INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Utilização de Conhecimento Semântico num sistema de Recuperação de Informação

Luís Maria Valadas Caselli

(Bacharel)

Dissertação de natureza científica realizada para obtenção do grau de Mestre em Engenharia Informática e de Computadores

Orientador: Professor-adjunto Paulo Trigo, ISEL

Júri:

Presidente: Professor-adjunto Pedro Pereira, ISEL

Vogais:

Professor-adjunto Paulo Trigo, ISEL

Professor-auxiliar Paulo Urbano, DI-FCUL

Setembro de 2010

Resumo

As soluções de recuperação de informação actuais, que classificam documentos de acordo com a sua relevância, baseiam-se na sua grande maioria em algoritmos estatísticos não recorrendo a conhecimento semântico sobre o domínio pesquisado. Esta abordagem tem sido adoptada em grande escala e com bons resultados, em grande medida devido à baixa complexidade que apresenta e à facilidade e universalidade com que pode ser aplicada.

Esta tese explora uma alternativa à pesquisa de informação baseada apenas na contagem de termos, tentando ao mesmo tempo manter a complexidade em níveis semelhantes. O trabalho desenvolvido baseia-se na utilização de Lógica de Descrição para representar conhecimento sobre um domínio e inferir pesquisas alternativas, que possam conduzir a um ganho na precisão dos resultados. Avaliam-se os ganhos obtidos por esta abordagem, nomeadamente na obtenção de resultados baseados em semelhança semântica e não puramente sintáctica. Esta abordagem permite obter resultados que não seriam considerados numa pesquisa puramente sintáctica.

Abstract

Most of today's information retrieval solutions which rank a collection of documents according to relevance, do so based purely on statistical metric's. Most of those solutions incorporate little or none semantic knowledge about the domain of discourse. However, those approaches, have been used in almost every field with a fair amount of success, mainly due to its low complexity, ease of implementation and range of applicability. Nonetheless, nowadays there is a growing interest (both academical and corporate) in the development of new and more effective ways of searching for information.

This thesis explores an alternative method of searching for information (as opposed to statistical ones), while trying to maintain the complexity at a similar level. Description Logics are used to represent the knowledge of a domain and to reason and infer alternative searches that lead to a gain in the precision of the results. The gains obtained by these approach are evaluated, namely in the obtention of results based on semantic similarity and not just syntactic similarity. This solution is able to retrieve results that would not be considered in a pure syntactic search approach.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Capítulo 1

Introdução

Nas últimas décadas tem se assistido a um crescimento elevado da informação disponível, com grande parte dos conteúdos produzidos (ou armazenados) em formatos não estruturados (**?**) orientados para o manuseamento humano. Neste cenário, a manipulação de informação torna-se de primordial importância, existindo diversas áreas de investigação (e.g. inteligência organizacional (*business-inteligence*), mineração de dados (*data-mining*), motores de pesquisa), que tentam aumentar a eficácia no tratamento de grandes volumes de informação. Procurar informação utilizando um motor de pesquisa, implica muitas vezes ler vários documentos para conseguir obter a informação procurada, não só porque alguns documentos não contêm o assunto desejado mas porque os documentos contêm normalmente apenas parte da informação. Os resultados apresentados por um motor de pesquisa podem divergir consideravelmente para dois critérios de pesquisa semanticamente semelhantes mas sintacticamente diferentes.

1.1 Motivação

A pesquisa de informação num corpo de conhecimento (conjunto de documentos, páginas *web* ou bases de dados) constitui um campo de investigação activo e actual, com grande ênfase nas técnicas que suportam a construção de motores de pesquisa. Actualmente, os motores de pesquisa baseiam-se sobretudo em métodos estatísticos aplicados a termos e mecanismos de ordenação para caracterizar a relevância dos resultados. Este tipo de abordagem, essencialmente sintáctica, baseia-se na análise das semelhanças entre termos e tem limitações, uma vez que numa pesquisa a relação entre o critério de pesquisa e o resultado tem muitas vezes uma componente semântica, que transcende a mera semelhança sintáctica.

É objectivo deste trabalho desenvolver técnicas que permitam incorporar informação semântica numa pesquisa com o intuito de aumentar a precisão dos resultados de pesquisa. Será avaliado o potencial desta abordagem para, dado um critério de pesquisa, obter resultados relacionados mesmo não existindo semelhança sintáctica.

