Motor de Pesquisa Baseado na Web Semântica

Rui Gaspar, Ricardo Clemente

{ruiandre, ricjorge}@student.dei.uc.pt

Resumo: Com este projecto pretende-se desenvolver um motor de pesquisa, que implemente conceitos da Web Semântica de forma a melhorar os resultados das pesquisas.

Palavras-chave: Futebol, Pesquisa, Semântica, Ontologia

1. Introdução

Com o crescimento exponencial da informação que existe na internet, torna-se essencial que os resultados dos motores de busca sejam os que o utilizador pretende encontrar. As pesquisas baseadas em *match* de palavras-chave são muito limitadas, uma vez que obriga o utilizador a dizer ao motor de pesquisa exactamente o que pretende encontrar. Isto funciona se o utilizador souber o que encontrar, mas se o utilizador não souber, torna o processo de pesquisa muito mais ineficiente.

Para contornar este problema, existe a pesquisa semântica, onde não apenas pesquisamos palavras, mas também conceitos.

Com este intuito, desenvolveu-se este projecto que tem como finalidade a pesquisa semântica num domínio específico.

2. Domínio

O domínio escolhido para este projecto é o Futebol.

3. Fonte de dados

Foram escolhidas dois tipos de fontes de dados. Um para retirar informação necessária para preenchimento da ontologia com dados dos jogadores, clubes e competições. Este preenchimento constitui o primeiro paço. O outro tipo de fonte será usado para enriquecimento da ontologia com a indexação de notícias retiradas de *feeds RSS*.

Fontes de dados para informação dos Jogadores, Clubes e Competições:

- ZeroZero, http://www.zerozero.pt/

Fonte de dados para *feeds RSS*:

- ZeroZero, http://www.zerozero.pt/rss/noticias.php
- A Bola, http://www.abola.pt/rss/index.aspx
- Uefa, http://pt.uefa.com/rssfeed/index.xml
- Rádio Renascença:
  - http://www.rr.pt/rssFeed.aspx?fid=73
  - http://www.rr.pt/rssFeed.aspx?fid=74
  - http://www.rr.pt/rssFeed.aspx?fid=76
  - http://www.rr.pt/rssFeed.aspx?fid=77
  - http://www.rr.pt/rssFeed.aspx?fid=78
- Jogo de Área, http://www.jogodearea.com/feed/
- Desporto Sapo, http://services.sapo.pt/RSS/Feed/sapo/desporto/teasers
- Futebol Finance, http://www.futebolfinance.com/feed
- Futebol 365, http://feeds.feedburner.com/futebol365/noticias?format=xml

4. Arquitectura

Neste projecto estão envolvidas duas aplicações distintas. Um servidor, e um *WebServer* cliente.

O servidor disponibiliza uma API através de *WebServices* que permite ao *WebServer* cliente aceder às suas funcionalidades.

Os serviços disponibilizados pelo servidor são: *Update*, *Browsing*, *Pesquisa* e *Sugestões*.

O *WebServer* utiliza os serviços fornecidos pelo servidor para disponibilizar ao *browser* cliente os dados requeridos por este.

Na camada de dados foi utilizada a biblioteca *Jena*. Esta biblioteca é responsável pela organização dos dados da ontologia na base de dados, fazendo o papel de homogeneização de dados para as camadas superiores.

