Semantic Search and Recommendation in Documents

Micael Pereira Reis
Departamento de Engenharia Informática
FCTUC
mpreis@student.dei.uc.pt
2010143871

Samuel Nunes
Departamento de Engenharia Informática
FCTUC
snunes@student.dei.uc.pt
2011158011

24 de Dezembro de 2015

Resumo

Este projeto teve como objetivo criar uma pequena aplicação web que fizesse uso dos artigos do reconhecido jornal americano *The New York Times*. Estes artigos foram extraídos e guardados de forma a ser possível construir uma aplicação em torno da Web Semântica, que desse uso à pesquisa e recomendação semântica.

Conteúdo

1 Introdução

Ao longo dos anos, a Internet tem se tornado num dos principais meios de comunicação em todo o mundo, sendo cada vez mais usada como principal fonte de informação e comunicação. Assim fica indispensável que a experiência dada ao utilizador seja aprofundada e melhorada.

Este trabalho trata-se de um projeto final para a cadeira de Web Semântica e pedia que fosse criado uma aplicação web que fizesse uso da Web Semântica. Para tal, foi necessário adquirir e aprofundar conhecimentos através das aulas da cadeira de Web Semântica, conhecimentos estes que foram fundamentais para a realização deste projeto.

A aplicação fez uso da ferramenta *Protege* e da linguagem de pesquisa *SPARQL* para pesquisar de forma eficiente artigos extraídos do jornal *The New York Times* bem como para fornecer ao utilizador recomendações de artigos que este possa achar interessante com base no seu historial de pesquisa.

Após a conclusão deste projeto é esperado ter-se ganho capacidades e conhecimento para a desenvolver um sistema de Web Semântica bem como para compreender como estes são desenvolvidos hoje em dia.

2 Arquitetura do Sistema

Sendo este um projeto relativamente complexo com diversas componentes, foram utilizadas duas linguagens de programação e várias ferramentas. As linguagens usadas foram Python e Java, e as ferramentas usadas foram o Protege, Apache Jena e o Apache Tomcat.

Estas duas linguagens desempenharam funções diferentes, com o Python a ser utilizado para a processo de extrair e lidar com a informação e o Java para desenvolver a parte Web do projeto. Efetuamos estas escolhas pois já possuíamos uma experiência considerável no uso destas linguagens para este tipo de funções, e porque estas facilitavam o uso das outras ferramentas a serem usadas.

No inicio do projeto foi necessário definir que tipo de informação e dados iria ser utilizada e como esta iria ser organizada. Para tal, foi criada uma Ontologia através da ferramenta Protege, que estrutura e cria uma ligação entre os dados como se tratasse de uma rede.

Depois de criar a Ontologia foi necessário enche-la com dados. Para isto, foi criado um pequeno programa desenvolvido em Python que faz pedidos à API do *The New York Times Top Stories* guarda-os em ficheiro JSON. Após a finalização da extração dos dados o programa começa a popular a Ontologia com os dados obtidos, de forma a obter um OWL(*Ontology Web Language*) com os dados todos organizados e ligados.

Com esta parte finalizada, passámos então á criação da interface Web. Nesta fase havia duas partes principais, a interface do utilizador e os pedidos à Ontologia.

Para efetuar pedidos à Ontologia foi usada a ferramenta Apache Jena que faz corresponder os dados com as classes e propriedades da Ontologia. Através desta ferramenta e com a ajuda da linguagem de pesquisa para RDF's, SPARQL, é possível fazer pesquisas de forma simples e eficaz à Ontologia.

Por último, para desenvolver a interface gráfica recorremos ao uso de JSP's (linguagem JAVA) que nos possibilita interligar os pedidos do utilizador aos pedidos à Ontologia. De forma a que a interface tivesse um visual mais apelativo foi usado o Skeleton que contém *Grids* e outros componentes já configurados para facilitar a implementação. Foi ainda usado o Apache Tomcat para fazer o deploy do web server.

3 Descrição da Ontologia

Uma Ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Por norma estas descrevem 4 aspetos fundamentais:

- **Indivíduos** - os objetos básicos, como por exemplo pessoas, animais, objetos concretos, etc
- **Classes** - conjuntos, coleções ou tipos de objetos, como por exemplo um grupo de pessoas;
- **Atributos** - propriedades, características que os objetos podem ter ou compartilhar;
- **Relações** - as formas como os objetos se relacionam uns com os outros.

Antes de criar a Ontologia foi necessário averiguar que dados iriam ser necessários e como estes se iriam relacionar. Para isso, primeiramente verificámos que informação a API em uso nos fornecia, e que partes desta nos iriam ser úteis. Após isto, foi então necessário pensar como estas se iriam relacionar.