O conceito de ontologia será usado para descrever o modelo de conhecimento sobre um domínio. A ontologia será especifica de um domínio contendo conceitos e relações que o caracterizem. O foco num domínio permite contextualizar e assim reduzir a complexidade associada à inferência de informação semântica a partir de um critério de pesquisa. Por outro lado, a utilização de uma ontologia com domínio demasiado genérico elevaria o problema ao campo do processamento de língua natural (NLP), que não é o objectivo deste trabalho. A ontologia será introduzida de forma modular para que seja possível a sua substituição para endereçar diferentes domínios.

1.1.1 Hipóteses

Este trabalho desenvolveu-se em torno da formulação de 3 hipóteses cuja validade se pretende explorar:

Hipótese 1 - A adopção de ontologias suporta a composição modular de conceitos e relações permitindo endereçar diferentes domínios;

Hipótese 2 - Recuperar informação incorporando relações semânticas permite obter resultados que transcendam a mera semelhança sintáctica;

Hipótese 3 - Uma pesquisa baseada em frases que incorpore inferência sobre ontologias permite construir um modelo com uma baixa complexidade (e.g., em relação ao NLP).

A primeira hipótese explora a utilização de ontologias para desenvolver um modelo flexível, de baixa complexidade e escalável.

A segunda hipótese explora a intuição de que um conhecimento semântico, aliado a pesquisa sintáctica, deve ter capacidade para aumentar a precisão dos resultados.

A terceira hipótese surge do pressuposto de que uma frase não é apenas uma colecção de termos, pelo que encontrar uma frase num documento é mais relevante do que encontrar os termos da frase dispersos pelo documento. Muitas vezes um motor de pesquisa devolve um documento com alta frequência dos termos pesquisados, mas ao ler o documento constata-se que não existem frases pertinentes para o assunto que motivou a pesquisa! Os motores de pesquisa sintácticos actuais já são sensíveis a esta questão e têm geralmente algoritmos que avaliam métricas de proximidade dos termos e as usam no cálculo da relevância do resultado. A hipótese aqui adiantada visa explorar a utilização de pesquisas baseadas em frases para aumentar a precisão dos resultados sem recorrer a algoritmos complexos (como o de NLP).

1.2 Organização do documento

Este documento está organizado em seis capítulos com o seguinte conteúdo:

Capítulo 1 - Introdução: Apresentação da motivação e objectivos do trabalho. Formulação e descrição das hipóteses que se pretendem explorar.

Capítulo 2 - Estado da Arte: Análise de tecnologias e teorias relevantes para o trabalho a desenvolver. Estudo comparativo de soluções de pesquisa semântica. Análise de alguns trabalhos académicos na mesma área ou relacionados.

Capítulo 3 - Modelo: Descrição do modelo proposto e das alternativas consideradas. Explicação de como o modelo permite responder aos objectivos do trabalho.

Capítulo 4 - Concretização: Descrição da concretização do modelo explicando as opções de implementação tomadas e as soluções encontradas.

Capítulo 5 - Validação: Validação das hipóteses formuladas e avaliação do modelo. Análise e interpretação dos resultados obtidos.

Capítulo 6 - Conclusões: Conclusões sobre o trabalho desenvolvido. Trabalho futuro.

Capítulo 2

Estado da Arte

Neste capítulo são analisadas abordagens de investigação assim como soluções e tecnologias relevantes para o trabalho a desenvolver.

2.1 Rede Semântica (*Semantic Web*)

O conceito de Rede Semântica (**?**) surgiu para auxiliar e facilitar o processamento automático de grandes volumes de dados não estruturados. Este conceito propõe a passagem de um contexto sintáctico em que os recursos existentes na *internet* são inteligíveis apenas por humanos, para um contexto semântico onde através da inferência sobre conceitos a informação disponível possa ser processada por máquinas.

A figura **??**[1] ilustra os formalismos subjacentes à construção do modelo de rede semântica. Neste modelo é utilizado o *Extended Markup Language* (XML) (ao nível mais elementar) como linguagem para representar a informação. O XML fornece as regras

[1] retirado de `http://www.w3c.rl.ac.uk/pasttalks/slidemaker/Pandora/talk/slide11-0.html`

sintácticas, sendo a informação semântica fornecida por formalismos superiores. O *Resource Description Framework* (RDF) (**?**) e o *RDF Schema* (RDFS) (**?**) constituem um modelo que através de tripletos descreve conceitos, propriedades e indivíduos formando grafos de inter-relações envolvendo conceitos e indivíduos através de propriedades e que se pode escrever (seriar) num dialecto XML.

