Instituto Superior de Engenharia do Porto

Indexação e Pesquisa de Informação com Base em Ontologias Interligadas

Ricardo André de Sousa Jafe

Dissertação para obtenção do Grau de Mestre em Engenharia Informática

Área de Especialização em Tecnologias do Conhecimento e da Decisão

Orientador: Prof. Doutor Nuno Silva

Júri:

Presidente: Prof. Doutora Maria de Fátima Coutinho Rodrigues, Professora Coordenadora, Instituto Superior de Engenharia do Porto

Vogais: Prof. José Carlos Ramalho, Professor Auxiliar, Universidade do Minho

Prof. Nuno Alexandre Pinto da Silva, Professor Coordenador, Instituto Superior de Engenharia do Porto

Porto, Outubro de 2011

Dedico este trabalho aos meus pais e à Mariana

Agradecimentos

Gostaria de agradecer em primeiro lugar aos meus pais, **Alberto Oliveira Jafe** e **Otília Santos Sousa Jafe** pela compreensão e suporte nos meus melhores e piores momentos, devo a eles a vasta maioria do que tenho conseguido atingir durante o meu percurso de vida.

Em segundo lugar um obrigado à **Mariana Baldé**, pelo enorme carinho e dedicação que me tem dado, apoio e motivação. Espero sempre conseguir seguir o seu exemplo enquanto ser humano.

Não podia deixar de agradecer aos meus amigos, em especial ao meu "bro" **Jorge Lopes**, ao *João Figueiredo* e ao *José Soares* pelo companheirismo e bons momentos proporcionados ao longo de mais de 20 anos. Sempre que preciso relembrar o que são bons amigos, a tarefa é fácil, basta recordar-me dos vossos nomes.

Quero também agradecer ao ISEP, mais propriamente a alguns docentes do Departamento de Engenharia Informática. Especificamente, ao *Doutor Filipe Pacheco Paulo*, *Doutor Jorge Pinto Leite*, *Doutor Paulo Jorge Machado Oliveira* e *Doutora Berta Pinheiro* cuja dedicação pelo trabalho e alunos me inspirou durante a minha Licenciatura e Mestrado.

Uma palavra especial de agradecimento ao meu orientador *Doutor Nuno Pinto Silva* que me acolheu como seu orientando e como colaborador no projecto Coalesce. Sem as suas críticas construtivas, palavras encaminhadoras e frontalidade total, este trabalho não teria sido possível.

Quero também agradecer à equipa de investigadores do GECAD que me acolheu e ajudou durante o meu percurso de aprendizagem e também durante a fase inicial deste projecto. Em especial, *Paulo Maio*, *Hélio Martins* pelos seus conselhos durante a apresentação do meu trabalho.

Um abraço aos meus companheiros de "guerra", *Rafael Peixoto*, *Óscar Miranda*, *Luís Correia* e *Ricardo Brandão*. Graças a vocês a sala B408 tem e terá sempre "outro sabor".

Um agradecimento ao grupo representativo do meu ano de entrada, 2003, intitulado *Os Treix*.

Finalmente um abraço sentido aos meus colegas de curso que me aturaram durante as minhas palhaçadas, piadas e desatinos. Um abraço a **Liliana Alexandre**, **Pedro Duque**, **António Sousa**, **Rui Silva**, **Vânia Lopes**.

Resumo Português

Tecnologias da Web Semântica como RDF, OWL e SPARQL sofreram nos últimos anos um forte crescimento e aceitação. Projectos como a DBPedia e Open Street Map começam a evidenciar o verdadeiro potencial da Linked Open Data. No entanto os motores de pesquisa semânticos ainda estão atrasados neste *crescendo* de tecnologias semânticas. As soluções disponíveis baseiam-se mais em recursos de processamento de linguagem natural. Ferramentas poderosas da Web Semântica como ontologias, motores de inferência e linguagens de pesquisa semântica não são ainda comuns.

Adicionalmente a esta realidade, existem certas dificuldades na implementação de um Motor de Pesquisa Semântico. Conforme demonstrado nesta dissertação, é necessária uma arquitectura federada de forma a aproveitar todo o potencial da Linked Open Data. No entanto um sistema federado nesse ambiente apresenta problemas de performance que devem ser resolvidos através de cooperação entre fontes de dados. O *standard* actual de linguagem de pesquisa na Web Semântica, o SPARQL, não oferece um mecanismo para cooperação entre fontes de dados. Esta dissertação propõe uma arquitectura federada que contém mecanismos que permitem cooperação entre fontes de dados. Aborda o problema da performance propondo um índice gerido de forma centralizada assim como mapeamentos entre os modelos de dados de cada fonte de dados. A arquitectura proposta é modular, permitindo um crescimento de repositórios e funcionalidades simples e de forma descentralizada, à semelhança da Linked Open Data e da própria World Wide Web.

Esta arquitectura trabalha com pesquisas por termos em linguagem natural e também com inquéritos formais em linguagem SPARQL. No entanto os repositórios considerados contêm apenas dados em formato RDF.

Esta dissertação baseia-se em múltiplas ontologias partilhadas e interligadas.

Palavras-chave

Web Semântica, Motores de Pesquisa, Ontologias

Abstract

Semantic Web technologies like RDF, OWL and SPARQL have suffered great growth and acceptance lately. Projects like the DBPedia and Open Street Map start to show the true potential of the Linking Open Data project. However Semantic Search engines still lag behind in this semantic technologies crescendo. The solutions available still rely on natural language processing. Powerful Semantic Web tools like ontologies, reasoners and semantic search languages are still not common.

Adding to this reality, there are some difficulties in implementing a semantic search engine. As demonstrated in this dissertation, a federated architecture is necessary to take advantage of the full potential of the Linked Open Data. However a federated system in such environment has performance problems that must be solved by allowing cooperation between data sources. The current standard of semantic search, SPARQL, does not provide a mechanism for data source cooperation. This dissertation proposes a federated architecture with built-in mechanisms for cooperation between data sources. It also addresses the performance problem by proposing a centrally managed index structure and mappings between the data models of each data source. The proposed architecture is modular, allowing for repository and feature growth in a simple and decentralized way, just like the Linked Open Data and Word Wide Web.

This architecture deals with query terms in natural language and with formal SPARQL queries. However the repositories considered are only of RDF data.

This dissertation is based on multiple shared and interconnected ontologies.

Keywords

Semantic Web, Search Engines, Ontologies

Índice

Índice de Imagens

Índice de Tabelas

Glossário

Apache Lucene – Motor de Indexação e Pesquisa open-source desenvolvido pela Apache Foundation. Utiliza índices invertidos como estrutura de armazenamento.

ARQ – Parte integrante da ferramenta Jena, trata-se de um motor de pesquisa de RDF que implementa SPARQL.

Crawler – Programa que procura recursivamente todos os *links* dentro de um site, indexando toda a informação que encontra. No contexto de Linked Open Data, pode ser utilizado para indexar todo um repositório.

Data Dump – Um ficheiro que contém uma cópia de todos os dados contidos num determinado repositório em determinada data. Repositórios de dados RDF normalmente fornecem este recurso de forma a simplificar o acesso aos dados por parte de programadores e investigadores.

Jena – Ferramenta de manipulação de ontologias escritas em RDF e OWL, desenvolvida pela empresa HP. Fornece uma API em Java para a criação, edição e consulta de ontologias.

LARQ – Componente da ferramenta Jena, permite utilizar índices invertidos baseados no Apache Lucene juntamente com pesquisas ARQ, de forma a optimizar o tempo de resposta.

Long Tail – Um inquérito pouco popular, que não é bem respondido por um motor de pesquisa normal, pois devido à escassez de resultados e número de vezes que é efectuado obtém uma fraca pontuação em algoritmos de ranking baseados na popularidade. É estimado que a totalidade de pesquisas Long Tail seja mais numerosa que as pesquisas mais populares[1].

OWL – Web Ontology Language, é uma linguagem de especificação de conhecimento, utilizada para criar ontologias. Possui vários dialectos e perfis, fornecendo vastos recursos de expressividade.

Query – As palavras *query* e inquérito são usadas ao longo deste relatório para se referirem ao mesmo processo. O acto de inquirir um repositório de dados com o objectivo de extrair um subconjunto desses dados que correspondam a um padrão presente no inquérito. É comum utilizar-se o estrangeirismo *query* para se referir a este tipo de inquéritos.

SPARQL Endpoint – Ponto de acesso a inquéritos em linguagem SPARQL, utilizando o protocolo HTTP.

[1] http://en.wikipedia.org/wiki/Long_Tail

Lista de Abreviaturas

HTTP – HyperText Transfer Protocol é um protocolo de comunicação utilizado em sistemas de informação de hipermédia distribuídos e colaborativos.

LOD – Linked Open Data, designação atribuída a dados, geralmente sob formato RDF, que se encontram ligados entre si através de URIs.

NLP – Natural Language Processing

RDF – Resource Description Framework, uma linguagem de especificação de recursos, baseada em XML.

SPARQL – **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage, um acrónimo recursivo que representa uma linguagem de pesquisa de recursos RDF. É uma recomendação W3C.

SGML – Standard Generalized Markup Language

SSE – Semantic Search Engine

SW – Semantic Web

TBL – Sir Tim Berners Lee, inventor da World Wide Web e pioneiro da Semantic Web

URI – Uniform Resource Identifier, um identificador único dado a um recurso, com o objectivo de tornar a referência a esse recurso única e não-ambígua.

1 Introdução

1.1 Enquadramento

Tecnologias da Web Semântica como RDF, OWL e SPARQL sofreram nos últimos anos um forte crescimento e aceitação. Projectos como a DBPedia e Open Street Map começam a evidenciar o verdadeiro potencial da Linked Open Data. No entanto os motores de pesquisa semânticos ainda estão atrasados neste *crescendo* de tecnologias semânticas. As soluções disponíveis baseiam-se mais em recursos de processamento de linguagem natural. Ferramentas poderosas da Web Semântica como ontologias, motores de inferência e linguagens de pesquisa semântica não são ainda comuns.

Adicionalmente a esta realidade, existem certas dificuldades na implementação de um Motor de Pesquisa Semântico. Conforme demonstrado nesta dissertação, é necessária uma arquitectura federada de forma a aproveitar todo o potencial da Linked Open Data. No entanto um sistema federado nesse ambiente apresenta problemas de performance que devem ser resolvidos através de cooperação entre fontes de dados. O *standard* actual de linguagem de pesquisa na Web Semântica, o SPARQL, não oferece um mecanismo para cooperação entre fontes de dados. Esta dissertação propõe uma arquitectura federada que contém mecanismos que permitem cooperação entre fontes de dados. Aborda o problema da performance propondo um índice gerido de forma centralizada assim como mapeamentos entre os modelos de dados de cada fonte de dados. A arquitectura proposta é modular, permitindo um crescimento de repositórios e funcionalidades simples e de forma descentralizada, à semelhança da Linked Open Data e da própria World Wide Web.

Esta arquitectura trabalha com pesquisas por termos em linguagem natural e também com inquéritos formais em linguagem SPARQL. No entanto os repositórios considerados contêm apenas dados em formato RDF.

Esta dissertação baseia-se em múltiplas ontologias partilhadas e interligadas.

1.2 Motivo

A visão de uma Web Semântica descrita por Tim Berners Lee em 2001 tem ficado cada vez mais próxima [Berners-Lee et al. 2001] . Desde então foram criados novos *standards* de representação de conhecimento como o RDF(S), OWL e OWL 2, foi estabelecida uma recomendação W3C para uma linguagem de pesquisa de ontologias (1), assim como elaborada uma linguagem de modificação de ontologias (2). Todas estas tecnologias podem ser utilizadas via pedidos HTTP, mantendo assim interoperabilidade com as várias facetas da internet, seguindo a filosofia original da World Wide Web onde cada camada é acrescentada à camada anterior, mantendo a compatibilidade entre recursos.

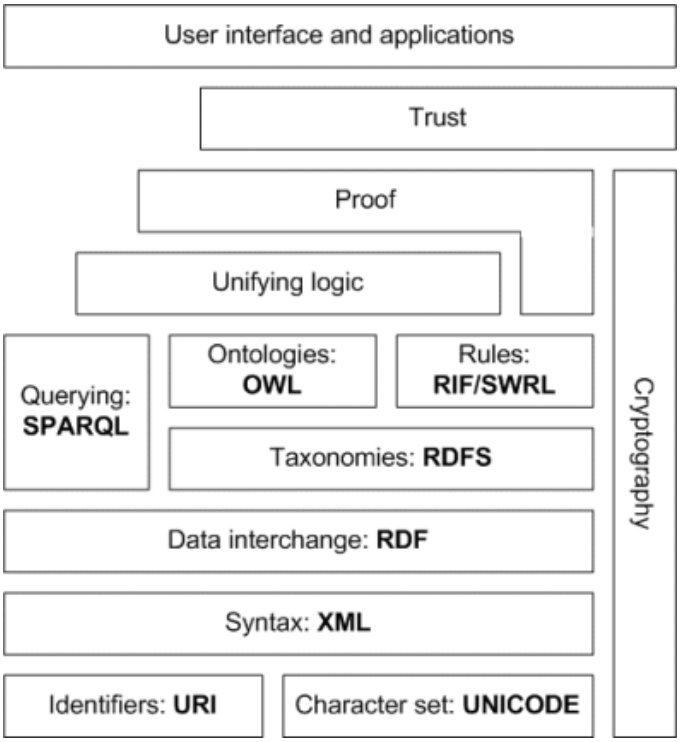

Imagem 1 - Stack de Tecnologias da Web Semântica[2]

Nessa visão de 2001, TBL falava de uma *web* que compreendia o que o utilizador pretendia, e disponibilizava o seu conteúdo a aplicações, denominadas agentes, que automaticamente efectuavam tarefas repetitivas do dia-a-dia. Marcação de consultas, busca de notícias relacionadas com os interesses do utilizador, pesquisa pelo melhor preço de determinado utensílio, etc. Uma *web* que torna a vida quotidiana humana mais simples, tornando aplicações mais poderosas e em última análise contribuindo para uma melhoria da qualidade de vida.

A ideia deste trabalho surgiu após o autor assistir a uma palestra no site da TED[3], onde TBL fala de um movimento de “Open Data”, onde cada utilizador faz a sua parte pondo os dados que possui online, de forma livre, anónima e gratuita.

[2] Fonte: http://en.wikipedia.org/wiki/File:Semantic-web-stack.png
[3] http://www.ted.com/talks/lang/eng/tim_berners_lee_the_year_open_data_went_worldwide.html

Esta atitude simples permite que dados em bruto de várias fontes sejam cruzados e processados por aplicações simples com as mais variadas utilizações: desde um mapa detalhado do mundo inteiro a estatísticas eleitorais que cruzam vários aspectos do nível de vida, risco e saúde do povo afegão de forma a tirar conclusões sobre as suas escolhas eleitorais até um mapa detalhado dos campos de refugiados após a tragédia no Haiti em 2010. Tudo isto construído com dados e intervenções da comunidade de internautas.

De forma a aproveitar ao máximo esta quantidade imensa de dados que surgem é necessária uma forma de interligar informações estruturadas. Os motores de pesquisa actuais fazem um bom trabalho de indexação da *web*, mas estão ainda a dar os primeiros passos no que toca a compreender esses dados.

O contributo que este trabalho espera deixar é uma análise detalhada das várias tecnologias necessárias para construir um Motor de Pesquisa Semântico, com recurso a artefactos de representação de conhecimento, as ontologias. Por se tratar de um projecto *web*, e porque a definição de ontologia assume que esta é partilhada e aceite por uma comunidade, seria pouco útil que as ontologias utilizadas para mapear o conhecimento extraído fossem estanques e fechadas. Assim, foi feita uma pesquisa no sentido de interligar ontologias com base em mapeamentos feitos pela comunidade de utilizadores, uma solução já adoptada pelo projecto DBpedia, numa ferramenta denominada DBPedia Mappings (3).

Como complemento a esta investigação do estado da arte foi construído um protótipo de um motor de pesquisa para dados RDF, que utiliza ficheiros locais assim como SPARQL Endpoints. Este protótipo assenta numa arquitectura federada, que será discutida e analisada em profundidade neste documento.

Na imagem 1 podemos ver o leque de tecnologias que compõe a hierarquia da Web Semântica. Embora durante este relatório não se fale pormenorizadamente de cada uma delas, por ser pressuposto algum conhecimento de fundo, nomeadamente em XML, RDF e URIs, é importante ilustrar a sua existência, pois as várias camadas serão referidas durante a dissertação.

Visto esta dissertação ser na área de motores de pesquisa semânticos, torna-se primordial fazer uma análise do que é considerado pelo autor como um motor de pesquisa semântico. Esta análise é realizada no capítulo seguinte.

1.3 Organização da Dissertação

Esta dissertação encontra-se organizada em 5 capítulos, organizados da seguinte forma:

- Capitulo 1 – Introdução: Descreve sucintamente os objectivos e resultados obtidos com este trabalho, assim como a motivação do autor em explorar este tema na sua dissertação de Mestrado.
- Capitulo 2 – Contexto: Apresenta definições e interpretações importantes para a compreensão da matéria abordada nesta dissertação, assim como a descrição de tecnologias e do Estado da Arte, incluindo um estudo a implementações concretas de motores de pesquisa semânticos.
- Capitulo 3 – Proposta de Arquitectura Federada com Cooperação entre Fontes de Dados: É feita uma descrição de cada componente da arquitectura proposta pelo autor, assim como evidenciadas as vantagens face às arquitecturas analisadas anteriormente. No final do capítulo são descritos alguns cenários de pesquisas.
- Capitulo 4 – Descrição da Solução Técnica Utilizada: É feita uma descrição da implementação da arquitectura efectuada para fins de testes e estudo prático. São descritas as tecnologias utilizadas e no final apresentado um exemplo prático de utilização do sistema.
- Capitulo 5 – Conclusões: São descritos os resultados deste trabalho, assim como caminhos futuros de investigação e desenvolvimento. É também feita uma breve análise à evolução do âmbito desta dissertação.

Como complemento aos capítulos foram adicionados 2 anexos:

- Anexo A – Tabelas elaboradas durante a fase de estudo do estado da arte: Tabelas que foram elaboradas a partir de testes práticos com os motores de pesquisa analisados no capítulo 4.
- Anexo B – Descrição de Propriedades de Linguagens de Especificação de Ontologias: Descrição das propriedades utilizadas para elaborar a tabela 5 desta dissertação.