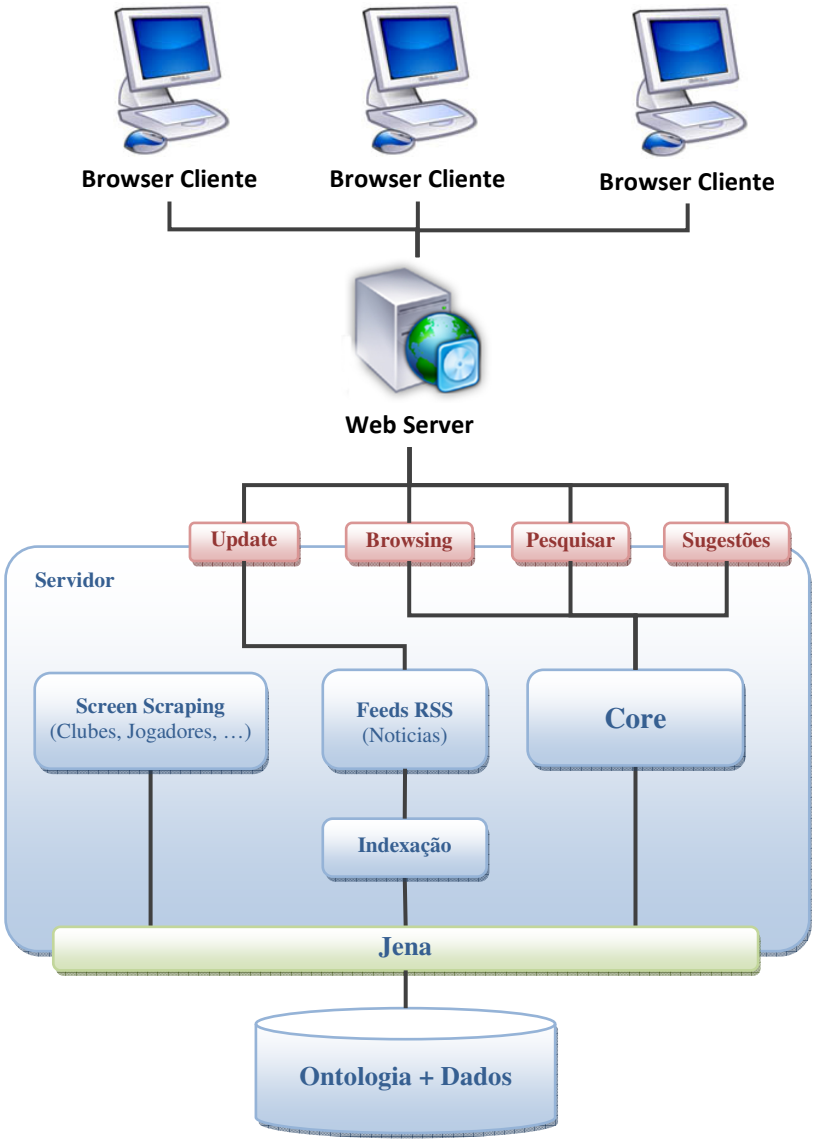

Figura 1 – Arquitectura do Sistema

5. Ontologia

A ontologia escolhida para este trabalho baseia-se em conceitos relacionados com a temática do futebol. Sendo assim, definimos um conjunto de 6 conceitos (Jogadores, Clubes, Competições, Cidades, Países e Noticias) e relações existentes entre estes, tal como poderá ser observado na figura seguinte.

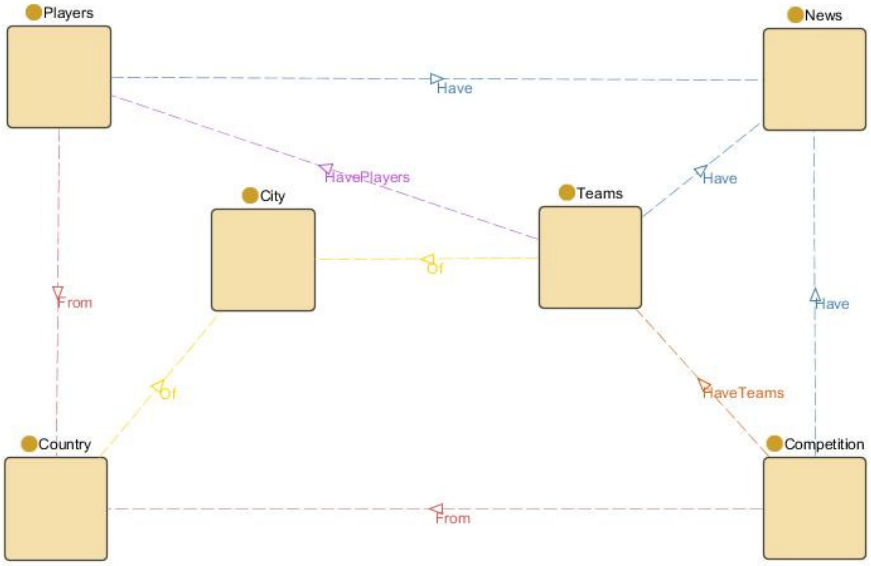

Figura 2 - Ontologia

6. Desenvolvimento

Para o desenvolvimento desta aplicação recorreu-se a plataforma J2EE e à seguinte lista de bibliotecas:

- *Jena* – Para armazenamento da ontologia;
- *Sparql* – Para optimização de *querys* à ontologia;
- *Rome* e *Jdom*- Para leitura de feeds.