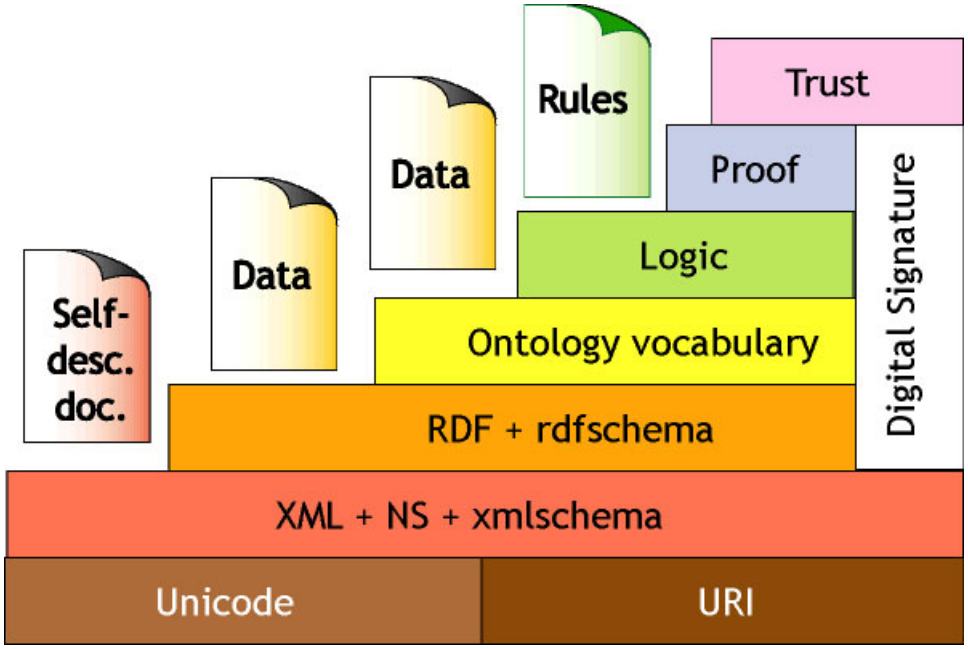

Figura 2.1: Estrutura da Rede Semântica

O RDFS estende o RDF para incluir as noções de subclasse, classe equivalente, subpropriedade, domínios e contradomínios das propriedades, fornecendo a semântica base que permite modelar e realizar inferência sobre um domínio. O *Ontology Inference Layer* (OIL) (**?**) foi proposto como uma linguagem que estende o RDFS, baseada em Lógica de Descrição (DL) e que permite especificar regras que tem de ser satisfeitas pelos conceitos, propriedades e indivíduos. A DL permite realizar inferência sobre os conceitos e verificar a consistência das regras.

2.2 Linguagens de descrição de ontologias

A primeira linguagem padronizada (*standard*) para descrição de ontologias, *Web Ontology Language* (OWL 1), foi proposta em 2004 pelo *World Wide Web Consortium* (W3C) e representa a consolidação e integração do OIL e do *DARPA Agent Markup Language* (DAML-ONT) (**?**) num standard de forma a promover a interoperabilidade e a potenciar a sua adopção e utilização. O *standard* OWL evoluiu e encontra-se actualmente na versão 3 existindo três variantes da linguagem — OWL Lite, OWL DL e OWL Full (**?**) baseadas em diferentes famílias de DL. A linguagem OWL permite definir taxionomias (TBox) através de axiomas de igualdade e de inclusão, e indivíduos (ABox) que seguem a estrutura da TBox. Existem ferramentas como o Protege [2] ou o OwlSight [3] que disponibilizam interfaces gráficas para a criação e manipulação de ontologias.

