo depois adicionados os estúdios e por fim os filmes e as séries.

Foi escolhida esta ordem para que quando fossem feitas as ligações no grafo não houvesse falta de qualquer nó.

6 Conclusions and Future Work

7 References