A aplicação é composta de duas partes: O Cliente e o Servidor.

a. Cliente

A aplicação cliente foi desenvolvida em JSP, e tem como função permitir a interacção do utilizador com a ontologia, usando como como servidor aplicacional o *GlassFish*. O utilizador pode assim navegar na ontologia e fazer pesquisas, tendo ainda acesso a sugestões. Para ter acesso às últimas noticias, o utilizador poderá fazer o download das mesmas recorrendo a função de Update que lhe é facultada na aplicação.

b. Servidor

Para o desenvolvimento do servidor foi criado um módulo EJB (*Enterprise JavaBean*) que também corre em cima do servidor aplicacional *GlassFish* e disponibiliza um *API* baseada em *webservices* às aplicações clientes.

Seguidamente serão descritas as funcionalidades do servidor.

i. Criação da ontologia

Para a criação da ontologia recorremos à aplicação protégé, que permite a geração da ontologia em formato OWL. Este ficheiro foi posteriormente usado pela aplicação juntamente com o jena para geração da ontologia na base de dados.

ii. População da ontologia

Relativamente a fase de população da ontologia, esta é efectuada em duas fases. A primeira consiste na utilização de *Screen Scraping* para recolha dos dados do site *zerozero.pt* com a respectiva população dos jogadores, clubes, competições, países e cidades.

A segunda fase já está disponível através de um serviço disponibilizado na API e consiste na população da ontologia recorrendo a *feeds* RSS. Após a recolha das *feeds*, estas passam por um processo de indexação. O processo de indexação consiste na associação das notícias aos diferentes elementos presentes na ontologia. Para isso, é verificado a presença de cada um desses elementos em cada notícia analisada. Caso se verifique essa presença, a notícia é associada ao elemento.

iii. Browsing

O Browsing é uma das várias funcionalidades disponibilisadas ao utilizador pelo nosso motor de pesquisa semântica, cujo objectivo é de permitir ao próprio a exploração dos dados existentes na ontologia.

Esses dados estão organizados em 3 níveis: Competições → Clubes → Jogadores. Ao seleccionar um dos elementos pertencentes a uma dessas categorias, o utilizador acede as noticias relacionadas com esse elemento, ordenadas por data das mais recentes às mais antigas.

iv. Pesquisa

Outra funcionalidade essencial para a nossa aplicação é a pesquisa. Através desta funcionalidade, o utilizador especifica ao motor de pesquisa qual a palavra ou conjunto de palavras que este deseja pesquisar na ontologia, de forma a aceder a noticias relacionadas com estas. No caso em que a *query* de pesquisa contenha mais do que um parâmetro, usamos o conceito de anagramas de forma a permitir uma exploração de várias combinações entre esses parâmetros e assim melhorar a nossa pesquisa. Parâmetros com menos de 3 letras são ignorados pelo nosso motor de pesquisa.

Os resultados são depois apresentados ao utilizador de forma ordenada, usando para tal um *ranking* associado a cada notícia, afinada recorrendo a data da notícia. O processo de *ranking* envolve assim 3 passos:

- Vai-se buscar todas as noticias associadas a cada anagrama obtido da query de pesquisa, privilegiando os de maior tamanho, e atribui-se uma pontuação inicial de acordo com esse tamanho.
- Analisa-se a ocorrência dos anagramas no titulo e corpo das menssagens, sendo que ocorrências no titulo obtém maior pontuação em relação as ocorrências no corpo da menssagem.
- Analisa-se as datas de todos as menssagens, melhorando as respectivas pontuações consoante a sua "idade" de forma a dar prioridade as menssagens mais recentes.

v. Sugestões

Por fim, a aplicação disponibiliza ainda sugestões para que o utilizador tenha acesso a notícias relevantes que não estejam directamente associadas aos parâmetros da pesquisa, mas que lhe estejam relacionadas.

A ideia é que, no caso do Browsing se efectuar uma pesquisa sobre o "Porto", possa ter acesso a noticias relacionadas com competições onde este esteja presente ou ainda com jogadores pertencentes a equipa.