2 Contexto

2.1 Definição de Semântica

Antes de proceder à definição de motor de pesquisa semântico, será útil discursar brevemente sobre o significado da palavra semântica. O étimo da palavra é de origem grega:

Tabela 1 - Étimo grego da palavra Semântica segundo Wikipédia *(4)*

Semântica (do grego *σημαντικός*, *sēmantiká*, plural neutro de *sēmantikós*, derivado de *sema*, sinal)

Significando o estudo do *significado.* Segundo esta vertente linguística, quando falamos na semântica de uma frase, *independentemente* da linguagem, estamos a focalizar a atenção nas relações entre elementos dessa frase e ainda nos atributos de cada um desses elementos (4). Como exemplo útil para este documento podemos introduzir uma frase de programação, em Java:

Tabela 2 - Exemplo de linha de programação Java

File myFile = new MyFile();

Se pretendermos atentar apenas à semântica desta frase, torna-se necessário primeiro estabelecer um conjunto de pressupostos que tornarão a análise semântica possível.

Comecemos por aceitar a sintaxe da linguagem de programação Java, tal como definida pela empresa Sun em 2000 no capítulo 18 da publicação “Java Language Specification – Second Edition” (5).

Aceitemos também que a frase foi escrita segundo uma convenção que determina que um nome de uma variável é escrito com a primeira letra minúscula e o resto das palavras em estilo CamelCase[4]. No caso de ser uma classe o nome é escrito com letra maiúscula.

Aceitemos finalmente que esta frase está sintacticamente correcta, e que não origina erro de compilação.

Após a definição destas condições, estamos aptos a *extrair* conhecimento da frase anteriormente descrita.

Sabemos que myFile deve ser uma variável, pois não está na lista de palavras reservadas da linguagem e começa por letra minúscula. Sabemos também que existe uma classe File e que a variável supra-referida é do tipo File. É possível verificar ainda que esta variável está a ser inicializada como uma instância da classe MyFile.

Podemos construir a seguinte lista de factos a partir da frase anterior:

[4] http://pt.wikipedia.org/wiki/CamelCase

1. File é_uma Classe_Java
2. myFile é_uma variável
3. myFile tem_tipo File
4. myFile é_instancia_de MyFile
5. MyFile é_uma Classe_Java
6. MyFile é_subclasse_de File

Outros factos poder-se-iam concluir, sendo que estes são suficientes para os propósitos desta demonstração.

O conjunto de factos extraído a partir da frase em linguagem formal constitui o significado, ou semântica que estão presentes na frase, e que foram agora capturados por esta análise. De notar que estas deduções não eram possíveis antes de serem determinados e aceites os três pressupostos iniciais.

Uma característica universal da semântica é que se baseia num conjunto de regras já existentes de forma a interpretar novos factos.

De facto, o processamento que um motor de pesquisa semântico tem que fazer a um qualquer documento (seja texto em linguagem natural, um ficheiro com triplos RDF ou uma base de dados) é bastante similar com o que um compilador tem que fazer a uma linha de código. No entanto, com objectivos bastante diferentes. A tarefa do compilador é garantir que a sintaxe da linguagem foi respeitada, e que não existem erros no código. Já a tarefa do motor de pesquisa semântico é a de interpretar o significado presente no documento, armazena-lo e indexa-lo, de forma a ser facilmente acessível.

Neste ponto penso que é útil propor uma igualdade entre “semântica” e “significado” para os objectivos desta dissertação.

Quando se fala de semântica durante este documento está-se a falar do significado de um determinado conjunto de palavras, no contexto de uma linguagem previamente aceite.

No caso do protótipo da arquitectura proposta na parte final da dissertação, o contexto é a ontologia.

2.2 Definir um Motor de Pesquisa Semântico

A forma tradicional de pesquisa na *web* consiste na inserção de termos de interesse num motor de pesquisa de forma a obter uma lista de documentos, os resultados. Posteriormente cabe ao utilizador navegar por cada um dos resultados à procura de informação que lhe interesse. Por norma, resultados melhores resultam de inquéritos simples e populares, pois estes terão um alto ranking nos algoritmos de pesquisa do motor, e serão piores para inquéritos complexos e pouco populares (4) [Wu et al. 2008].

Imagem 2 - Modo tradicional de Pesquisa

Uma área onde os motores de pesquisa normalmente falham é na busca de termos similares. Esperar obter um resultado que fale de “canetas” quando se pesquisou apenas por “esferográficas” está ainda longe do que um motor de pesquisa tradicional consegue fazer sem ajuda de alguma SEO (*Search Engine Optimization*).

Temos ainda que ter em consideração o reconhecimento do contexto de um inquérito. Se for efectuada uma pesquisa no Google por “Jaguar adulto exposição Portugal” os primeiros resultados a surgirem referem-se a automóveis da marca Jaguar, e não a locais em Portugal onde podemos encontrar um Jaguar adulto em exposição. O motor teve dificuldade em compreender o que era pretendido pelo inquérito.

Com os três parágrafos anteriores é possível esboçar uma definição do que é esperado de um motor de pesquisa semântico. Trata-se de um tipo de motor de pesquisa que é sensível ao contexto do inquérito, filtrando resultados em áreas de conhecimento que não interessam, e ao mesmo tempo é capaz de reconhecer cruzamentos entre diferentes tipos

de dados, estabelecer ligações entre os termos do inquérito e processar essas ligações de forma a devolver resultados que sejam semanticamente relevantes e não apenas que possuam os termos inseridos no inquérito [Ilyas et al. 2005]. Deve portanto perceber o que lhe é perguntado [Manjula & Geetha 2004].

Um cenário exemplo de utilização:

O utilizador escreve na caixa de texto do motor semântico:

“Em que situações o actual primeiro-ministro Português se contradisse quanto à actual subida de impostos?”

A tecnologia por trás do motor de pesquisa semântico deve efectuar uma análise da frase inserida, identificando as partes do discurso relevantes (POS – *part of speach*). Posteriormente identificar o domínio da pergunta, recorrendo-se, por exemplo, a uma ontologia de conceitos e a algoritmos de identificação de contexto. As *keywords* Primeiro-ministro Português e Impostos deveriam identificar o domínio da pergunta como, por exemplo, “Política Interna Portuguesa”.

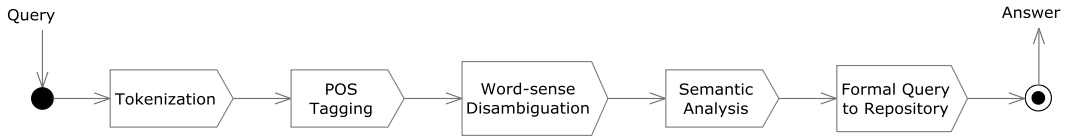

Imagem 3 - *Pipeline* de processos de um SSE

Após esta fase de tratamento linguístico, o sistema teria uma representação formal do que é pretendido pelo inquérito. Era altura de identificar os repositórios de dados que podem responder ao inquérito e efectuar a procura e agregação de informação.

Finalmente seria devolvida ao utilizador uma lista de declarações onde o Primeiro-ministro afirma não subir os impostos no seu mandato actual.

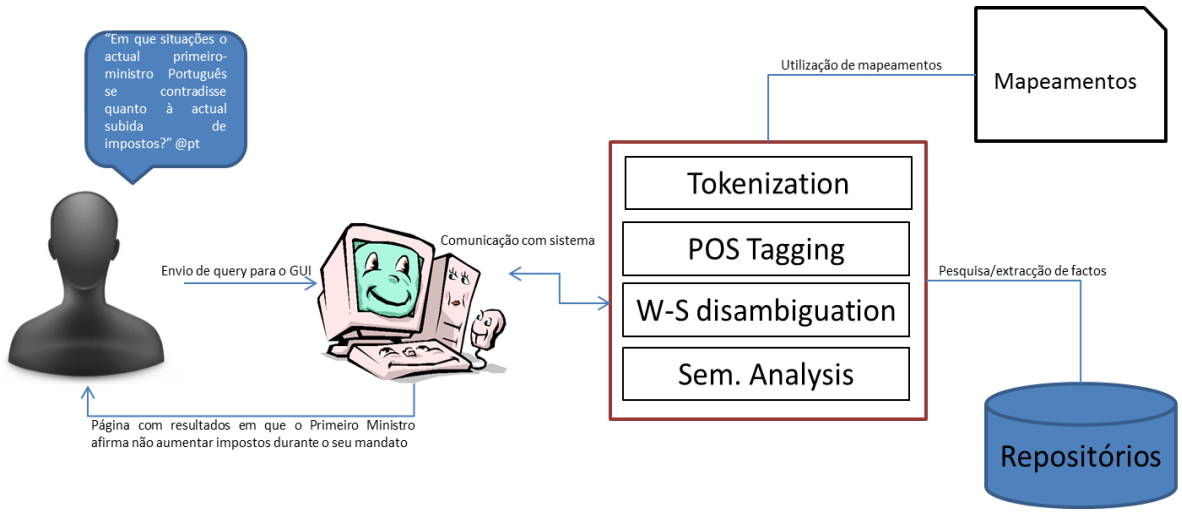

Imagem 4 - Exemplo de interacção Utilizador-SSE

2.3 Semântica – Duas formas de a Encarar

Para que o sistema descrito no capítulo anterior seja possível é necessário existir uma linguagem comum entre o utilizador e o motor de pesquisa. Nesta linguagem reside a semântica que torna possível uma análise complexa aos inquéritos do utilizador.

No entanto esta semântica deve estar também presente nos repositórios de informação de onde o sistema vai extrair a informação pretendida. De nada serve ao sistema perceber o inquérito feito se não conseguir responder ao mesmo.

Deste ponto de vista, a semântica pode ser vista por duas perspectivas:

- A semântica enquanto contrato Esquema-Repositório
- A semântica extraída da interpretação de uma ontologia

Ambas as perspectivas são descritas nos próximos subcapítulos.

2.3.1 A semântica enquanto Contrato Esquema-Repositório

Quando uma aplicação consulta uma base de dados, está subentendido um acordo formal quanto à estrutura dos dados. Este acordo é conhecido como o esquema da base de dados. Desde que esse esquema se mantenha, a aplicação consegue extrair dos dados do repositório a informação que necessita, assim como correctamente interpretar a sua semântica.

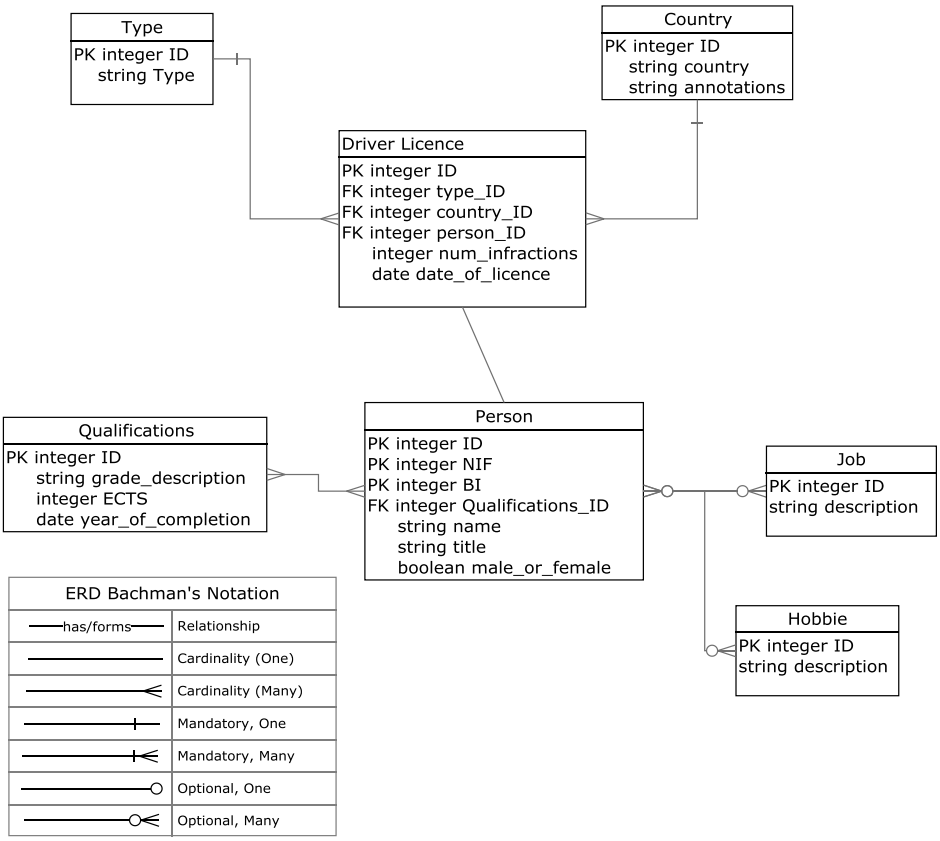

Imagem 5 – Esquema de base de dados típico (ER), segundo notação de Bachman[5]

Com este pressuposto, o interface gráfico de um motor de pesquisa pode ser elaborado de forma a obrigar o utilizador a aceitar este esquema de base de dados. Por exemplo, se estivermos a falar de uma base de dados de livros sobre Medicina, pode ser apresentado ao utilizador um menu com os vários temas, uma caixa de texto onde deve especificar o autor e outra para o título.

Dentro de este sistema existe uma semântica muito restrita e clara, e porque a base de dados contém toda a estrutura necessária para responder aos inquéritos, podemos afirmar que o sistema responde correctamente a cada inquérito do utilizador.

Mas na verdade isto não é dizer muito, pois o universo de inquéritos possíveis é muito limitado e nada flexível. Para adicionar mais flexibilidade é necessário alterar o esquema da base de dados, o motor de pesquisa e melhorar/aumentar a linguagem de inquéritos utilizada.

2.3.2 A semântica extraída da Interpretação de uma Ontologia

Quando temos um repositório com dados escritos numa determinada linguagem conhecida pelo sistema, mas cuja estrutura interna de dados é desconhecida, a tarefa de recolha de informação é consideravelmente mais complexa. Ao permitir que a estrutura dos dados varie

[5] http://en.wikipedia.org/wiki/Entity-relationship_model

torna-se necessário implementar um sistema capaz de extrair e organizar a informação contida no repositório sem metaconhecimento disponível no momento de implementação. As relações semânticas serão descobertas em tempo real, de acordo com o esquema escolhido para o repositório em questão (ontologia).

No entanto torna-se claro que este tipo de abordagem dá origem a um sistema mais flexível a mudanças de contexto e a evoluções tanto do repositório de dados como da ontologia pela qual esse repositório é interpretado.

Actualmente, um documento escrito em RDF permite descrever qual o esquema de dados utilizado, através da especificação de *namespaces* e fornece ainda a possibilidade de realizar inferência sobre os dados do documento, através de motores de inferência. Linguagens de especificação de conhecimento como RDF e OWL possuem uma expressividade mais alargada que uma linguagem típica de base de dados (como MySQL ou MS SQL Server). Explorando ao máximo estas capacidades é possível inferir novo conhecimento, a partir de factos contidos num repositório, utilizando por exemplo um motor de inferência como o Pellet [Sirin et al. 2007] ou o FaCT [Horrocks 1998] para descobrir novos factos a partir dos já existentes no repositório de informação.

Para além desta capacidade, o documento permite ao sistema descobrir a ontologia que especifica o conhecimento daquele documento, a partir do seu URI. Com esta informação é possível estabelecer mapeamentos entre diferentes ontologias, assim como extensões e evoluções que permitem ao sistema analisar a informação e processá-la, mesmo sem nunca ter sido programado para aquela especificação concreta de conhecimento. Uma limitação deste sistema é o requisito de um mapeamento entre conceitos da nova ontologia com outra conhecida do sistema. Por regra geral este mapeamento é feito por especialistas da área em questão, ou em alternativa feito por uma comunidade de utilizadores. A área de investigação de Ontology Matching encarrega-se de desenvolver ferramentas manuais [Maedche et al. 2002], supervisionadas [Hu & Qu 2008] e automáticas de mapeamento de ontologias, não havendo no entanto até ao momento métodos automáticos que produzam resultados suficientemente estáveis para serem utilizados em aplicações comerciais.

Uma questão importante deste processo de extracção de conhecimento é o reconhecimento de instâncias individuais. Este processo numa base de dados é garantido através da utilização do esquema da base de dados com identificadores únicos para cada tabela. No entanto num cenário de utilização de duas ontologias distintas mas relacionadas, é importante fazer um alinhamento de conceitos mas também de instâncias.

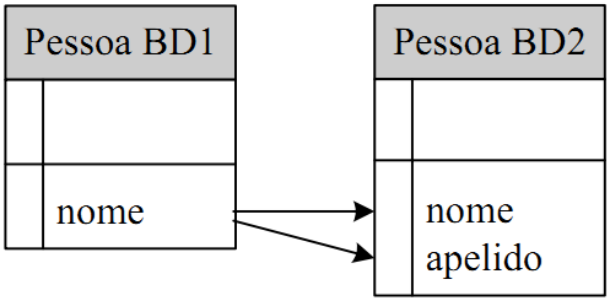

Imagem 6 - Exemplo de Integração de Informação segundo [Almeida 2009]

A imagem 6, extraída da Dissertação de Ricardo Almeida [Almeida 2009], demonstra um exemplo simplificado de integração de informação. Na base de dados BD1 o atributo que caracteriza o nome de uma Pessoa é um só. Já na base de dados BD2 o nome é dividido em dois atributos, nome e apelido. Se um sistema necessitar de efectuar uma pesquisa a ambas as bases de dados, de forma a conciliar a informação de ambas é necessário mapear BD1:Pessoa.nome[6] para BD2:Pessoa.nome e BD2:Pessoa.atributo. Em [Almeida 2009] é proposto um algoritmo que se encarrega deste processo de integração utilizando expressões regulares e mapeamentos manuais efectuados com a ferramenta MAFRA [Maedche et al. 2002].

Neste mesmo exemplo são utilizadas bases de dados genéricas, no entanto o mesmo se aplica no caso de duas ou mais ontologias sob as quais está a ser feito um mapeamento. Após o mapeamento de conceitos e atributos é necessário proceder à integração ou mapeamento de instâncias.

De forma a clarificar o conceito de ontologia na área da ciência dos computadores segue-se uma definição de ontologia normalmente aceite, e que é a utilizada nesta dissertação.