Depois de termos a estrutura da Ontologia pensada e elaborada, passámos á criação da mesma. Para tal, utilizámos a ferramenta Protege, que nos ajuda a criar e a exportar a Ontologia. Nesta ferramenta foi criada uma classe e as respetivas propriedades e relações. De seguida é apresentada a arquitetura da Ontologia.

3.1 Classes

Como a única informação que tínhamos de guardar na Ontologia era a dos Artigos do *The New York Times* apenas foi necessário criar uma classe:

- **Articles** - classe que representa cada artigo e que contém todas as suas informações.

3.2 Propriedades

As propriedades da classe foram pensadas tendo em conta a informação que queríamos que fosse mostrada e disponibilizada ao utilizador. Assim, tal como referido acima, olhámos para a informação que a API nos fornecia e escolhemos para guardar na Ontologia apenas a que seria útil.

Estas contêm ainda dois atributos referentes a elas, o *Domain* que define as classes que detêm a Propriedade e o *Range* que diz respeito ao tipo da Propriedade, por exemplo String, Integer, Date, etc.

As propriedades criadas possuem todas o mesmo *Domain*, a classe *Articles*, e o seu *Range* está descrito em baixo juntamente ás propriedades:

- **Section** [String] - propriedade da classe que guarda a secção à qual o artigo se integra.
- **Subsection** [String] - propriedade da classe que guarda a subsecção à qual o artigo se integra.
- **Title** [String] - propriedade da classe que guarda o titulo do artigo.

- **Abstract** [String] - propriedade da classe que guarda o resumo sobre do artigo.
- **Author** [String] - propriedade da classe que guarda o autor do artigo.
- **URL** [String] - propriedade da classe que guarda o URL do artigo original do site do *The New York Times*.
- **Published Date** [DateTimeStamp] - propriedade da classe que guarda a data em que o artigo foi publicado.
- **Subject Description** [String] - propriedade da classe que guarda as tags sobre a descrição do assunto do artigo.
- **Organization** [String] - propriedade da classe que guarda as tags sobre as organizações que são referidas no artigo.
- **Person** [String] - propriedade da classe que guarda as tags sobre as pessoas que são mencionadas no artigo.
- **Location** [String] - propriedade da classe que guarda as tags sobre os locais que são referidos no artigo.
- **Image** [String] - propriedade da classe que guarda o URL para a imagem do artigo.

4 Módulos

Sendo este um projeto de Web Semântica, as partes mais importantes e fulcrais foram a Pesquisa Semântica e a Recomendação Semântica. Estes mecanismos fornecem ao utilizador uma melhor experiência de uso dando-lhes a possibilidade de eles efetuarem pesquisas sobre os assuntos que mais lhe interessa.

Se seguida vamos explicar como estas partes foram desenvolvidas para obtermos uma melhor entendimento sobre as mesmas.

Pesquisa Semântica

A pesquisa semântica foi possível através da linguagem de pesquisa SPARQL. Esta faz uso de *queries* para pesquisar no ficheiro OWL que contém a ontologia populada com os dados dos artigos. Um pouco parecido á forma de pesquisar da linguagem SQL.

Foram criados diversos modos de efetuar uma pesquisa, com o intuito de disponibilizar ao utilizador uma melhor experiência de uso e mais formas de este interagir com o site. Os modos de pesquisa foram os seguintes:

- Normal;
- Secção;
- Subsecção;
- Autor;
- Tags;

Para efetuarmos a pesquisa para estes diversos métodos foi criada uma função global que recebia vários parâmetros, e depois usava-os para fazer a pesquisa, tendo em conta o que era pretendido. Por exemplo se pesquisarmos uma secção, apenas o campo da secção tem uma string para pesquisa, enquanto que os outros campos de pesquisa estão vazios não afetando assim o filtro de pesquisa.

No caso da pesquisa normal, se o utilizador escreve-se uma palavra para pesquisar, a *query* tratava de pesquisar tanto no título como no resumo dos artigos á procura daqueles em que a palavra estivesse presente. Isto foi conseguido através do uso do *union* e do *regex*.

De salientar que para melhorar a pesquisa, era ignorado o facto de os caracteres estarem em maiúscula ou minúscula, os resultados eram obtidos de forma ordenada consoante a data em que foram publicados e estes eram sempre distintos para impedir que houvessem aritogs repetidos nos resultados.

Recomendação Semântica

A recomendação é obtida através do historial de secções visitadas pelo utilizador, por exemplo "Sports", "Arts", etc. Com este historial, conseguimos saber quais secções o utilizador costuma visitar mais frequentemente, possibilitando-nos dar recomendações mais precisas tendo em conta os gostos do utilizador.

Como na nossa interface apenas são apresentadas 6 artigos de cada vez, escolhemos apresentar e gerar as recomendações da seguinte forma:

1. Caso o utilizador ainda não tenha feito nenhuma pesquisa, são apresentados artigos aleatórios.

2. Se o utilizador ainda só tenha pesquisado artigos de uma secção, são apresentados 4 artigos aleatórios referentes à secção e 2 artigos aleatórios. Escolhemos este método para evitar que as recomendações sejam todas sobre a mesma secção e para mostrar uma maior variedade de artigos.