No caso de uma pesquisa por query, seleccionamos a notícia mais relevante devolvida pela mesma. Usando o titulo dessa noticia como query, realiza-se o mesmo processo que aquele usado para uma pesquisa normal por query.

Mais uma vez, as notícias são ordenadas usando a mesma abordagem seguida na pesquisa por query.

Todas as funcionalidades revistas anteriormente, estão ilustradas na figura 3.

Figura 3 – Apresentação do cliente

7. Estrutura e organização do código

Relativamente à estrutura dó código, este foi organizado com recurso aos *packages* em java. Ficando o servidor com a seguinte estrutura:

Package	Descrição
Database	**Este *package* contém as classes base da aplicação.**
Database.Anagrama	Classe usada para definir as operações referentes aos anagramas usados em operações de pesquisa. É aqui que é atribuída a ponderação a cada notícia e posteriormente são ordenadas todas as notícias antes de serem apresentadas ao utilizador.

Database.Config	Classe usada para definição de parâmetros de gestão da base de dados.
Database.DataBase	Classe principal da aplicação. É aqui que se encontra praticamente todo o processamento central da aplicação. Contém métodos para a indexação das notícias, para carregamento da ontologia, para actualização das notícias, métodos responsáveis por processarem as pesquisas, etc.
Database.SearchResultsNews	Classe que representa uma noticia ponderada. Usada quando se processam as pesquisas efectuadas pelo utilizador.
FutebolService	**Esta *package* contém as classes usadas pelo *webservice*.**
FutebolService.Tree	Classe que define toda a árvore dos elementos presentes na ontologia, posteriormente apresentada no browser do utilizador.
FutebolService.TreeCompetition	Classe que define um elemento "competição" na árvore.
FutebolService.TreePlayer	Classe que define um elemento "jogador" na árvore.
FutebolService.TreeTeam	Classe que define um elemento "clube" na árvore.
Indexação	**Esta *package* contém as classes usadas para a indexação das notícias.**
Indexação.DataComparator	Classe responsável por fazer a comparação de 2 datas. Usada quando é feita a ordenação das noticias por data.
Indexação.News	Classe que define uma noticia na ontologia, sem qualquer ponderação.
Indexação.RSS	Classe que define um feed RSS. Usada para guardar a informação dos RSS extraídos quando é feita a actualização das notícias na base de dados ontológica.
ScreenScrapping	**Esta *package* contém todas as classes usadas na operação de *ScreenScraping*.**
ScreenScrapping.Clube	Classe que define um elemento "clube" na ontologia.
ScreenScrapping.Jogador	Classe que define um elemento "jogador" na ontologia.
ScreenScrapping.Liga	Classe que define um elemento "competição" na ontologia.

Util	
Util.Util	Classe contendo métodos gerais usados por algumas classes.

8. Conclusão

Este trabalho permite observar uma das potencialidades que a Web Semântica pode trazer aos motores de pesquisa.

Usando uma plataforma de conhecimento (ontologia), este motor de pesquisa simplificado consegue mostrar ao utilizador noticias relacionados, não só com as palavras da *query* de pesquisa, como também com elementos relacionados com essas palavras, através da análise do significado das mesmas palavras.

O utilizador vê assim o seu espaço de informação alargado a informação que lhe pode ser relevante para a sua pesquisa.

Relativamente ao trabalho desenvolvido, verificou-se após vários testes que a plataforma gera resultados relativamente satisfatórios.

9. Carga horária extra aula

- **Meta 1:** Definição o Domínio, Arquitectura, Fonte de dados, e Ontologia (6 horas)
- **Meta 2:** *Screen Scrapping*, Extracção de tópicos e Indexação (64 horas)
- **Meta 3:** Implementação da interface com *pesquisa*, *browsing* e *sugestões* (108 horas)
- **Meta 4:** Relatório final, apresentação e melhorias (14 horas)

Nota: O número de horas apresentado refere-se ao trabalho efectuado em grupo, como o grupo é composto por dois elementos, ter-se-á de multiplicar por 2.

10. Referências

[1] Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/

[2] Protégé, Open source ontology editor and knowledge-base framework, http://protege.stanford.edu/

[3] Rome, Open source Java tools for parsing, generating and publishing RSS and Atom feeds, https://rome.dev.java.net/

[4] Sparql, RDF query language, http://www.w3.org/2009/sparql/wiki/Main_Page