2.4 Definição de Ontologias Interligadas e Comunitárias

O elemento base da proposta deste documento é a ontologia. Uma ontologia trata-se de um artefacto escrito numa linguagem formal e que especifica uma conceptualização partilhada de um determinado domínio do conhecimento [Gruber 1993]

Segue-se uma paráfrase da definição original de Thomas Gruber:

“Uma ontologia é uma especificação explícita de uma conceptualização. […] Em tal ontologia, definições associam nomes de entidades no universo do discurso (por exemplo: classes, relações, funções) com textos perceptíveis por seres humanos que descrevem o significado dos nomes e os axiomas formais que restringem a interpretação e o uso indicado desses termos” [Gruber 1993]

[6] Sintaxe utilizada: [Base_de_Dados]:[Tabela].[atributo]

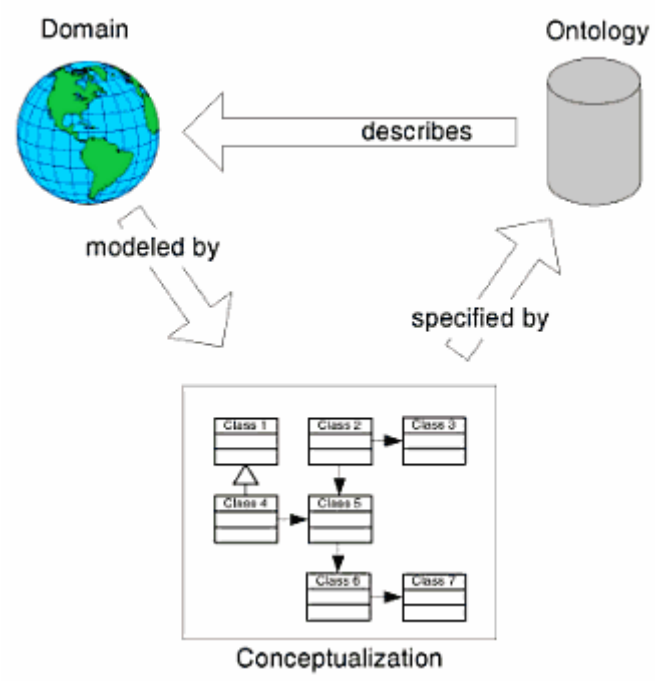

Imagem 7 - Definição de Ontologia segundo T. Gruber [Lacy 2005]

Gruber considera três conceitos na sua definição de ontologia, tal como evidenciado pela imagem 7.

1. O Universo do Discurso, ou Domínio
2. A Conceptualização desse Domínio
3. A Ontologia, que traduz em linguagem formal a Conceptualização.

O uso de ontologias tem especial importância na Web Semântica por permitir a partilha de conhecimento, e sua extensão, de uma forma que é compreensível por máquinas e seres humanos. Esta propriedade é benéfica pois permite o processamento desse conhecimento por parte de agentes, conforme já referido neste documento.

Embora a linguagem XML possa ser utilizada para representar explicitamente conhecimento, em [Kim 2002] é demonstrado que o XML não possui a expressividade requerida para eliminar definições ambíguas. Por este motivo, surgiram linguagens de especificação e consulta de ontologias que serão abordadas no próximo subcapítulo.

Neste trabalho o conceito de ontologia interligada representa a propriedade que uma ontologia tem de se relacionar com outras ontologias, através de recursos fornecidos pela linguagem de especificação, como é o caso de OWL e OWL 2 ou através de mapeamentos efectuados entre duas ontologias.

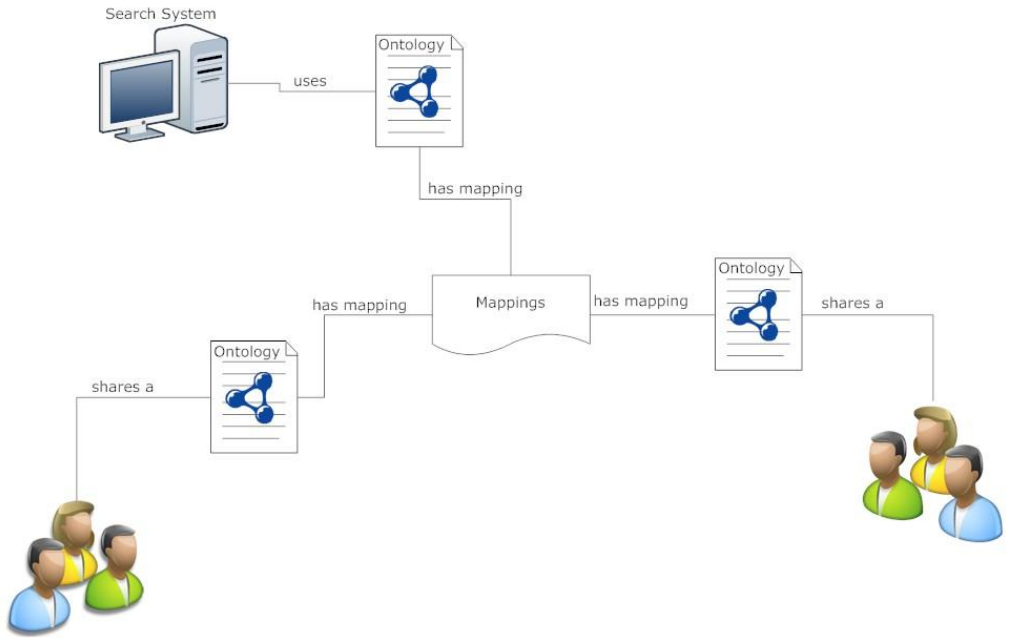

Imagem 8 - Ontologias Interligadas e Partilhadas

Conforme ilustrado na imagem 8, uma determinada comunidade aceita entre si e partilha uma conceptualização de um domínio, materializada sob a forma de uma ontologia. O mesmo acontece com outra comunidade. No caso de os domínios serem idênticos ou relacionados, é possível definir um conjunto de mapeamentos entre conceitos e propriedades de uma ontologia com conceitos e propriedades de outra. Desta forma torna-se possível que as duas comunidades comuniquem entre si, eliminando ambiguidades e contradições que poderiam existir.

Torna-se também possível que um agente, representado na imagem 8 por um sistema de pesquisa de informação, utilize uma ontologia individual onde mapeia os seus conceitos com os conceitos das ontologias da comunidade. Desta forma pode pesquisar e indexar informação em todas as comunidades subscritas.

Para que este intercâmbio seja possível é necessária uma linguagem de especificação de conhecimento, assim como uma que permita pesquisas de informação e actualizações, tal como acontece com uma base de dados relacional.

O órgão encarregado por definir os *standards* da Web Semântica é o World Wide Web Consortium, ou W3C (7). De acordo com este órgão, os *standards* de especificação e pesquisa na Web Semântica são a linguagem OWL e SPARQL respectivamente.

Seguidamente é feita uma descrição destas duas recomendações W3C, estudando suas capacidades e limitações, comparando-as com outras formas de especificação de conhecimento.

2.5 Linguagens de Especificação de Ontologias

Antes de abordar as características da linguagem OWL é importante justificar o porquê de se utilizar esta linguagem para especificação de ontologias e não outra. Desde que foi criado (8), o XML tem sido amplamente utilizado para especificar conhecimento nas mais variadas áreas, sempre que é necessário partilhar informação (9).

Tabela 3- Exemplo de XML

```
<?xml version="1.0"?>
<note>
  <to>Maria</to>
  <from>Ricardo</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Sendo um subtipo da SGML, o XML foi criado com o propósito principal de facilitar a partilha de informação na internet. A forma como o XML comunica o seu significado dentro da WWW pode ser vista como análoga à forma como um formulário comunica o que cada campo significa ao seu leitor. Por exemplo, ao utilizar XML é assumido que tanto o emissor como o receptor têm o mesmo entendimento do que o campo *<heading></heading>* significa. Esta assumpção não tem que ser necessariamente feita quando se utiliza uma ontologia, pois o campo *<heading></heading>* pode ser explicitamente definido.

Tabela 4 - Formulário com a mesma informação que a Tabela 3

Note

To: Maria

From: Ricardo

Heading: Reminder

Body: Don't forget me this weekend!

Esta análise foi elaborada em [Kim 2002], onde é considerado que o XML é uma tecnologia mais madura que as ontologias para a Web Semântica, no que toca ao tamanho da comunidade de utilizadores, ferramentas de suporte e viabilidade de modelos de negócio. Por este facto, o autor do estudo, Henry Kim, considera que uma ontologia deve ser utilizada em situações onde a capacidade de representação semântica se sobrepõe às vantagens da maturidade do XML. Como corolário da sua análise, Kim afirma que em situações onde é importante reduzir a complexidade da partilha de informações o XML deve ser utilizado. Já no caso em que é importante reduzir a incerteza da informação a partilhar, uma linguagem de especificação de ontologias deve ser utilizada.

À data do artigo OWL não tinha sido ainda especificado, e somente em 2004 se tornaria uma recomendação W3C, pelo que a alternativa mais popular era o RDF.

Com a evolução da tecnologia da Web Semântica, que estava ainda no início em 2002, limitações da expressividade do RDF tornaram-se evidentes. De forma a suprimir essas

limitações a W3C iniciou em 2002 o desenvolvimento da linguagem OWL, que em 2004 se tornou recomendação para a Web Semântica. OWL tem por base o RDF, estendendo-o com novos recursos como reificação, suporte para inferência alargado e uma maior expressividade em geral. Em [I. Horrocks et al. 2003] é feita uma análise ao processo de especificação da linguagem que teve que obedecer ao paradigma da stack de linguagens da Web Semântica. Esta realidade obrigou a que alguns compromissos fossem feitos na criação do OWL, compromissos esses que visavam manter compatibilidades com outras linguagens existentes como SHOE, OIL e DAML-OIL. A linguagem devia evidenciar os seus pontos fortes em contraponto com as demais, de forma a ser facilmente adoptada.

Tabela 5 - Comparação de Capacidades de XML, RDF e OWL (10)

	XML(S)	RDF(S)	OWL 1	OWL 2
Listas Ligadas	NÃO	SIM	SIM	SIM
Restrições de Cardinalidade	SIM	NÃO	Parcial	SIM
Expressões de Classes	NÃO	NÃO	SIM	SIM
Tipos de Dados	SIM	SIM	SIM	SIM
Classes Definidas	NÃO	NÃO	SIM	SIM
Enumerações	SM	NÃO	SIM	SIM
Equivalência	NÃO	NÃO	SIM	SIM
Extensibilidade	NÃO	SIM	SIM	SIM
Semântica Formal	NÃO	SIM	SIM	SIM
Herança	NÃO	SIM	SIM	SIM
Inferência	NÃO	NÃO	SIM	SIM
Restrições Locais	NÃO	NÃO	SIM	SIM
Restrições Qualificadas	NÃO	NÃO	NÃO	SIM
Reificação	NÃO	SIM	SIM	SIM

OWL 1 foi dividida em três sub-linguagens, de forma a fornecer diferentes níveis de complexidade de acordo com as necessidades do utilizador. Cada sub-linguagem apresenta expressividade progressivamente maior. Os três diferentes perfis (designação utilizada na linguística) de OWL são (11):

- OWL Lite – Fornece suporte a utilizadores que precisam maioritariamente de uma hierarquia de classificações assim como capacidade de implementar restrições simples. Por exemplo, OWL Lite suporta limitações de cardinalidade de valor 0 ou 1. É em princípio mais fácil fornecer ferramentas de suporte para OWL Lite do que para os seus relativos mais expressivos, oferecendo uma rápida percurso de migração para enciclopédias e outras taxonomias.
- OWL DL – Fornece suporta a utilizadores que pretendem utilizar o máximo de expressividade possível sem perder completude computacional (todas as relações tem garantia de computação) assim como definibilidade (todas as computações terminam garantidamente em tempo finito) dos sistemas de inferência.
- OWL Full – Direccionado para utilizadores que precisam de expressividade máxima, assim como a liberdade sintáctica do RDF, perdendo no entanto garantias computacionais. Por exemplo, em OWL Full uma classe pode ser tratada simultaneamente como um conjunto de indivíduos e como um individuo. Outra

diferença importante de OWL Full é que uma `owl:DatatypeProperty` pode ser marcada como uma `owl:InverseFunctionalProperty`. OWL Full permite aumentar o significado pré-definido do vocabulário. Devido a estas características é improvável que um motor de inferência consiga suportar a globalidade de OWL Full.

Tabela 6 - Exemplo de Sintaxe OWL

```
<owl:Ontology rdf:about="http://www.example.org/wine">
  <rdfs:comment>An example OWL ontology</rdfs:comment>
  <owl:priorVersion rdf:resource="http://www.example.org/wine-
112102.owl"/>
  <owl:imports rdf:resource="http://www.example.org/food.owl"/>
  <rdfs:label>Wine Ontology</rdfs:label>
<owlx:Class owlx:name="Winery" owlx:complete="false" />
<owlx:Class owlx:name="Region" owlx:complete="false" />
<owlx:Class owlx:name="ConsumableThing" owlx:complete="false" />
</owl:Ontology>
```

Embora a experimentação prática com OWL 1 tenha sido bastante positiva ficou também claro que a linguagem tinha algumas limitações e poderia beneficiar de melhoramentos. Em [Grau et al. 2008] são identificadas limitações de OWL 1, tais como a falta de algumas construções que são geralmente necessárias na especificação de domínios complexos, restrições de cardinalidade qualificadas, expressividade relacional (expressividade das propriedades), expressividade dos tipos de dados, especificação de chaves primárias (keys), entre outras.

A especificação OWL 2 vem melhorar a expressividade de OWL, superando as limitações supramencionadas.

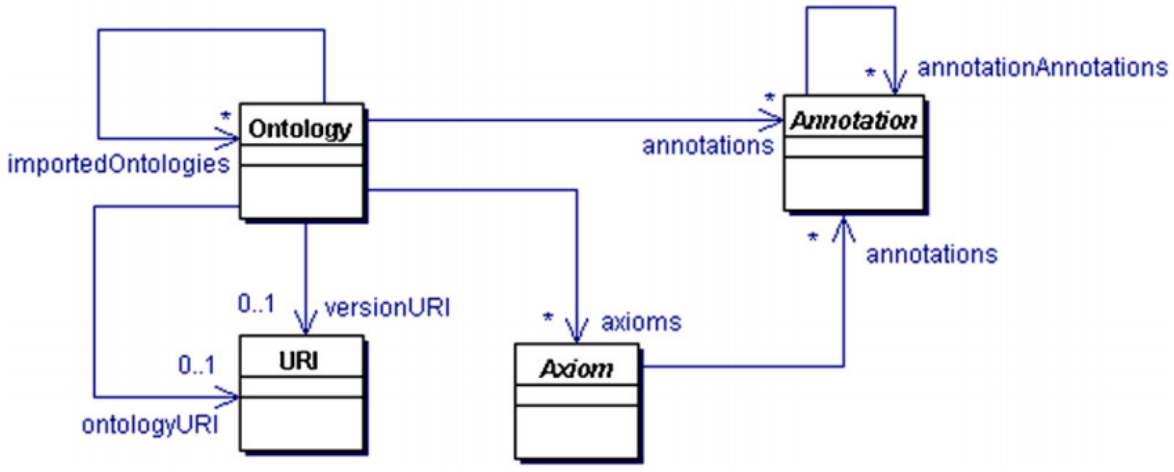

Imagem 9 - Modelo de Objectos de uma Ontologia OWL 2 [Grau et al. 2008]

De forma a fornecer subconjuntos da linguagem que melhor se adequam às várias necessidades da engenharia de ontologias, foram definidos três perfis diferentes de OWL 2.

- OWL 2 EL – Destinado a utilizadores que precisam de inferência eficiente sob um grande conjunto de terminologias. De forma a conseguir esta eficiência, algumas construções não são permitidas, como é o caso da negação, disjunção, cardinalidade e restrições AllValuesFrom.

- OWL 2 QL – Desenhada para conseguir inferência eficiente com grandes conjuntos de dados que estejam estruturados de acordo com esquemas relativamente simples. Não são permitidas disjunções nem restrições AllValueFrom, entre outras. São no entanto permitidas restrições de domínio e alcance de propriedades e restrições de pertença. Este perfil oferece a maioria das funcionalidades necessárias para captar modelos conceptuais, tal como diagramas UML e esquemas de base de dados.
- OWL 2 RL – Desenhado de forma que muitas das tarefas de inferência possam ser implementadas como um conjunto de regras dentro de um sistema de regras *forward-chaining*. Para garantir esta característica a expressividade deste perfil é mais reduzida que a expressividade de OWL 2. Embora permita a generalidade das construções de OWL 2, o perfil RL põe restrições na forma como estas construções podem ser utilizadas.

Para mais informação sobre os perfis OWL 2 assim como uma comparação de funcionalidades entre OWL e OWL 2 consultar os guias respectivos das linguagens no site da W3C, (11) e (12), assim como o artigo fonte desta análise, [Grau et al. 2008].

2.6 SPARQL – Uma Linguagem de Pesquisa de Ontologias

De forma a ver a Web Semântica como uma rede de dados (*web of data*), rapidamente se torna evidente a necessidade de uma linguagem de inquérito, análoga ao SQL existente para bases de dados tradicionais. Após esta análise de linguagens de especificação de ontologias falemos brevemente da linguagem de pesquisa de dados RDF, o SPARQL.

Sendo uma linguagem orientada a dados, SPARQL não oferece mecanismos de inferência. Fornece um potente motor de correspondências entre grafos cujas construções básicas são padrões de triplos. Durante a avaliação dos inquéritos, as variáveis contidas dentro dos padrões são correspondidas com a informação contida no repositório RDF. A sintaxe utilizada é similar à das linguagens SQL. SPARQL permite diferentes tipos de inquéritos, nomeadamente SELECT, ASK, CONSTRUCT e DESCRIBE. Permite também várias operações entre os padrões de triplos, via utilização dos operadores AND, FILTER, OPTIONAL e UNION. Na tabela 7 podemos ver um exemplo de um inquérito do tipo SELECT em SPARQL. Na tabela 8 podemos ver uma possível resposta a esse inquérito.

Tabela 7 - Inquérito SPARQL do tipo SELECT

```
SELECT ?x
WHERE { ?x  <http://www.w3.org/2001/vcard-rdf/3.0#FN>  "John Smith" }
```

Conforme exemplificado na tabela 7, uma variável em SPARQL começa por um ponto de interrogação '?' seguido de texto, sem espaços. Um padrão de triplos é representado por *{sujeito,predicado,objecto}* que que cada um dos elementos pode ser uma variável, um URI ou texto. Na tabela 7 está um exemplo das três situações. O que o motor SPARQL vai fazer é procurar dentro do repositório definido por todos os triplos que contenham como predicado um objecto identificado pelo URI `http://www.w3.org/2001/vcard-rdf/3.0#FN`, cujo valor do objecto seja "John Smith". Como no cabeçalho do SELECT apenas é pedido o

primeiro elemento do triplo, através da variável '`?x`', será devolvido apenas o sujeito de cada triplo encontrado.

Tabela 8 - Resultado em formato texto a um Inquérito SPARQL

```
---------------------------------
| x                             |
=================================
| <http://somewhere/JohnSmith/> |
---------------------------------
```

Tabela 9 - Dados em formato RDF utilizados como fonte para o Inquérito das Tabelas 7 e 8

```
@prefix vCard:   <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :        <#> .