3. Quando o utilizador pesquisou sobre duas secções são apresentados 3 artigos sobre a secção mais pesquisada, 2 artigos sobre a segunda secção mais pesquisa e 1 artigo aleatório.

4. Por fim, se o utilizador já pesquisou várias secções, serão apresentados 3 artigos sobre a secção mais pesquisada, 2 artigos sobre a segunda mais pesquisada e 1 artigo sobre a terceira secção mais pesquisada. Desta forma, conseguimos apresentar uma vasta variedade de artigos que o utilizador possa ter interesse.

Escolhemos não utilizar a subsecção, o autor e as tags para as recomendações pois estas na maioria das vezes tinham apenas 1 ou 2 artigos, fazendo que a recomendação utilizando estes aspetos não fosse relevante pois não haveria mais artigos sobre os mesmos.

5 Estrutura do Programa

Como referido anteriormente, o programa foi desenvolvido em duas linguagens diferentes, Python e Java. O Python tratou de popular a Ontologia e o Java da interface e das pesquisas à Ontologia.

5.1 Python

Nesta parte do programa, era necessário extrair os dados através da API do *The New York Times* e guardar estes dados em ficheiros JSON, para depois poderem ser usados para popular-mos a Ontologia criada no Protege.

Os ficheiros necessários para efetuar esta parte do projeto podem ser encontrados na pasta "WSProject/Ontology"e está organizado da seguinte forma:

- **populate.py** - programa em Python que extrai os dados através da API, guarda-os em ficheiros JSON e depois usa-os para popular a Ontologia. Estes processos foram executados com a ajuda das livrarias

JSON e RDFLIB para Python. No final da execução é criado um ficheiro com os dados inseridos na Ontologia.

- **query.py** - programa em Python criado para testar a pesquisa na Ontologia entre outras coisas.
- **ontology.owl** - ficheiro com a Ontologia gerado pela ferramenta Protege.
- **populated.owl** - ficheiro gerado pelo programa em Python que contém a Ontologia populada com os dados.
- **Pasta "Data"** -
  - **Pasta "Parsed"** - pasta com ficheiros que contêm a informação obtida pela API de forma organizada para uma melhor leitura e compreensão dos dados obtidos.
  - **Pasta "Retrieved"** - pasta com os ficheiros JSON obtidos através da API.
  - **parser.py** - programa em Python que lê os ficheiros JSON e transforma-os em ficheiros txt com apenas os dados necessários de forma organizada.

5.2 Java

Para desenvolver esta fase do projeto foi utilizado o IDE para desenvolvimento em Java, Eclipse. O IDE permitiu-nos facilitar o uso das bibliotecas externas necessárias para o Apache Jena bem como para o Apache Tomcat.

Esta parte ficou dividida em duas pastas principais, a primeira que contém as classes para a pesquisa à Ontologia e a segunda que contém a interface web.

- **Pasta "src"**
  - **beans/SearchBean.java** - classe em Java que contém a função de pesquisa na Ontologia bem como a função que obtém as recomendações.
  - **entities/Article.java** - classe em Java dos artigos.
  - **recommendation.txt** - ficheiro com os "logs"das secções que o utilizador visita usado para criar as recomendações.

- **Pasta "WebContent"**
  - **index.jsp** - ficheiro jsp que contém a interface web do projeto.
  - **css/style.css** - ficheiro css que contém todo o css necessário para melhorar o aspeto da interface web.
  - **js/style.css** - ficheiro javacript que contém todos os scripts necessários para a interface web.

6 Conclusões e Continuidade

Hoje em dia a Web Semântica está presente num elevado número de sites, mesmo naqueles que menos esperamos. Esta é assim uma parte importante da interação do utilizador com a Internet, estando presente em sites de pesquisa, compras, notícias, etc.

Com este projeto foram adquiridos conhecimentos sobre a Web Semântica e sobre as várias partes que a integram. O conhecimento necessário para o desenvolvimento dum sistema destes também foi adquirido pois a aplicação foi desenvolvida de raiz.

Em relação á continuidade, um dos objetivos é aprofundar os conhecimentos sobre este tipo de sistemas, pois esta tecnologia está em constante crescimento e evolução, sendo que o que hoje é necessário ou obrigatório daqui a pouco tempo pode ficar banal.

7 Bibliografia

- http://jena.apache.org/
- http://www.w3.org/
- http://rdflib.readthedocs.org/
- http://www.tutorialspoint.com/
- http://www.w3schools.com/
- http://stackoverflow.com/
- http://docs.python.org
- http://docs.oracle.com

- http://developer.nytimes.com/
- http://getskeleton.com/
- http://wikipedia.org