<http://somewhere/MattJones/>
    vCard:FN    "Matt Jones" ;
    vCard:N     [ vCard:Family
                              "Jones" ;
                  vCard:Given
                              "Matthew"
                ] .

<http://somewhere/RebeccaSmith/>
    vCard:FN    "Becky Smith" ;
    vCard:N     [ vCard:Family
                              "Smith" ;
                  vCard:Given
                              "Rebecca"
                ] .

<http://somewhere/JohnSmith/>
    vCard:FN    "John Smith" ;
    vCard:N     [ vCard:Family
                              "Smith" ;
                  vCard:Given
                              "John"
                ] .

<http://somewhere/SarahJones/>
    vCard:FN    "Sarah Jones" ;
    vCard:N     [ vCard:Family
                              "Jones" ;
                  vCard:Given
                              "Sarah"
                ] .
```

Um estudo demonstrou que a complexidade do SPARQL é PSpace-Completa, devido ao operador OPTIONAL [Schmidt et al. 2010]. Nesse estudo são propostas algumas estratégias de optimização de inquéritos SPARQL.

2.6.1 SPARQL Endpoints

A grande utilidade de uma linguagem de pesquisa de informação na Web Semântica é permitir o acesso a informação de variadas fontes, utilizando um protocolo comum e estandardizado. O projecto Linking Open Data (13) tem como objectivo interligar vários repositórios RDF, e desde a sua criação o número de repositórios deste projecto tem crescido exponencialmente (14). Este crescimento é possível através da partilha de informação sob a forma de SPARQL Endpoints.

Um SPARQL *endpoint* trata-se de um URL que implementa o protocolo SPARQL através do protocolo de comunicações HTTP. Desta forma um repositório de triplos torna-se acessível a toda a *web* através de um protocolo *standard* e bem conhecido. Esta característica permite interligar dados de fontes distintas, aumentando a capacidade de resposta de um motor de pesquisa SPARQL.

Uma característica importante da linguagem SPARQL neste contexto é a capacidade de indicar dentro do inquérito qual a fonte de dados a pesquisar. Com o operador FROM é possível indicar ao motor SPARQL qual o *endpoint* a ser inquirido. A sintaxe de um pedido desse género é demonstrada na tabela 10.

Tabela 10 - Inquérito SPARQL com definição da fonte de dados

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?name
FROM    <http://example.org/foaf/aliceFoaf>
WHERE   { ?x foaf:name ?name }
```

2.6.2 Potencialidades dos SPARQL Endpoints

Um SPARQL Endpoint pode oferecer serviços apenas de pesquisa a um repositório de dados ou pode fornecer também a possibilidade de alterar informações no grafo do repositório. Como é evidente, não é aconselhável que um *endpoint* destinado a acesso público permita edição de informação do repositório, desta forma a W3C criou duas especificações distintas da linguagem de forma a facilitar esta divisão. A implementação que apenas permite inquéritos a um repositório denomina-se SPARQL Query. A implementação que permite actualizações ao repositório denomina-se SPARQL Update. É recomendado que um *endpoint* que implemente SPARQL Update esteja por trás de um mecanismo de autenticação, de forma a controlar quem efectua alterações aos dados.

2.6.3 Limitações dos SPARQL Endpoints

Embora a linguagem SPARQL permita definir a fonte de dados, não está estabelecido um mecanismo que permita cooperação entre *endpoints.* Conforme referido em [Görlitz & Staab 2011] a maioria das estratégias de optimização de inquéritos utilizadas em sistemas federados e distribuídos dependem da cooperação entre as diferentes bases de dados envolvidas (que no contexto deste documento, são SPARQL *endpoints*).

s	p	o
db:dbo.Objective/1	vocab:Objective_UIDPerspective	db:dbo.Perspective/4
db:dbo.Objective/3	vocab:Objective_UIDPerspective	db:dbo.Perspective/4
db:dbo.Objective/5	vocab:Objective_UIDPerspective	db:dbo.Perspective/2
db:dbo.Objective/6	vocab:Objective_UIDPerspective	db:dbo.Perspective/2
db:dbo.Objective/1	vocab:Objective_description	"Increase Satisfaction"
db:dbo.Objective/3	vocab:Objective_description	"Reduce Costs"
db:dbo.Objective/5	vocab:Objective_description	"Reduce Costs"
db:dbo.Objective/6	vocab:Objective_description	"Increase Revenue"
db:dbo.Objective/1	rdfs:label	"dbo.Objective #1"
db:dbo.Objective/3	rdfs:label	"dbo.Objective #3"

Imagem 10 - Exemplo de SPARQL Endpoint que fornece Interface Gráfica em Browser

Uma característica importante que uma *web of data* deve ter é a capacidade de manter informação sobre a proveniência dos dados que utiliza. Diferentes fontes de dados têm diferentes níveis de credibilidade, e quando duas fontes de dados contêm factos contraditórios o utilizador pode desejar aceitar a fonte com mais credibilidade do seu ponto de vista. Uma limitação actual da linguagem SPARQL e por consequência da maioria dos SPARQL *endpoints* é que não prevê informação de proveniência. Existem no entanto algumas implementações de *endpoints* que adicionam esse recurso, como é o caso da ferramenta Pubby (15) que pode ser configurada para servir de *frontend* de um SPARQL *endpoint*, fornecendo informação sobre a proveniência dos resultados servidos.

2.7 Motores de Pesquisa Semânticos actuais

De forma a entender melhor que soluções práticas de motores de pesquisa semânticos estão actualmente disponíveis foi considerado útil efectuar uma pesquisa e análise de algumas aplicações de pesquisa de informação na *web* que se auto-intitulam como Motores de Pesquisa Semânticos. Foram escolhidas cinco aplicações, Cognition, PowerSet, Hakia, DeepDyve e SenseBot. Foi feito um estudo prático, analisando que funcionalidades disponibilizam ao público e quando possível foi feita uma análise à arquitectura interna de funcionamento. Posteriormente foi feito um resumo comparativo entre as cinco aplicações.

Este estudo não teve em atenção o formato de armazenamento dos dados, teve apenas atenção aos mecanismos utilizados para extrair informação e a interpretar. Assim, embora esta dissertação seja focalizada em repositórios de dados em formato RDF, neste estudo prático foram considerados sistemas que armazenam dados em várias formas.

Em [Ilyas et al. 2005] o autor considera que os componentes básicos de um motor de pesquisa semântico são: *Ontology Editor*, *Ontology Mapper*, *Ontology Translator*, *Web page annotator*, *Ontology Crawler*, *Web Crawler*, *Query Builder*, *Knowledge Base* e *Inference Engine*.

Neste estudo tentou-se perceber se esta consideração é válida para implementações actuais, e que componentes da arquitectura dos sistemas analisados tem a mesma responsabilidade que os componentes supramencionados. De um ponto de vista do

utilizador foram analisadas várias características que estão sintetizadas nas tabelas do Anexo B desta dissertação.

2.7.1 Cognition

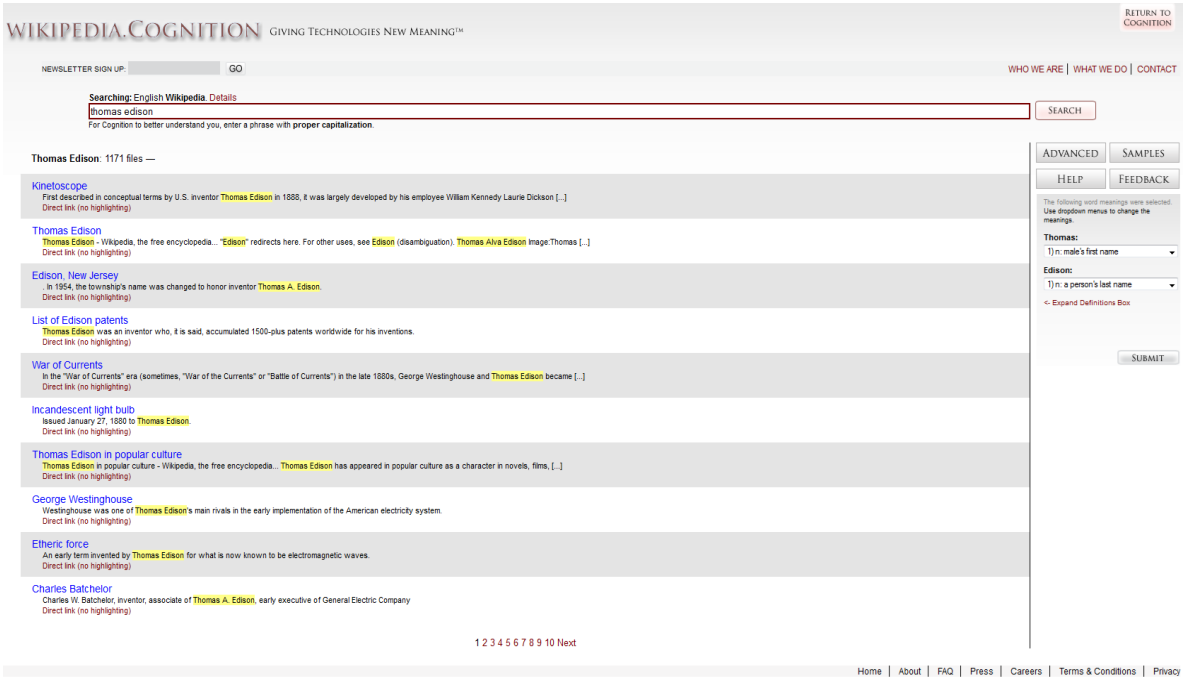

Imagem 11 - Screenshot do Cognition

Na literatura disponibilizada no site do Cognition é apresentada uma imagem que representa os componentes principais da arquitectura do sistema. A imagem 12 foi extraída do documento [Dahlgren 2007].

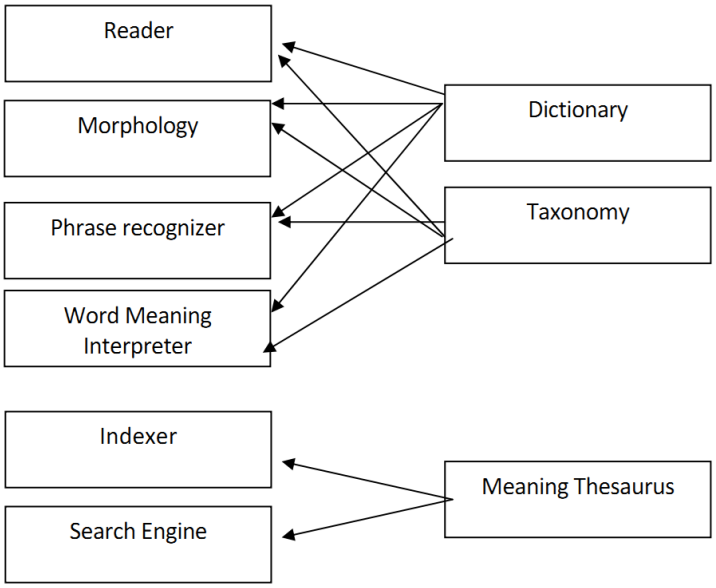

Imagem 12 - Constituintes da Arquitectura do Cognition

Pela documentação percebe-se que o Cognition utiliza uma solução mista, onde utiliza ontologias para definir conceitos e significados, mas tira bastante proveito do processamento de linguagem natural (NLP). De acordo com testes efectuados no estudo [Cognition Technologies 2008] esta abordagem mista permite obter resultados com maior relevância que um dos motores analisados, o PowerSet.

2.7.2 PowerSet

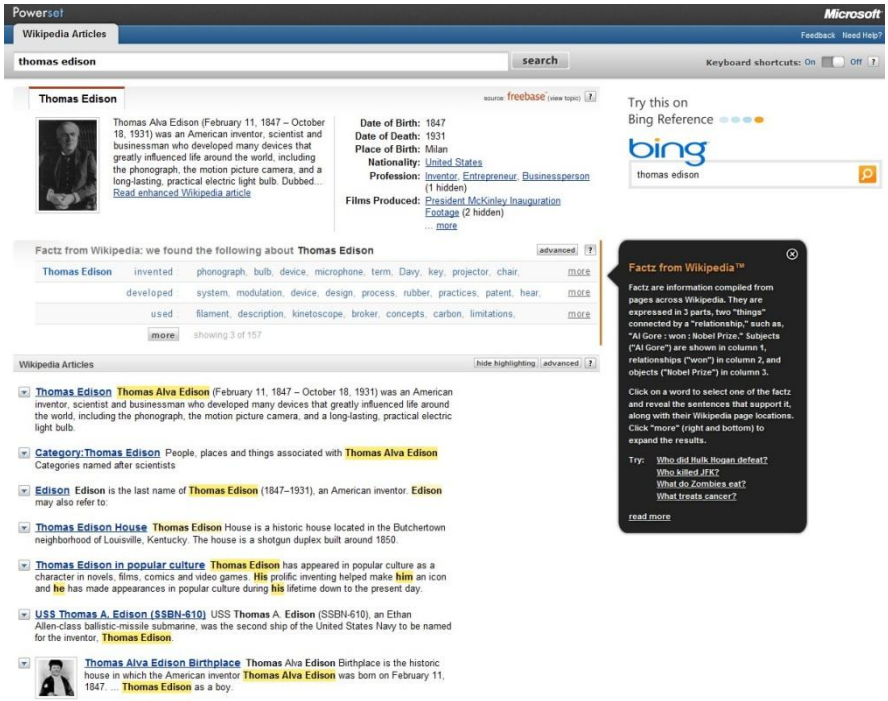

Imagem 13 - Screenshot do Powerset

O PowerSet trata-se de um motor de pesquisa que utiliza processamento de linguagem natural de forma a determinar o contexto do inquérito. Posteriormente percorre o seu repositório de informação de forma a encontrar correspondências com o que foi inquirido. De acordo com a bibliografia disponível, não se trata de um motor de pesquisa generalista e é mais direccionado para trabalhar com pequenas quantidades de dados estruturados. Embora não tenha sido possível obter uma descrição da arquitectura do sistema, é sabido que o PowerSet, antes de ser adquirido pela empresa Microsoft, utilizava Amazon Web Services (16) para armazenar os seus índices, pelo que pode ser admitido que utilizava uma arquitectura distribuída.

Uma pesquisa posterior (17) sugere que o PowerSet inspecciona a estrutura HTML de enciclopédias *online* tais como Wikipedia e indexa a informação nela contida. Posteriormente são aplicadas técnicas de NLP aos dados indexados.

No artigo (18) é revelado que o PowerSet utiliza um método de indexação baseado em índices invertidos, à semelhança do popular Google[7]. No entanto apoia este índice com algoritmos de processamento de linguagem natural e grande poder computacional.

2.7.3 Hakia

Imagem 14 - Screenshot do Hakia

O Hakia divide os resultados de cada inquérito em separadores, denominados internamente como Galerias de Informação. Este processo de divisão é semiautomático, consistindo em catalogação automática baseada em tecnologias do conhecimento e processamento editorial realizado por especialistas. O motor de pesquisa é o primeiro dos analisados a

[7] http://www.google.pt/

utilizar explicitamente o conceito de ontologia, embora na documentação seja referido que ao ontologias utilizadas foram desenvolvidas com o propósito de permitir ao sistema boa performance de indexação e pesquisa. Não têm o intuito de serem partilhadas com o exterior.

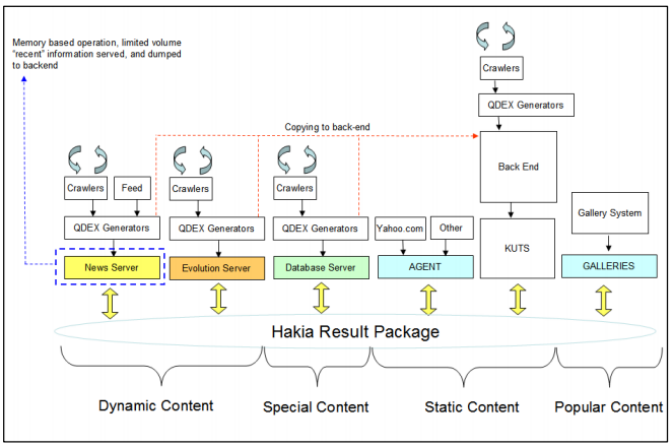

Imagem 15 - Arquitectura do Hakia

No lugar de um índice invertido o projecto Hakia utiliza uma tecnologia de indexação proprietária denominada QDEX. O QDEX analisa o conteúdo de páginas *web* e calcula todas as pesquisas que podem ser efectuadas aquele conteúdo. Este processo é realizado em modo *offline*, antes do sistema receber algum inquérito. De acordo com (19) esta característica faz com que o sistema gere um índice onde a cada inquérito possível corresponde um grupo restrito de palavras de um documento, tornando o tamanho de cada grupo activo pequeno. Desta forma o sistema consegue realizar associações semânticas em tempo real, utilizando uma fracção do poder computacional necessário para efectuar a mesma tarefa com índices invertidos.

Para além de ontologias e a tecnologia QDEX o sistema utiliza um algoritmo proprietário para efectuar a pontuação de cada resultado. Este algoritmo é denominado SemanticRank.

De acordo com [Hakia Inc 2010] o algoritmo SemanticRank trata-se de um módulo independente do sistema principal que refina os resultados encontrando as frases mais de um parágrafo que melhor se adequam ao inquérito efectuado. Este processo recorre a processamento sintáctico, ontológico e morfológico [Hakia Inc 2010].

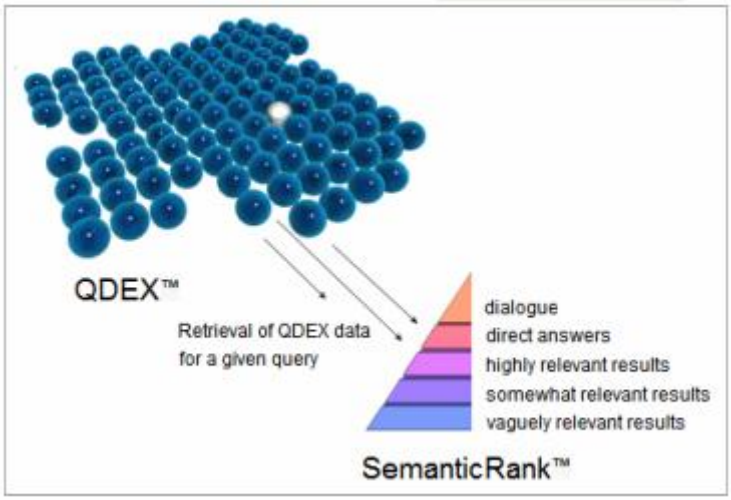

Imagem 16 - Diagrama do Algoritmo SemanticRank [Hakia Inc 2010]

2.7.4 DeepDyve

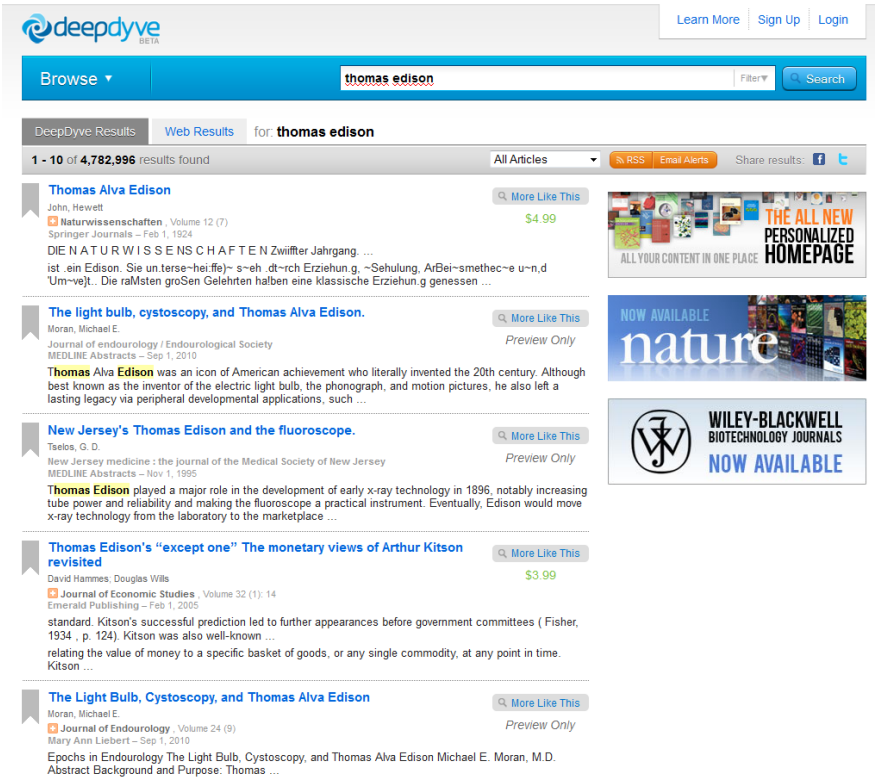

Imagem 17 - Screenshot do DeepDyve

Embora chamado de Motor de Pesquisa Semântico em alguns artigos (20) o DeepDyve não é um motor de pesquisa semântico no sentido abordado nesta dissertação. Este sistema permite efectuar inquéritos bastante longos, de até 25 000 caracteres, oferecendo um catálogo de jornais, revistas e artigos científicos, assim como bases de dados. Permite filtrar

resultados, refina-los e dispô-los de acordo com vários critérios. No entanto, para obter meta-informação sobre os dados no seu catálogo serve-se da estrutura dos artigos e bases de dados em questão. Trata-se portanto de um motor de pesquisa semântico estático. Se a dado momento a estrutura de um artigo for alterada, ou o esquema de uma das bases de dados oferecidas mudar, o sistema precisará ser actualizado manualmente antes de suportar a nova informação.

Por não fazer uso de ontologias ou processamento de linguagem natural, as suas capacidades semânticas são bastante mais limitadas que os demais motores de pesquisa.

A pouca documentação disponível *online*[8], informa que o DeepDyve não utiliza lógica Booleana ou taxonomias na sua indexação. Não foi no entanto possível descobrir mais factos sobre a sua arquitectura interna, pelo que o estudo deste sistema é apenas superficial.

2.7.5 SenseBot

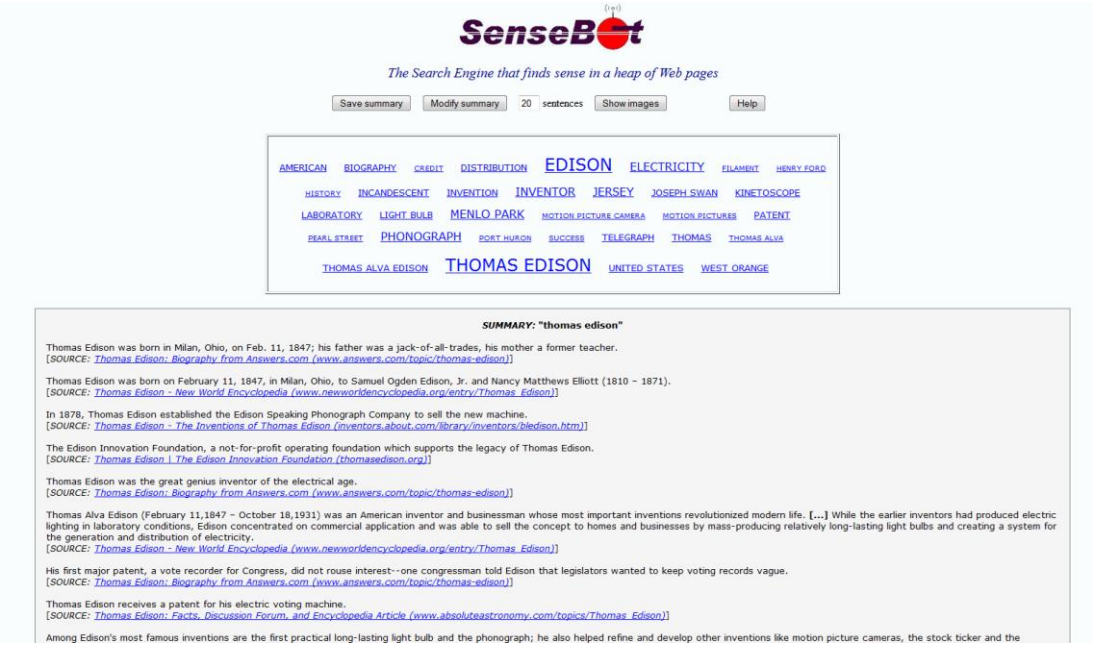

Imagem 18 - Screenshot do SenseBot

O SenseBot utiliza o conceito de *sumário* para apresentar os resultados aos seus utilizadores. Um sumário contém um resumo dos pontos mais importantes do inquérito que foi realizado.

Como fonte de informação o SenseBot recorre a motores de pesquisa externos, como por exemplo o Google ou Yahoo. O utilizador pode escolher que motor de pesquisa utilizar antes de efectuar o inquérito. Posteriormente, utilizando técnicas de *data mining* o motor extrai os conceitos chave de cada fonte para o inquérito em questão. Esses conceitos são mostrados ao utilizador numa nuvem de *tags*, localizada no topo da página, conforme pode ser visto na imagem 18. A cada conceito extraído é atribuído um peso que depois vai determinar o que é incluído no resumo (21).

[8] http://www.deepdyve.com

Os resultados do SenseBot serão mais precisos se os documentos devolvidos pelo motor de pesquisa simples estiverem realmente relacionados com o inquérito. Por este motivo o SenseBot está dependente da qualidade dos resultados devolvidos.

Para se tornar auto-suficiente, a partir de 2010 o SenseBot oferece também a possibilidade de escolher um motor de pesquisa interno como fonte de dados. O método de indexação e pesquisa da informação deste motor não foi disponibilizado.

Este motor de pesquisa semântico é mais indicado para procurar rapidamente definições ou termos científicos e também para conseguir em pouco tempo ter uma ideia geral de um tema.

2.7.6 Capacidades

Os cinco sistemas estudados conseguem responder bem a inquéritos do tipo *long tail,* no entanto no que toca ao aproveitamento das capacidades de ontologias e linguagens de especificação de conhecimento apenas um, o Hakia faz uso directo destas vantagens.

Devido a estes seres sistemas comerciais, detalhes mais específicos sobre a arquitectura das soluções utilizadas foram difíceis de obter. No caso do PowerSet o facto de durante a elaboração desta dissertação este ter sido adquirido pela empresa Microsoft dificultou o seu estudo.

A característica do SenseBot de produzir resumos de um dado tema é bastante interessante, no entanto testes práticos mostraram que este tipo de abordagem pode tornar a leitura dos resultados bastante confusa com determinados inquéritos.

2.7.7 Limitações

Os sistemas analisados baseiam-se na sua maioria em técnicas de processamento de linguagem natural, não tirando partido de ontologias para fornecer capacidades de processamento de inquéritos complexos.

Não são utilizados *standards* da Web Semântica, pelo que cada sistema comporta-se como um sistema fechado que não poder ser explorado por outros sistemas. Esta propriedade vai contra o conceito de Web Semântica e *Web of Data*.

O Cognition exige que o utilizador seleccione primeiro a área do conhecimento a pesquisar.

Todos os sistemas foram projectados para inglês, utilizadores de outras línguas, nomeadamente Português, obtêm resultados de qualidade bastante inferior.

Embora considerado um motor de pesquisa semântico, o DeepDyve não apresenta flexibilidade suficiente. Baseia-se na estrutura dos artigos que indexa e não foram encontradas evidências de utilização de tecnologias semânticas.

Durante este estudo foram elaboradas tabelas comparativas de resumo das várias funcionalidades dos cinco motores de pesquisa semânticos estudados. Estas tabelas podem ser consultadas no Anexo B desta dissertação.

2.8 Arquitecturas de um Motor de Pesquisa Semântico:

A infra-estrutura de um sistema de indexação e pesquisa de informação semântica pode ter várias combinações de características. No que se refere ao armazenamento de dados, este pode ser centralizado ou distribuído, o sistema de índice pode também ser centralizado ou distribuído e no que toca à cooperação entre fontes de dados, estas podem cooperar entre si para responder a um inquérito ou actuarem de forma completamente independente [Görlitz & Staab 2011]. No entanto, algumas das combinações não fazem sentido, sendo que podemos identificar as que restam:

- Repositório Central
  Os dados em formato RDF podem ser recolhidos de diversas fontes, utilizando *crawlers* ou descarregando-os de uma só vez de uma fonte através de *data-dumps*
- Federação
  Numa federação, a integração de fontes de dados distribuídas é obtida através de um mediador centralizado. O mediador mantém um índice global com informação estatística que é utilizada no mapeamento entre *queries* e fontes de dados, e também para optimização. A existência de cooperação entre fontes de dados permite reduzir o custo de processamento de pesquisas por parte do mediador.
- Gestão Peer-to-peer
  Dados RDF e dados sobre o índice podem ser geridos de uma forma distribuída se todas as fontes de dados cooperarem entre si. O resultado neste caso é uma rede Peer-to-peer onde todas as fontes de dados partilham a responsabilidade de gestão dos dados RDF e índice. Por este motivo, não é necessário um mediador e o armazenamento de dados e custos de processamento podem ser balanceados por todas as fontes de dados envolvidas. Sem cooperação das fontes de dados não é possível implementar um índice distribuído.

Tabela 11 - Combinação de Características de diferentes tipos de Infra-estruturas [Görlitz & Staab 2011]

<table>
<tr><td></td><td colspan="2">Central Data Storage</td><td colspan="2">Distributed Data Storage</td></tr>
<tr><td>Independent Data Sources</td><td>n/a</td><td rowspan="2">Central Repository</td><td rowspan="2">Federation</td><td>n/a</td></tr>
<tr><td>Cooperative Data Sources</td><td>n/a</td><td>P2P Data Management</td></tr>
<tr><td></td><td>Distr. Index</td><td colspan="2">Central Index</td><td>Distr. Index</td></tr>
</table>

Tendo em conta estas características, em [Görlitz & Staab 2011] são identificadas três variações de arquitecturas:

2.8.1 Sistema Centralizado

São as abordagens mais encontradas em soluções de pesquisa de triplos RDF. Todos os *datasets* são percorridos e a informação é guardada num repositório central. Esta abordagem tem como vantagem o facto de serem normalmente utilizadas estruturas de índices optimizadas, criadas localmente que permitem uma resposta eficiente a inquéritos. No entanto, as cópias locais assim como os dados do índice precisam ser actualizados constantemente, caso contrário os resultados podem ficar desactualizados ou inconsistentes.

2.8.2 Processamento Explorativo de Inquéritos

Esta arquitectura é baseada no princípio de navegação dos dados. Um inquérito é inicialmente avaliado sob um *dataset* de forma a encontrar correspondências entre entidades nos dados assim como ligações interessantes para outros *datasets*. Estes, podem conter mais entidades que satisfação o inquérito. De forma iterativa as ligações são resolvidas e os dados descobertos são fornecidos ao sistema sob a forma de um *stream*, assim que se tornam disponíveis. A pesquisa termina quando não existem mais ligações a serem exploradas que ofereçam potenciais resultados. Esta abordagem avalia o inquérito directamente sob a LOD e não requer cópias dos dados ou estruturas de índice adicionais. Isto no entanto implica que haja uma comunicação maior no sistema de forma a processar todas as ligações de interesse que apontam para outras fontes de dados. Além destes aspectos, o ponto inicial de procura influencia significativamente a qualidade dos resultados.

2.8.3 Sistema Federado

Uma arquitectura federada oferece um bom nível de flexibilidade e escalabilidade no que toca à pesquisa de LOD. No entanto existem bastantes diferenças em relação a bases de dados federadas e distribuídas. As fontes de dados LOD estão fracamente ligadas entre si e são normalmente controladas por partes independentes, o que significa que o esquema de dados difere de repositório para repositório e também que os dados em bruto podem não ser directamente acessíveis. Assim, os requisitos base para uma federação de dados são que todas as fontes forneçam uma implementação de uma linguagem de pesquisa *standard* (SPARQL), de forma a permitir a extracção dos dados RDF pretendidos. Em adição a esta característica assume-se que cada fonte de dados LOD fornece estatísticas como o número de ocorrências de um termo dentro do *dataset*, que são utilizadas para identificar fontes de dados adequadas para um inquérito, e para optimizar a execução de um inquérito. A responsabilidade da federação é a de manter as estatísticas sobre os dados num *índice de federação* e coordenar a interacção com as fontes de dados.

2.8.4 Limitações das Propostas actuais

A melhor forma de exploração do potencial da LOD é através de inquéritos complexos efectuados a diferentes fontes de dados, estes inquéritos permitem aproveitar todo o potencial contido dentro da Linked Data. No entanto conforme foi analisado neste capitulo, inquéritos complexos necessitam de um mecanismo de cooperação entre fontes de dados, mecanismo esse que não é actualmente suportado pela linguagem recomendada para pesquisas da LOD, o SPARQL. Por este mesmo motivo, a optimização de inquéritos à LOD não pode ser feita da mesma forma que nas tradicionais bases de dados. A arquitectura proposta por [Görlitz & Staab 2011] não propõe uma solução de cooperação entre fontes de dados.

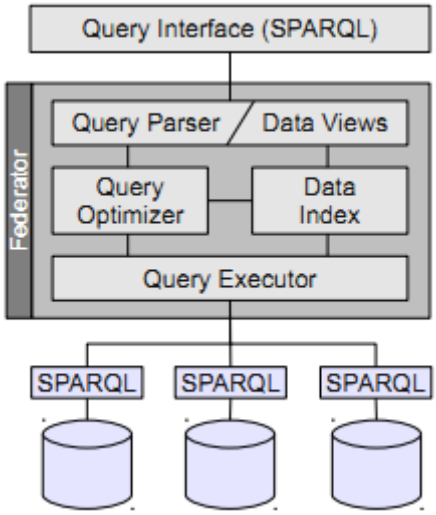

Imagem 19- Arquitectura de uma LOD Federation segundo [Görlitz & Staab 2011]

Embora a navegação por *links* seja uma forma natural de explorar a LOD, não há garantias de encontrar a informação pretendida num dado número de passos. Por contraste, uma pesquisa baseada em inquéritos complexos oferece maior flexibilidade to que diz respeito a especificar propriedades e relações entre entidades RDF pretendidas [Görlitz & Staab 2011]. No entanto este género de pesquisa é baseado em álgebra Booleana, encontrando apenas correspondências exactas. Por este motivo é necessário que os utilizadores de tal sistema possuam um bom conhecimento sobre a estrutura interna dos dados, o que para um conjunto de dados tão grande como a LOD é incomportável.

Se as pesquisas foram feitas utilizando um vocabulário pré-estabelecido e posteriormente o processador generalizar o inquérito, realizar inferência e aproximações assim como outras técnicas, a qualidade e quantidade dos resultados será melhorada. Desta forma consegue-se a flexibilidade que um inquérito complexo consegue fornecer, aliada à abstracção dos modelos de dados das fontes utilizadas [Görlitz & Staab 2011].

3 Proposta de Arquitectura Federada com Cooperação entre Fontes de Dados.

3.1 Objectivos

Segue-se uma descrição dos objectivos desta proposta de arquitectura.

3.1.1 Extracção de Dados de Múltiplas Fontes RDF

De forma a respeitar a filosofia por trás da Web Semântica, os dados utilizados por um Motor de Pesquisa Semântico devem estar acessíveis na web [T. Berners-Lee et al. 2001]. De forma a serem referenciáveis, devem estar unicamente identificados através de URIs.

A natureza distribuída da web dita que a melhor solução para aproveitamento destes dados seja também distribuída, por ser mais dinâmica que uma solução centralizada que necessita de actualizações mais frequentes. Apesar deste facto, de forma a garantir níveis de performance aceitáveis, um sistema de pesquisa deve manter um catálogo de dados [Görlitz & Staab 2011]. Nesta proposta, o catálogo de dados do sistema é composto por índices de texto, índices RDF e ficheiros de mapeamentos.

Segue-se uma explicação detalhada dos problemas que esta proposta pretende resolver:

3.1.2 Extracção de dados de Múltiplos Repositórios

Para explorar todas as capacidades da LOD, um Motor de Pesquisa Semântico deve ser flexível o suficiente para conseguir inquirir todos os repositórios RDF que fazem parte do sistema. Isto significa que, a menos que todos os repositórios partilhem o mesmo esquema de dados, deve existir algum tipo de negociação automática entre eles ou têm que existir mapeamentos entre os diferentes esquemas. Generalização e adaptação de inquéritos deve ocorrer num momento anterior à execução do inquérito [Görlitz & Staab 2011]. Para além deste facto, os resultados provenientes dos diversos repositórios vão conter em alguns casos redundâncias e discrepâncias que têm que ser resolvidas por um processo de agregação, antes de ser possível apresentar resultados homogéneos ao utilizador.

3.1.3 Melhorar Performance

A natureza descentralizada da LOD implica que um Motor de Pesquisa Semântico não seja capaz de garantir um tempo mínimo de resposta a um inquérito. O tamanho dos dados torna impraticável carregar em memória todo o grafo de informação. Um sistema de pesquisa federado deve, sempre que possível, balancear o custo de processamento de um inquérito pelos seus vários nós, efectuando decisões em tempo-real sobre quais os repositórios a serem contactados, que resultados podem ser extraídos em determinado nó e que resultados devem ser reencaminhados para outros nós para processamento adicional, ou enviados directamente para um módulo de agregação. Estas tarefas são efectuadas nos componentes de reescrita do inquérito, optimização do inquérito e geração do plano de execução. Este processo precisa comunicar com um índice dedicado a este efeito, assim

como uma estrutura de mapeamento, de forma a conseguir um plano de execução optimizado.

3.1.4 Manter/Obter Escalabilidade

O sistema deve ser modular o suficiente para permitir um fácil crescimento à medida que novos repositórios RDF são adicionados à LOD. Deve escalar bem à medida que os dados aumentam e manter performance mesmo com os índices a crescerem durante a indexação de novos repositórios.

3.1.5 Manter Informação de Proveniência dos Dados

Informação sobre a origem dos dados deve ser preservada à medida que estes viajam pelos vários nós do sistema, para fins estatísticos mas principalmente para gestão de confiança e de forma a auxiliar o motor de inferência. A informação deve manter-se ligada ao repositório de onde originou [Görlitz & Staab 2011].

3.2 Descrição da Arquitectura

3.2.1 Vista Geral

Na Imagem 19 pode ser visualizada uma vista geral de todos os componentes que fazem parte da arquitectura proposta nesta dissertação. Os componentes encontram-se divididos em unidades lógicas. Cada uma destas unidades lógicas tem um papel bem definido e comunica com as demais através de um interface que permite um fácil desenvolvimento de diferentes módulos e uma rápida troca por módulos com responsabilidades idênticas. A única regra que um módulo deve respeitar é implementar o interface de comunicação estabelecido.

As divisões lógicas são as seguintes:

- Interface com Utilizador (User Interface)
- Processamento de Inquéritos (Query Processing)
- Gestão de Indexação e Mapeamento (Indexing and Mapping Management)
- Acesso a Repositórios de Dados (Data Repository Access)
- Agregação de Resultados (Result Set Aggregation)

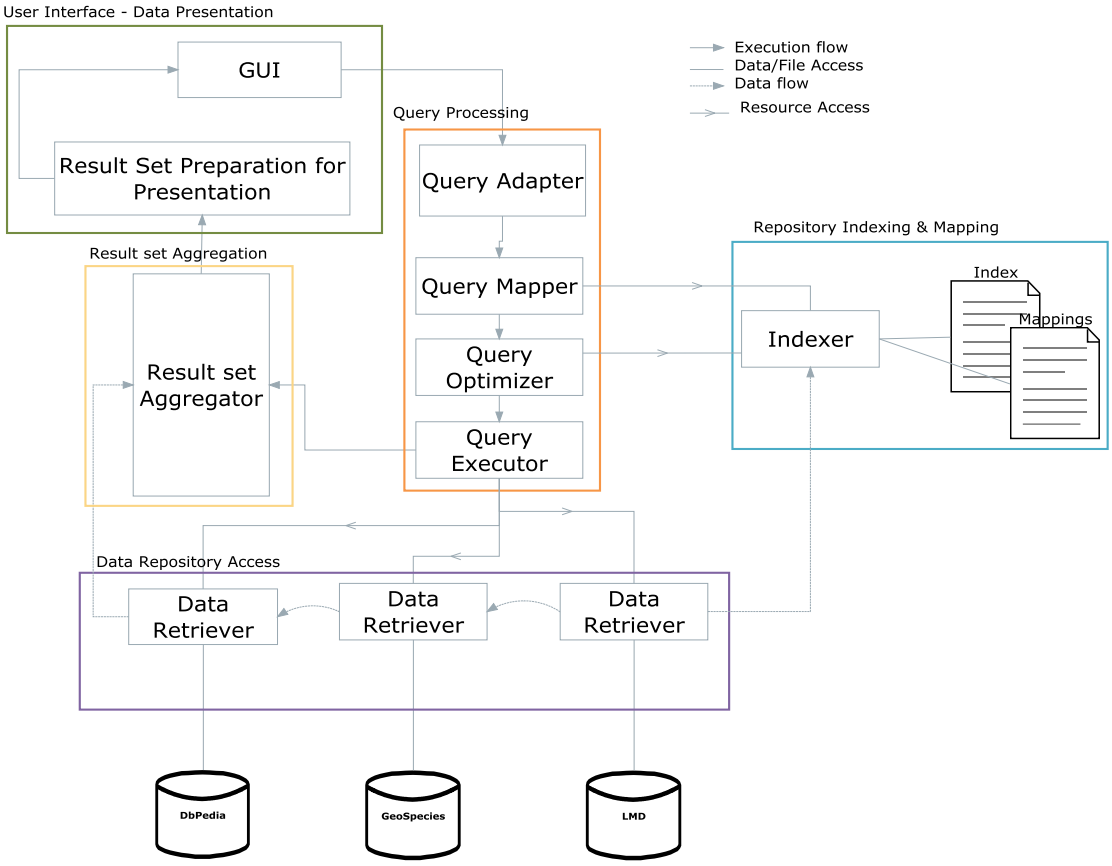

Imagem 20 – Diagrama Geral da Arquitectura Proposta

Nas páginas seguintes segue-se a descrição detalhada de cada componente.

3.2.2 GUI – Interface Gráfico

O componente de Interface Gráfico permite ao utilizador seleccionar a ontologia sob a qual o inquérito vai ser realizado, assim como a escrita do próprio inquérito. Após receber os resultados do inquérito trata de os apresentar ao utilizador, de acordo com o tipo de dados. A apresentação dos dados varia caso sejam dados estatísticos, documentos ou ligações externas.

A escrita do inquérito pode ser em linguagem formal, utilizando SPARQL e seleccionando que repositórios devem ser consultados ou em alternativa qual a área do conhecimento pretendida, de entre as áreas presentes na ontologia de destino.

Embora não seja previsto nesta proposta de arquitectura, este componente pode providenciar meios de autenticação, através de *login* e *password* ou utilizando tecnologias mais sofisticadas como FOAF+SSL [Story et al. 2009]. A utilização de perfis FOAF permitiria definir listas de permissões de acesso a determinados documentos RDF, assim como a gestão de estatísticas de utilização que poderiam ser úteis na determinação automática do contexto do inquérito, sem necessidade de definição explícita por parte do utilizador.

3.2.3 Query Processing – Processamento de Inquéritos

3.2.3.1 Query Adapter

A responsabilidade do componente Query Adapter é interpretar a linguagem utilizada na construção do inquérito pelo utilizador, transformando-a numa linguagem formal, como o SPARQL, caso o utilizador tenha efectuado um inquérito em linguagem natural. É também responsável por aplicar correcções gramaticais e técnicas de relaxamento e generalização de inquéritos, recorrendo a algoritmos de *stemming* como por exemplo Snowball[9].

O Query Adapter encontra termos similares e efectua a correspondência entre estes e um determinado contexto.

3.2.3.2 Query Mapper

O componente Query Mapper estabelece relações entre os conceitos utilizados no inquérito submetido e os conceitos presentes nas ontologias dos repositórios de dados que se encontram definidos no catálogo do sistema. Para este efeito, este componente recorre aos mapeamentos disponibilizados no componente Query Indexer de forma a encontrar classes correspondentes (por exemplo, um repositório que utilize uma ontologia onde está definida a classe 'Pessoa' pode ser utilizado para responder a inquéritos sob uma ontologia que define a classe "Person", se estiver definido um mapeamento no sistema entre 'Pessoa' e 'Person').

3.2.3.3 Query Indexer

O componente Query Indexer tem a responsabilidade de construir e gerir o índice de documentos RDF assim como os mapeamentos existentes entre conceitos de diferentes repositórios. Quando a informação de um repositório é alterada, este componente deve actualizar o índice correspondente e verificar se o esquema de dados se mantem. Caso os mapeamentos no sistema precisem de actualizações, o sistema deve alertar para esse facto. Nesta arquitectura é pressuposto que os mapeamentos entre conceitos são feitos de forma manual, em modo *offline*. O sistema apenas se encarrega de os consultar, utilizando-os para estabelecer correspondências entre repositórios RDF.

O Query Indexer fornece um interface pelo qual outros componentes do sistema acedem aos seus serviços, trabalhando de forma independente dos outros módulos. Nesta arquitectura, este interface é utilizado pelos componentes Query Mapper e Querry Optimizer.

[9] http://snowball.tartarus.org/

3.2.3.4 Detalhes:

Embora nesta descrição o Query Indexer seja representado como um único componente, a sua implementação prática é composta por vários componentes. Nessa descrição é focalizada a arquitectura geral da aplicação, sendo que detalhes sobre a estrutura do índice e mapeamentos e que tecnologias/ferramentas são utilizadas na sua implementação foram deixadas para o capítulo seguinte.

3.2.4 Query Plan Optimizer

De acordo com [Görlitz & Staab 2011], a responsabilidade de um optimizador de inquéritos é a de elaborar um plano de execução do inquérito que permita a melhor performance possível do sistema. Este componente recebe o *output* vindo do componente Query Mapper e decide quais os repositórios capazes de responder ao inquérito completo ou a partes do inquérito. Se for possível responder ao inquérito através de um só repositório, este é enviado para esse repositório apenas. Se for necessário dividir o inquérito por um conjunto de repositórios, este componente decide que estratégia de agregação de resultados vai ser utilizada. O plano de execução é criado tendo em conta que repositórios fornecerão resultados intermédios e qual a ordem de comunicação entre repositórios. Este plano estratégico é serializado e enviado para os vários repositórios. O plano define ainda se resultados intermédios devem ser enviados directamente para o componente Query Aggregator de forma a apresentar mais rapidamente algum feedback ao utilizador (estratégia utilizada quando o tempo de resposta é elevado).

3.2.5 Query Executor

O componente Query Executor recebe o plano de execução de inquérito gerado no Query Plan Optimizer e executa-o, enviando-o juntamente com a *query* formal já processada para os repositórios RDF relevantes.

Relativamente à localização dos repositórios RDF, existem três cenários possíveis:

3.2.5.1 Dados armazenados Localmente

Os documentos estão armazenados localmente no sistema. Esta arquitectura não prevê este tipo de cenário.

3.2.5.2 Dados armazenados em Memoria

Os documentos estão armazenados na memória principal do sistema. Este cenário apenas faria sentido para pequenos conjuntos de dados. Devido ao tamanho da memória necessária para manter todos os dados presentes numa imagem da LOD este cenário não é abordado nesta arquitectura.

3.2.5.3 Dados armazenados Remotamente

Os documentos estão armazenados remotamente e o sistema acede aos mesmos através de *endpoints* (SPARQL Endpoints) que permitem acesso de leitura aos dados em formato RDF. Os resultados originam de vários repositórios e devem ser agregados antes de estarem prontos para apresentação ao utilizador. Este é o cenário considerado nesta proposta de arquitectura.

3.2.6 Data Retriever

Conforme referido anteriormente nesta dissertação, a linguagem SPARQL não permite utilizar mecanismos de federação de forma a garantir cooperação entre *endpoints*. De forma a superar esta limitação foi introduzido o componente Data Retriever que trata de receber o plano de execução e *query* formal e o processa junto a um determinado SPARQL Endpoint. O componente Data Retriever pode também actuar como um wrapper para a linguagem de pesquisa utilizada. É este componente que procede à extracção dos triplos relevantes do repositório de dados RDF a ele associado.

Para cada SPARQL Endpoint registado no sistema, existe um Data Retriever configurado para o aceder. Este Data Retriever funciona de forma independente do sistema e é responsável também por notificar o sistema central de alterações aos dados ou ao esquema dos dados.

3.2.6.1 Parametrização

Cada Data Retriever pode ser parametrizado no que toca ao repositório de dados que acede, que informação extraí desse repositório, se é utilizado um motor de inferência para processar a informação extraída e ainda que informação de proveniência deve ser adicionada, caso o *endpoint* não a forneça de origem.

3.2.6.2 Suporte de Serialização

Os dados extraídos dos repositórios devem ser serializador para outros Data Retrievers ou para o componente de agregação. Para tal deve ser utilizada uma linguagem que permita serialização de dados.

3.2.7 Componente Aggregator

O componente Aggregator recebe o conjunto de resultados proveniente dos diversos Data Retrievers e procede à integração de informação de forma a obter um conjunto de

resultados homogéneo e coerente. Esta operação encarrega-se de eliminar factos duplicados e resolver incoerências e também de utilizar o algoritmo de integração de informação de forma a encontrar equivalências entre instâncias. Em adição a esta funcionalidade, o agregador utiliza um algoritmo de *ranking* de resultados de forma a apresentar os resultados com maior relevância em primeiro lugar ao utilizador.

3.2.8 Duas abordagens diferentes:

A. Os factos são enviados de Data Retrievers para Data Retriever assim que ficam disponíveis. Esta abordagem implica que a agregação é realizada de uma forma serializada. O plano de execução deve calcular a ordem pela qual a informação passa de nó em nó, e cada nó sabe o seu lugar no *pipeline* daí proveniente (sabe de que nó irá receber informação e para que nó a reencaminhará após processamento dos dados). Após receber esta informação, cada nó extrai a informação que lhe corresponde e espera pelo resultado do nó que o antecede. Quando os resultados do outro nó chegam a informação é processada de forma idêntica ao que aconteceria no componente Aggregator. Posteriormente, o novo conjunto de resultados é enviado para o próximo Data Retriever. Esta metodologia permite que o sistema possa obter uma solução intermédia a qualquer momento do processamento. Caso um nó demore muito tempo a processar a sua parte, o conjunto intermédio pode ser mostrado ao utilizador. Como o tempo de resposta do sistema é importante do ponto de vista do utilizador, esta abordagem é uma solução possível para sistemas que demorem demasiado tempo a processar a totalidade de inquéritos mais complexos.
B. Os factos são extraídos dos repositórios e enviados directamente para o componente Aggregator. Nesse componente são realizadas todas as operações de agregação e integração de informação necessárias.

3.2.9 Result Set Preparation

O componente Result Set Preparation encarrega-se de preparar os dados para os apresentar ao utilizador. De acordo com as configurações do sistema, os dados podem ser apresentados como um conjunto de *links* para os documentos RDF relevantes, como um grafo de informações navegável ou com informação estatística.

3.2.10 Geração do Plano de Execução do Inquérito

A geração do plano de execução de inquérito processa-se da seguinte forma:

Após o inquérito ser submetido ao sistema este é avaliado. Se se tratar de um inquérito em linguagem natural, o índice de termos é consultado à procura de entradas dos termos

contidos no inquérito. Se se tratar de uma *query* formal em linguagem SPARQL são analisados os URIs contidos na *query* e procurados no catálogo de mapeamentos por mapeamentos com esses URIs nos vários repositórios. Se não forem encontrados mapeamentos ou entradas no índice, uma *query* SPARQL genérica é enviada a todos os Data Retrievers, para a eventualidade de existirem dados nos repositórios que não estejam indexados.

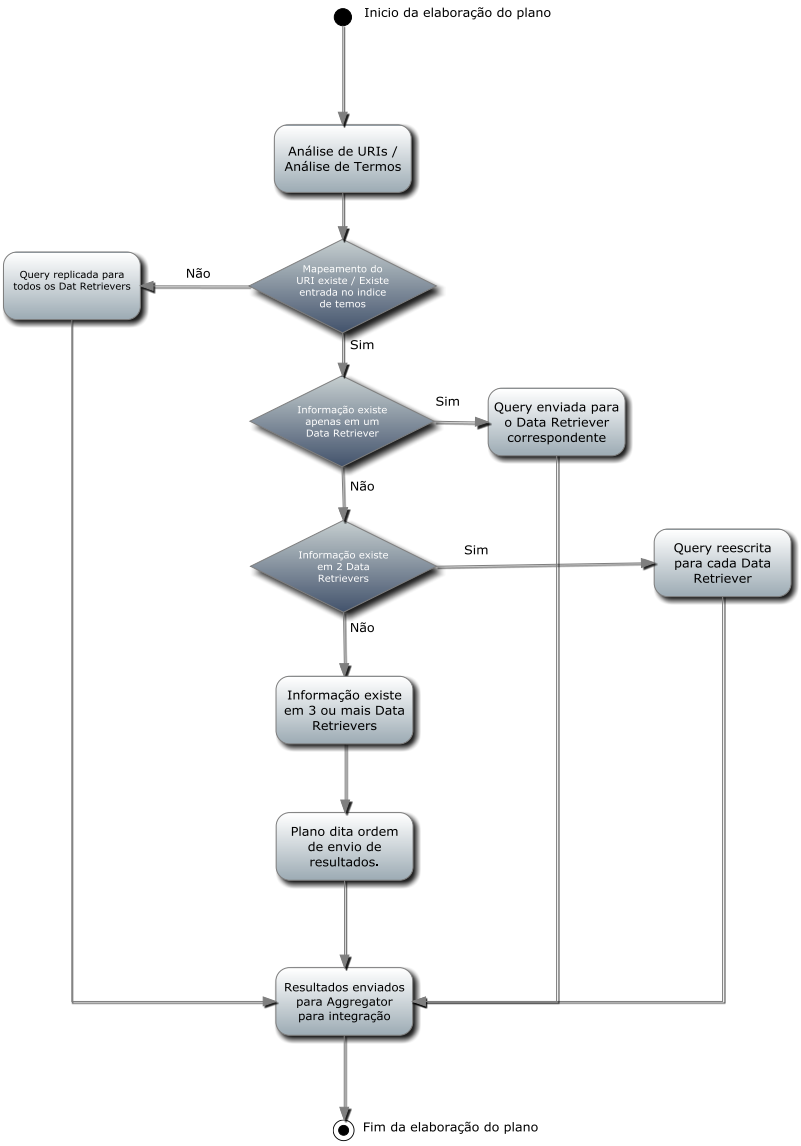

Imagem 21 - Diagrama de Elaboração do Plano de Execução de Query

No caso de existirem mapeamentos ou entradas em índice, se a informação estiver contida num repositório apenas, a query é reescrita para esse repositório e o plano de execução criado contempla apenas esse repositório.

No caso de existirem dois Data Retrievers com a informação pretendida, o plano gerado dá indicação a cada repositório para enviar os resultados para o componente Aggregator que fará a integração final de informação. Uma query SPARQL é gerada para cada repositório.

Na eventualidade de existirem mais de dois Data Retrievers com a informação, é gerada uma query para cada repositório e o plano de execução gerado informa os Data Retrievers da ordem de envio de dados entre eles. Cada Data Retriever processa a informação do nó anterior antes de efectuar a sua pesquisa.

Após processamento das pesquisas a informação é sempre enviada para o Aggregator de forma a ser processada para apresentação.

4 Descrição da Solução técnica utilizada

4.1 Linguagem de Programação utilizada

De forma a testar a arquitectura proposta foi implementado um protótipo do sistema. Este protótipo foi desenvolvido na linguagem Java, utilizando a tecnologia Swing para implementação do ambiente gráfico.

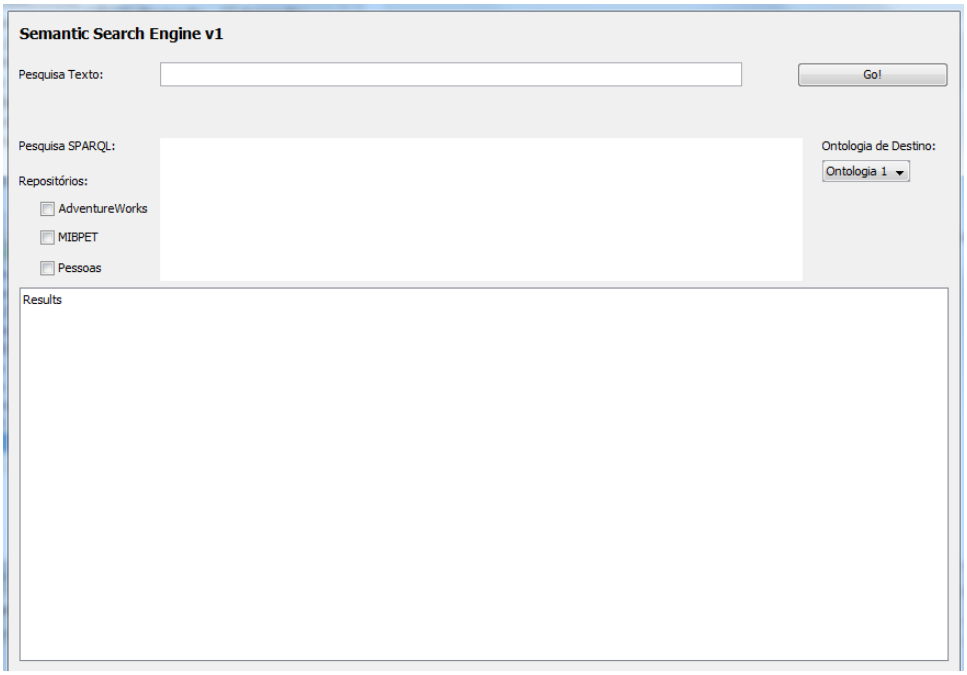

Imagem 22 - Exemplo do interface gráfico da aplicação de testes

4.2 Repositórios de Dados Utilizados

Embora fosse possível utilizar repositórios de dados RDF *online* tais como a DBPédia[10], devido a complexidade do esquema de dados utilizado nestes repositórios optou-se por simular um ambiente em que a informação está contida em vários formatos, mas é traduzida para triplos RDF conforme necessário.

Assim a solução implementada utiliza repositórios de informação em três formatos:

1. Uma base de dados relacional implementada com tecnologia MS SQL Server à qual foi aplicado um *wrapper* que disponibiliza um SPARQL Endpoint que permite efectuar pesquisas SPARQL a essa base de dados, devolvendo triplos RDF.

[10] http://pt.wikipedia.org/wiki/DBpedia

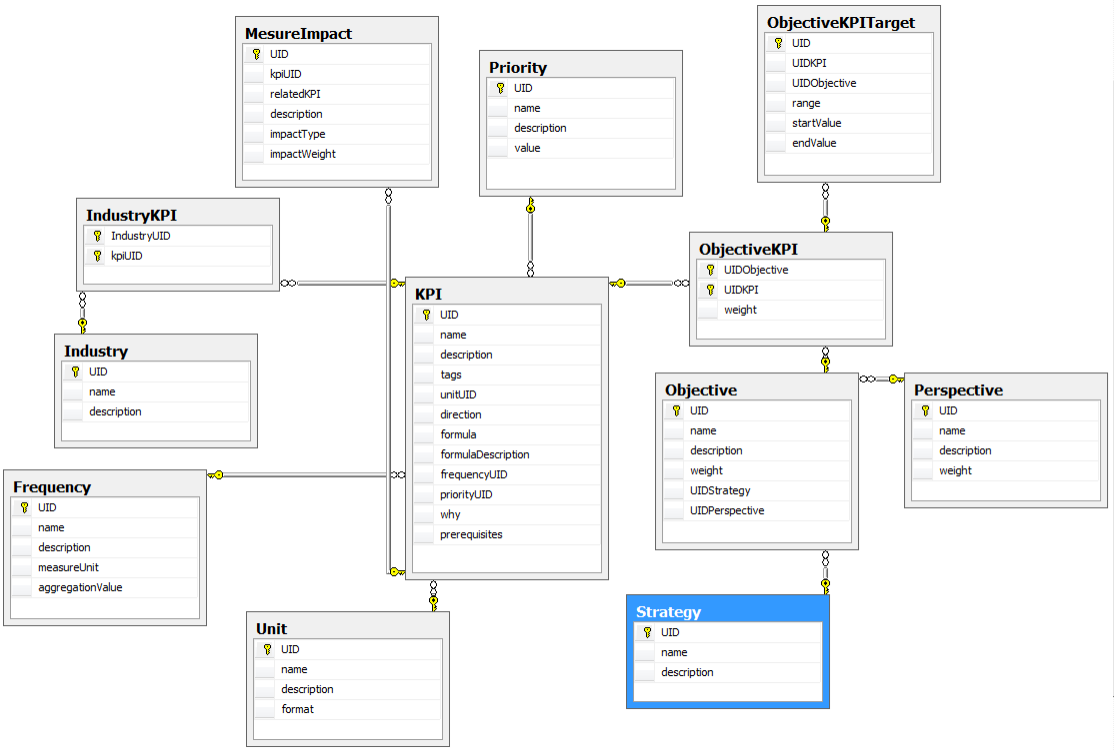

Imagem 23 - Esquema de Base de Dados de um dos Repositórios Utilizados

O *wrapper* utilizado foi a tecnologia D2RQ [11] que efectua um mapeamento entra as tabelas da base de dados relacional para classes e propriedades representadas em RDF. Este ficheiro de mapeamentos é gerado automaticamente após algumas configurações.

Foi considerada esta solução por se adequar a alguns contextos onde não é possível migrar os dados de bases de dados tradicionais (relacionais) para bases de dados de triplos RDF (Triple Stores). A utilização desta ferramenta permitiu uma familiarização com esta tecnologia que permite manter a base de dados relacional.

2. Um SPARQL Endpoint implementado com a ferramenta Joseki. Joseki trata-se de um servidor de SPARQL Endpoints disponibilizado dentro do âmbito do projecto Jena[12].

[11] http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
[12] http://jena.sourceforge.net/

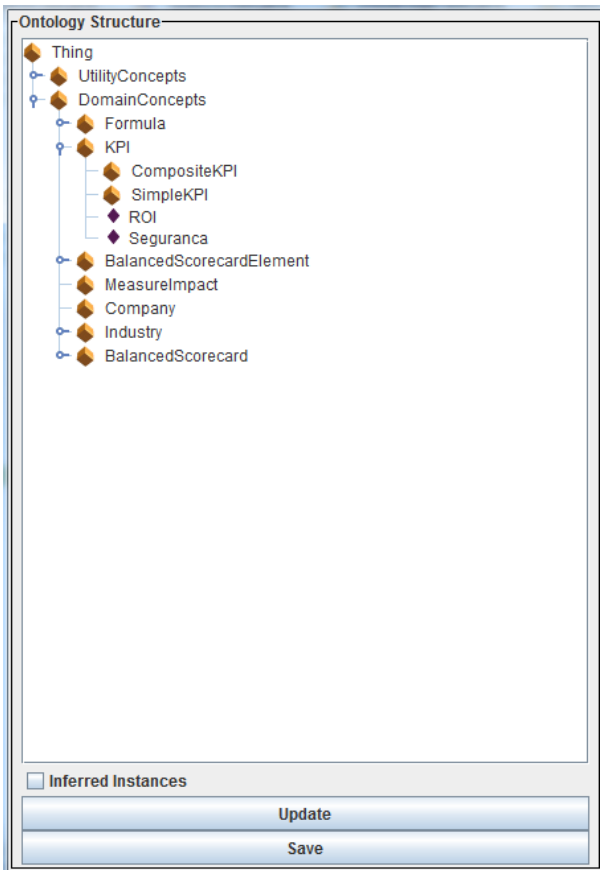

Imagem 24 - Explorador de Conceitos das Ontologias de Origem

3. Um ficheiro RDF que contém informações sobre pessoas, definido pelo autor desta dissertação.

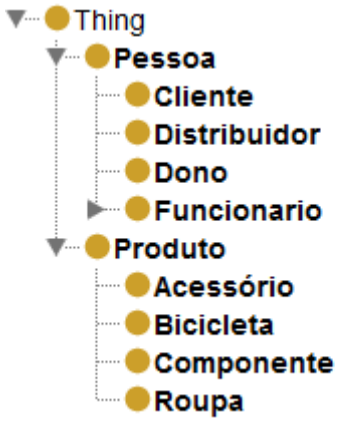

Imagem 25 - Representação da Ontologia Definida para Testes

Após escolha dos repositórios foram procuradas tecnologias de indexação de dados RDF, assim como técnicas de mapeamento manual de ontologias.

4.3 Tecnologia de Indexação escolhida

A escolha da tecnologia de indexação recaiu sobre o motor de pesquisa Apache Lucene, pois este vem incluído com a ferramenta de manipulação de ontologias escolhida para este projecto o Jena. O Jena fornece uma API de pesquisa de ontologias que implementa o protocolo SPARQL, denominada ARQ. Dentro do ARQ, existe o módulo de pesquisa de texto em linguagem natural, denominado LARQ (Lucene + ARQ).

Como ferramenta de mapeamento manual foi utilizado o MAFRA toolkit. Esta ferramenta foi utilizada para efectuar mapeamentos entre as ontologias fonte e as ontologias de destino e para efectuar os mapeamentos necessários aos testes efectuados com o algoritmo de integração escolhido [Almeida 2009].

O algoritmo de integração utilizado, efectua integração de informação por migração de dados. Isto significa que os dados são copiados das ontologias fonte para a ontologia destino. Esta migração baseia-se em atributos integradores.

Tabela 12 - Exemplo de ficheiro de Mapeamento gerada com MAFRA toolkit

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<harmonisation>
  <semanticmap.path>mapa2_semmap.rdf</semanticmap.path>
  <projectServices.path>mapa2_services.rdf</projectServices.path>
  <type>6</type>
  <name>mapa2</name>
  <source.path>OF2.rdfs</source.path>
  <target.path>OD.rdfs</target.path>
  <source.ontology>OF2.rdfs</source.ontology>
  <target.ontology>OD.rdfs</target.ontology>
</harmonisation>
```

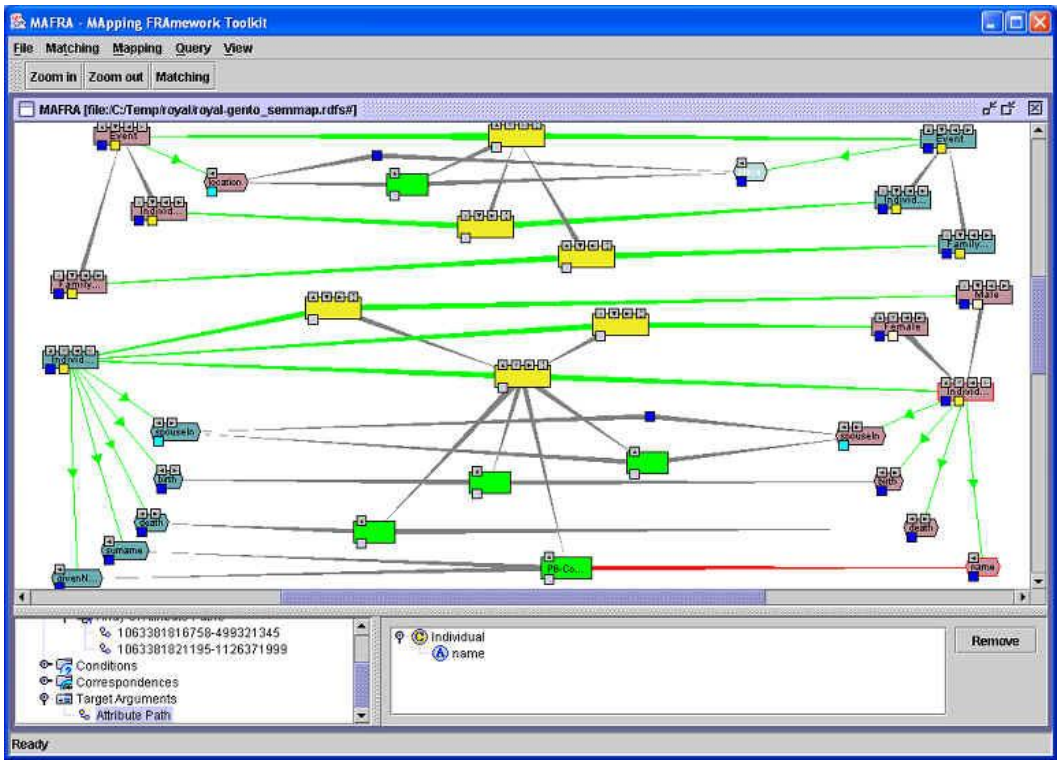

Imagem 26 - MAFRA toolkit

4.4 Exemplos de Inquéritos ao Sistema

Na imagem 27 é possível ver-se um exemplo de um inquérito em linguagem SPARQL efectuado ao sistema. Os resultados devolvidos, nesta fase do protótipo são sob a forma de triplos {sujeito,predicado,objecto}.

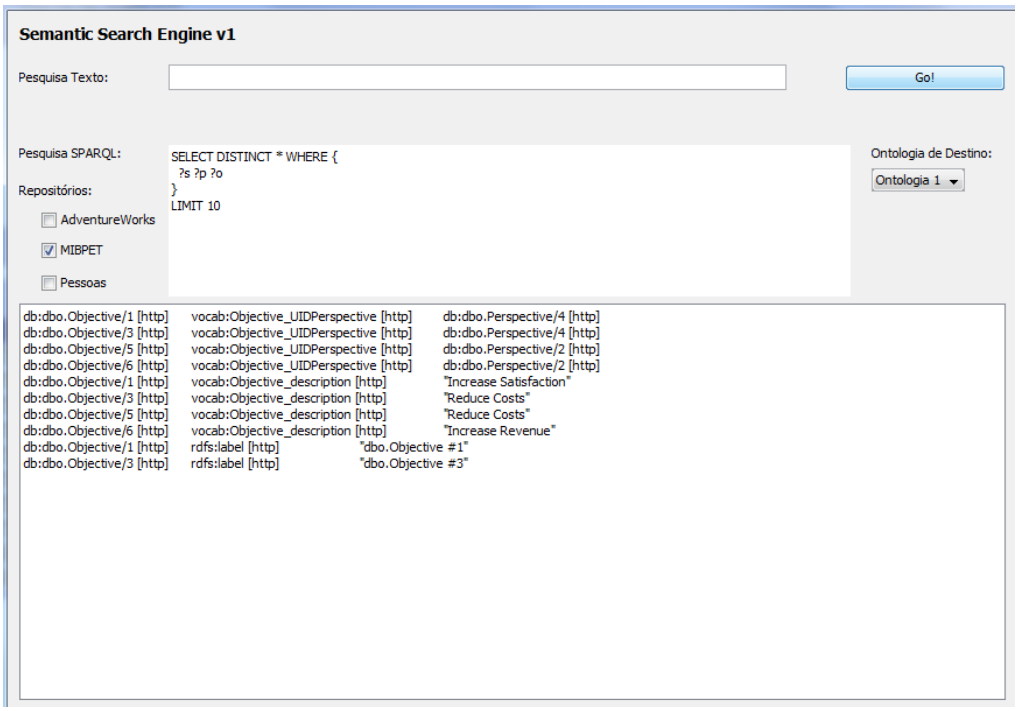

Imagem 27 - Exemplo de inquérito formal em linguagem SPARQL

Na imagem 28 está um exemplo de um inquérito em linguagem natural. Neste género de inquéritos o sistema utiliza o índice criado com a tecnologia LARQ para verificar se existe algum repositório com os termos introduzidos. Após esta análise uma pesquisa em linguagem SPARQL é efectuada aos repositórios relevantes, sendo retornados mais uma vez triplos RDF que contêm a informação pretendida.

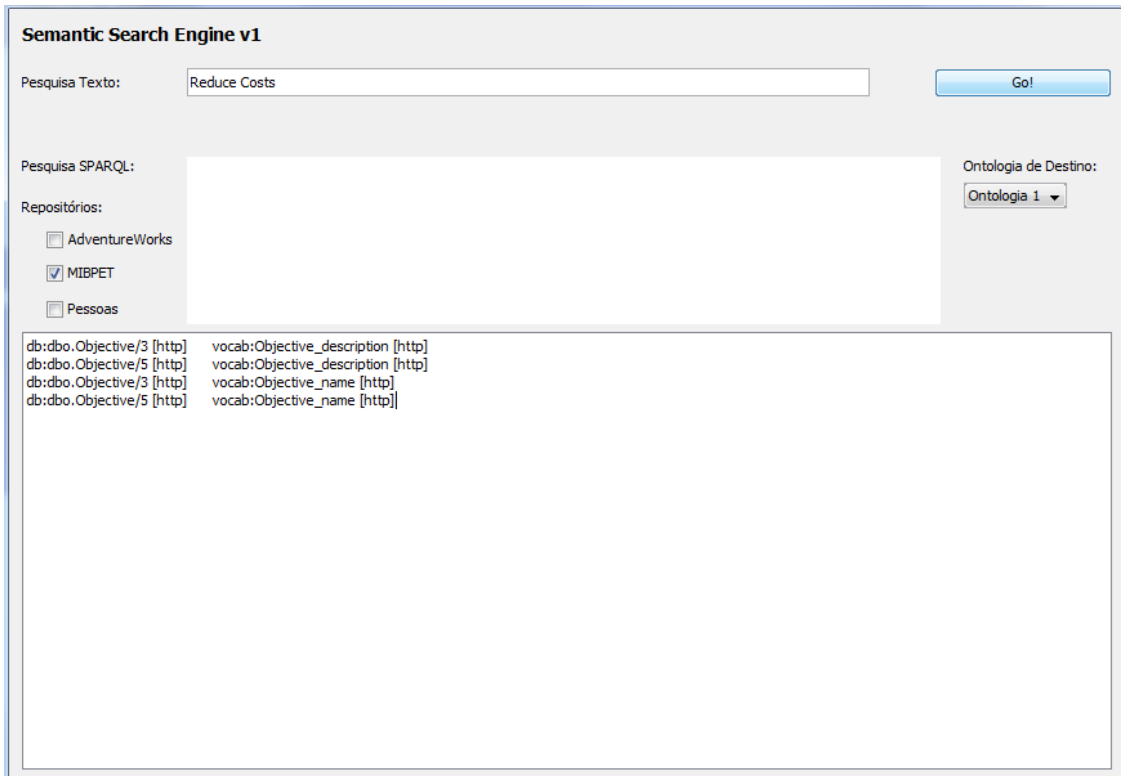

Imagem 28 - Exemplo de inquérito em linguagem natural

É possível escolher entre duas ontologias de destino possíveis através do menu no painel direito. No painel esquerdo é possível escolher quais os repositórios a considerar.

O Painel central de pesquisas SPARQL não é sensível á sintaxe da linguagem, pelo que o utilizador apenas saberá se a query está bem escrita após executar a mesma.

5 Conclusões

5.1 Objectivos Atingidos

Este trabalho teve como objectivo o estudo e sistematização das tecnologias de conhecimento disponíveis para implementação de Motores de Pesquisa Semânticos. Foi pretendido estudar também que soluções existem actualmente implementadas e quais as funcionalidades disponibilizadas por essas soluções.

Foi feito um estudo a modelos de arquitectura existentes e identificadas as limitações desses modelos. Como este trabalho se debruçou sobre a pesquisa e indexação de dados em formato RDF, foi analisada qual a melhor arquitectura para explorar fontes de dados desse tipo.

No final desta dissertação foi proposta uma arquitectura para um Sistema de Pesquisa e Indexação de Informação Federado, que apresenta uma solução de cooperação entre fontes de dados baseada em ontologias partilhadas e interligadas. Conforme analisado no capítulo 2, a cooperação entre fontes de dados é característica crucial num sistema federado que pretenda usufruir das potencialidades de Linked Open Data.

Na fase final desta dissertação foi escolhida uma abordagem prática onde se procurou aplicar a arquitectura proposta de forma a estudar a sua viabilidade. Foram analisadas tecnologias e implementado um sistema protótipo que permite indexar e pesquisar fontes de dados RDF interligadas (Linked Data). O sistema protótipo permite ainda escolher sob que ontologia os dados devem ser devolvidos e escolher que repositórios de dados devem ser contemplados durante o processamento do inquérito.

5.2 Limitações

O sistema actual não permite a adição automática de novos repositórios de dados. Para tal é necessário implementar um componente denominado Data Retriever por cada novo repositório de dados que se deseje adicionar ao sistema.

Os mapeamentos entre as ontologias de cada repositório têm que ser definidos manualmente, não sendo possível a sua geração automática.

A geração do plano de execução está simplificada, pois este estudo não incidiu sobre essa questão. Assim é possível optimizar este processo recorrendo a técnicas de optimização, como por exemplo as sugeridas por [Schmidt, Meier & Lausen 2010b] e por (22).

5.3 Trabalho Futuro

De forma a tornar o sistema mais robusto é necessário efectuar testes de stress e de performance, comparando-os com outros sistemas actuais. É necessário melhorar o

mecanismo de comunicação entre fontes de dados de forma a permitir melhor escalonamento das pesquisas.

Há ainda espaço para melhoramento na integração de informação no componente Aggregator, pois o algoritmo utilizado nos testes de [Almeida 2009] pressupõe a migração dos dados das ontologias fonte para a ontologia destino. Este processo exige computação adicional, que poderia ser distribuída entre os Data Retrievers.

É possível efectuar melhoramentos na estrutura do índice, se for implementado um mecanismo de extracção automática da estrutura de dados de cada repositório. Ao criar um índice específico para cada repositório ao invés da solução actual, que corresponde a um índice genérico, pode ser possível melhorar a performance do sistema. Desta forma troca-se espaço de armazenamento do índice por rapidez de resposta.

A implementação do protótipo está numa fase inicial, pelo que existe muito potencial para outras melhorias, tais como melhorias ao interface gráfico utilizado, utilização de triple stores para armazenamento dos triplos, criação de um interface *web* e disponibilização do serviço através de *web* services.

Referências Bibliográficas

Almeida, R., 2009. Integração de Informação por Migração em Sistemas Distribuídos e Heterogéneos.

Berners-Lee, T., Hendler, J. & Lassila, O., 2001. The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5), pp.34–43.

Cognition Technologies , Inc, 2008. Why Powerset Misses the Point: Relevancy Matters!

Dahlgren, K., 2007. *Technical Overview of Cognition's Semantic NLP™(as Applied to Search)*, Technical report, Cognition Technologies, Inc.

G\örlitz, O. & Staab, S., 2011. Federated Data Management and Query Optimization for Linked Open Data.

Grau, B.C. et al., 2008. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4), pp.309-322.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. *KNOWLEDGE ACQUISITION*, 5, p.199--220.

Hakia Inc, 2010. Semantic Search Technology - Making sense of the World's Information.

Horrocks, I., Patel-Schneider, P.F. & van Harmelen, F., 2003. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1), pp.7-26.

Horrocks, I.R., 1998. Using an Expressive Description Logic: FaCT or Fiction? *IN PROC. OF KR-98*, p.636--647.

Hu, W. & Qu, Y., 2008. Falcon-AO: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3), pp.237-239.

Ilyas, M. et al., 2005. A conceptual architecture for semantic search engine. Em *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*. pp 605–610.

Kim, H., 2002. Predicting how ontologies for the semantic web will evolve. *Communications of the ACM*, 45(2), pp.48–54.

Lacy, L.W., 2005. *OWL: representing information using the web ontology language*, Trafford Publishing.

Maedche, A. et al., 2002. MAFRA—A mapping framework for distributed ontologies. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pp.69–75.

Manjula, D. & Geetha, T.V., 2004. Semantic Search Engine. *Journal of Information & Knowledge Management (JIKM)*. Available at: http://www.worldscinet.com/jikm/03/0301/S0219649204000729.html.

Schmidt, M., Meier, M. & Lausen, G., 2010a. Foundations of SPARQL query optimization. Em *Proceedings of the 13th International Conference on Database Theory*. pp 4–33.

Schmidt, M., Meier, M. & Lausen, G., 2010b. Foundations of SPARQL query optimization. Em *Proceedings of the 13th International Conference on Database Theory*. pp 4–33.

Sirin, E. et al., 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2), pp.51-53.

Story, H. et al., 2009. FOAF+ SSL: RESTful authentication for the social web. Em *Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*.

Wu, F., Hoffmann, R. & Weld, D.S., 2008. Information extraction from Wikipedia. Em ACM Press, p 731. Available at: http://dl.acm.org/citation.cfm?id=1401978 [Acedido Outubro 9, 2011].

Referências de Sites Web

1. **The World Wide Web Consortium.** SPARQL Query. [Online] 2008. [Citação: 04 de 09 de 2011.] http://www.w3.org/TR/rdf-sparql-query/.

2. —. SPARQL Update. [Online] 2008. [Citação: 04 de 09 de 2011.] http://www.w3.org/Submission/SPARQL-Update/.

3. **DBpedia.** DBpedia Mappings. [Online] [Citação: 05 de 09 de 2011.] http://mappings.dbpedia.org/index.php/Main_Page.

4. **Wikipedia.** Semântica. *Wikipedia.* [Online] Wikimedia Foundation, 2011. http://pt.wikipedia.org/wiki/Sem%C3%A2ntica.

5. **Oracle and Sun.** Basic Java Syntax. *Oracle and Sun.* [Online] 2000. http://java.sun.com/docs/books/jls/second_edition/html/syntax.doc.html.

6. **Wikipedia.** Long Tail. *Wikipedia.* [Online] Wikimedia Foundation. http://en.wikipedia.org/wiki/Long_Tail.

7. **World Wide Web Consortium.** About W3C. *World Wide Web Consortium.* [Online] [Citação: 10 de 10 de 2011.] http://www.w3.org/Consortium/.

8. —. Extensible Markup Language (XML) 1.0. *World Wide Web Consortium.* [Online] 10 de Fevereiro de 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

9. —. Happy Fifth Birthday, XML. *World Wide Web Consortium.* [Online] 10 de Fevereiro de 2003. http://www.w3.org/2003/02/xml-at-5.html.

10. **DARPA.** Language Feature Comparison. *The DARPA Agent Markup Language Homepage.* [Online] 2002. [Citação: 10 de Outubro de 2011.] http://www.daml.org/language/features.html.

11. **World Wide Web Consortium.** OWL Web Ontology Language Guide. *World Wide Web Consortium.* [Online] 10 de Fevereiro de 2004. http://www.w3.org/TR/owl-guide/.

12. —. OWL 2 Web Ontology Language Document Overview. *World Wide Web Consortium.* [Online] 27 de Outubro de 2009. http://www.w3.org/TR/owl2-overview/.

13. —. SweoIG/TaskForces/CommunityProjects/LinkingOpenData - W3C Wiki. *World Wide Web Consortium.* [Online] Outubro de 2011. http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.

14. **Cyganiak, Richard.** The Linking Open Data cloud diagram. *Richard Cyganiak's Homepage.* [Online] 2007. http://richard.cyganiak.de/2007/10/lod/.

15. —. Pubby - A Linked Data Frontend for SPARQL Endpoints. *WWW4 - research application server of the Lehrstuhl für Wirtschaftsinformatik.* [Online] Janeiro de 2011. http://www4.wiwiss.fu-berlin.de/pubby/.

16. **Amazon Web Services.** AWS Case Study: Powerset. *Amazon Web Services.* [Online] 2008. http://aws.amazon.com/solutions/case-studies/powerset/.

17. **Morrison, Chris.** Powerset: Don't call us a search engine. *VentureBeat.* [Online] Abril de 2008. http://venturebeat.com/2008/04/10/powerset-dont-call-us-a-search-engine/.

18. **Butler, Phil.** Powerset and hakia - Quest For The Semantic Web. *ReadWriteWeb.* [Online] Julho de 2007. http://www.readwriteweb.com/archives/powerset_and_hakia_quest_for_semantic_web.php.

19. **Hakia, Inc.** Query Detection and Extraction - QDEX - System. *Hakia.* [Online] 2010. http://company.hakia.com/new/qdex.html.

20. **Pandia.** Top 5 Semantic Search Engines. *Pandia Search Engine News.* [Online] Fevereiro de 2009. http://www.pandia.com/sew/1262-top-5-semantic-search-engines.html.

21. **Radhakrishnan, Arun.** Summarization, the Answer to Web Search : Interview with Dmitri Soubbotin of SenseBot. *Search Engine Journal.* [Online] Dezembro de 2007. http://www.searchenginejournal.com/summarization-the-answer-to-web-search-interview-with-dmitri-soubbotin-of-sensebot/6094/.

22. **World Wide Web Consortium.** SPARQL 1.1 Federation Extensions. *World Wide Web Consortium.* [Online] Junho de 2010. http://www.w3.org/TR/sparql11-federated-query/.

Anexo A – Tabelas comparativas das várias tecnologias semânticas analisadas

Tabela 13 - Comparação entre Motores de Pesquisa Semânticos 1

	Hakia	SenseBot	Powerset (Bing)	DeepDyve	Cognition
Search Method / Método de Pesquisa	Keyword-based Query	Keyword-based Query	Keyword-based Query	Query or paragraph	User must select a database and then insert query
Scope / Âmbito	General Purpose	General Purpose	Small sized structured corpora (ie Wikipedia DB, Enterprise DB)	Databases, Scientific Journals/Papers, Research Statistics	Databased, 4DBs available:(public.resource.org;MEDLINE;Wikipedia in english;New English Translation Dicionary, Bible in English)
Presentation Method / Método de Apresentação de Resultados	Results are separated by tabs, each tab represents a source (Web, Trusted Sites, Blogs, Images, Twitter,etc)	Organises a Resume of the given topic (snipets of info about the query). Tries to give the answer to the query in the snippet.	No Information	Displays Title of scientific journal/paper, Author and Cost of reading full article	Title of Resource and very brief description
Technologies / Tecnologias Utilizadas	Proprietary: Qdex,SemanticRank ,Comercial Ontology	No Information	Amazon Web Services infrastructure (as of May,2008)	No Information	No Information
Multilanguage / Multilingue	Almost none	NO	? Info N\a	NO	NO
Search Result Quality / Qualidade dos Resultados	Good. Results are relevant to the query and semantically acurate in most cases	Average. Some queries produce few results of almost zero relevance	? Info N\a	Dependant of search query. Good for scientific research.	Results only available in separate areas, and only in the areas described in Scope. Lacks relevance, seems to search for keywords in text instead of doing semantic analysis
Short Tail Queries	Yes	Yes	Yes	Yes	Yes

Tabela 14 - Comparação entre Motores de Pesquisa Semânticos 2

	Hakia	SenseBot	Powerset (Bing)	DeepDyve	Cognition
Long Tail Queries		No practical support for questions or phrases. Limited support for long queries	Documentation says it supports small phrases - not tested	Yes - up to 25000 characters	
Ontologias	Yes	No	No	NO	Yes
RDF	No	No	No	No	No
OWL	No	No	No	No	No
Query Refinement / Parametrizavél	No	No	No	Yes - Limited (Filters:Words in Title, Author, Date, Data Source, Content Info)	Yes (infered)
Final Version	No	No	No	Yes	No indication of beta state on site
Beta	Yes	Yes	Yes - Present state unknown	No indication of beta state on site	No

Tabela 15 - Comparação entre Motores de Pesquisa Semânticos 3

	Hakia	SenseBot	Powerset (Bing)	DeepDyve	Cognition
Open Source	No	No	No	No	No
SaaS (Software as a Service)*	Yes	Yes	Yes (bought by Microsoft)	Yes	Fast
Pay per View	No	No	No	YES	Simple
Rapidez	Slow	Sluggish	Slow (documentation speaks of 56 second search time)	OK	Simple to use, but limited scope
Simplicity of Use / Simplicidade	Simple	Simple to make query / Hard to read results	Simplle	Some learning curve, but Simple enough	Limited to the Databases present. Semantic Search not evident. No possibility of refinement
Usability / Usabilidade	Computing time is a usability problem.	Not very usable due to computing time necessary	Not usable at all (not available)	Simple to use and learn to refine a query	
Shortcomings / Limitações	Long computing time , NPL not accurate; word stem not solid	Long computing times (some results don't compute at all). Resumes don't offer clarity. Hard to find relevant sites (sources). Result presentation is confusing. No possibility of altering the query. No multimedia content. No query refinement possible	Powerset is not available for testing to the general public. Is now part of Bing Search Engine, but if the technology is in use in Bing.com is yet unknown	User has to PAY to have access to the full result (paper/journal or other source) ; Only good for scientific data search. No description of result content.	

Tabela 16 - Testes Práticos aos Motores de Pesquisa Semânticos Estudados

Test Queries:	Was the answer in the search results?	Time of search	Readability of results	Relevance
why do people laugh?				
Hakia	No	10s	confusing	no relevance
Sensebot	No (no result)	>60s (no response)	confusing	n\a
Powerset*	n\a	n\a	n\a	n\a
Deepdyve	no	5s	High	no relevance
Cognition	no	8s	Medium	no relevance
semantic web				
Hakia	Yes, first result (Wikipedia)	5s	High	High
Sensebot	Yes	23s	Readable	Medium-High
Powerset*	n\a	n\a	n\a	n\a
Deepdyve	No	7s	High	High
Cognition	Yes	8s	High	High
in what year was Bill Clinton president?				
Hakia	Yes, second result (answers.com)	2s	High	High
Sensebot	Yes	10s	Medium	Medium
Powerset*	n\a	n\a	n\a	n\a
Deepdyve	No	5s	High	No relevance
Cognition	No	11s	Medium	No relevance
Who is George Bush?				
Hakia	Yes, first and second results (Wikipedia)	25s (blog tab 20s)	High	High
Sensebot	Yes	12s	Confusing	High
Powerset*	n\a	n\a	n\a	n\a
Deepdyve	No	8s	Medium	No relevance
Cognition	Yes	10	Medium	High
e.coli				
Hakia	Yes, first result (Wikipedia)	11s	High	High
Sensebot	Yes	15s	High	High
Powerset*	n\a	n\a	n\a	n\a
Deepdyve	Yes	8s	High	High
Cognition	Yes	10s	High	High

* O Powerset não estava disponível na data em que os testes foram efectuados, por esse motivo não foram obtidos resultados nestes testes.

Anexo B – Descrição das Propriedades da Tabela 5 (10)

- Listas Ligadas
  - Listas de elementos não ordenados em que o número de elementos da lista pode ser especificado. Por exemplo, as tags `rdf:Seq` e `rdf:Bag` não fornecem indicação de que a lista se encontra completa
- Restrições de Cardinalidade
  - Limitam o número de *statements* dentro de um mesmo sujeito e predicado. Por exemplo, os operadores Kleene `?` (0 or 1), `*` (0+), e `+` (1+) no XML fornecem mecanismos básicos de restrição de cardinalidade.
- Expressões de Classes
  - Definir uma classe como uma `unionOf`, `disjointUnionOf`, `intersectionOf`, ou `complementOf` (respectivamente, união de classes, união disjunta de classes, intersecção de classes ou complemento de classe).
- Tipos de Dados
  - Definição do tipo de dados dos literais (inteiro, sequencia de caracteres, data, etc).
- Classes Definidas
  - Definição de condições necessárias e suficientes de pertença a uma classe. Por exemplo, uma Criança é uma Pessoa com idade inferior a 18 anos. Esta condição define que todas as Pessoas com idade inferior a 18 anos são também membros da classe Criança.
- Enumerações
  - Especificação de um conjunto restrito de valores possíveis para um determinado atributo.
- Equivalência
  - Definição de uma classe, propriedade ou instância como equivalente a outra classe, propriedade ou instância.
- Extensibilidade
  - Permitir o uso de novas propriedades dentro de classes existentes, assim como permitir estender o conjunto de classes existentes, na mesma ou em outra linguagem.
- Semântica Formal
  - Permitir definir axiomas e relações semânticas de forma explicita (formal).
- Herança
  - Permitir relações `subClassOf` e `subPropertyOf` (subclasses e subpropriedades).
- Inferência
  - Permitir relações `TransitiveProperty`, `inverseOf`, e `disjointWith` que fornecem informação adicional para um motor de inferência (respectivamente propriedade transitiva, inverso de, disjunção).
- Restrições Locais
  - Possibilidade de definir o `domain` e o `range` de uma propriedade (respectivamente o domínio e alcance).
- Restrições Qualificadas
  - Permitir restrições do tipo “Todos os Ratos são mais pequenos que qualquer Elefante”.
- Reificação
  - Permitir que uma *statement* seja o sujeito de outro *statement.* O mecanismo de reificação fornece um meio de registo de fontes de dados, timestamps e outros, sem ser intrusivo no modelo de dados.