2.3 Lógica de Descrição (DL)

A Lógica de Descrição compreende uma família de formalismos focados na representação e inferência de conhecimento. A DL permite construir uma base de conhecimento (KB) na forma de conceitos e de relações entre conceitos. Em DL a escolha dos construtores da linguagem permite controlar o equilíbrio entre o poder de representação e o poder de inferência (**?**) conseguindo-se desta forma tornar o problema da inferência tratável. Estudos elaborados relativamente à complexidade dos algoritmos de inferência, possibilitaram detalhar a implicação dos construtores da linguagem na complexidade de inferência, permitindo especificar linguagens decidíveis. Estes linguagens asseguram uma decisão sobre a validade de uma asserção em tempo finito. A influência dos construtores da linguagem na complexidade associada à inferência é de tal modo importante que existem várias linguagens de DL, cada uma com um conjunto de construtores. Uma linguagem DL é

[2] `http://protege.stanford.edu`

[3] `http://pellet.owldl.com/ontology-browser`

normalmente referenciada pelos construtores que disponibiliza.

A representação de conhecimento em DL é feita através da definição de conceitos recorrendo aos construtores da linguagem que permitem definir conceitos atómicos e conceitos não atómicos (definidos com base em conceitos atómicos). Desta forma é definida uma taxionomia (TBox) onde se exprimem restrições e relações entre conceitos. Os indivíduos da ABox são associados aos conceitos da TBox através de relações de pertença entre indivíduos e conceitos. O conjunto dos conceitos e axiomas da TBox e da ABox definem uma base de conhecimento (KB) da DL. Esta KB tem no entanto um carácter abstracto sendo necessário associar-lhe uma realidade concreta através de uma interpretação que associe indivíduos concretos aos indivíduos abstractos definidos na ABox.

Para realizar inferência sobre uma KB é primeiro necessário garantir a sua consistência. Uma KB é consistente se existir pelo menos um modelo que satisfaça a KB (TBox + ABox). A consistência de uma KB garante a não existência na TBox de conceitos subsumidos pelo conjunto vazio ($\perp$).

2.3.1 OWL-DL

Cada variante da OWL (OWL-Lite, OWL-DL e OWL-Full) corresponde a uma linguagem DL. Cada variante (na ordem indicada) representa uma extensão da anterior com poder expressivo acrescido. Neste trabalho é usada OWL-DL para modelar as ontologias e para realizar inferência. A OWL-DL é uma variação sintáctica da linguagem $SHOIN^{(D)}$ (**?**) e suporta os seguintes construtores de linguagem:

- S (ALC) - Linguagem atributiva (intersecção de conceitos, restrições universais, quantificação existencial limitada,negação de conceitos atómicos) com negação de conceitos não atómicos;
- H - Hierarquia de propriedades;

- O - Nominais (restrição sobre valores);
- I - Propriedade inversa;
- N - Restrição de cardinalidade;
- $^{(D)}$ - Utilização de tipos XML;

A OWL-DL foi concebida para ser decidível oferecendo uma capacidade expressiva elevada, mantendo os processos de inferência computacionalmente tratáveis.

O Protege é uma ferramenta gráfica de construção de ontologias OWL-DL e o Pellet[4] (**?**) é um motor de inferência OWL-DL (disponível como *plugin* do protege). Os dois juntos fornecem um ambiente de desenvolvimento integrado (IDE) para modelar e realizar inferência sobre ontologias OWL-DL. Para integração em aplicações informáticas é necessário recorrer à interface de programação de aplicação (API) OWL que permite a manipulação das ontologias e inclui uma interface (`OWLReasoner`) para realizar inferência. O Pellet fornece uma API que implementa a interface `OWLReasoner` (classe `PelletReasoner`).

Existem outras ferramentas para realizar inferência (e.g. FACT++ e Hermit), e existem *frameworks* para manipulação e realização de inferência sobre ontologias OWL-DL. Uma das mais proeminentes é a JENA que reúne um conjunto extenso de características (suporte para ontologias definidas em RDF e OWL, persistência de ontologias em BD, inferência, SPARQL-DL). Esta *framework* não implementa a interface definida (pelo W3C) na API OWL recorrendo a uma interface própria que lhe permite lidar de forma transparente com ontologias definidas em RDF, OWL-Lite, OWL-DL ou OWL-Full. A *framework* JENA disponibiliza uma interface (inexistente na API OWL) para realizar interrogações SPARQL-DL. O SPARQL-DL é uma linguagem de interrogação de ontologias que realiza inferência (ao contrário da versão SPARQL que apenas interroga informação explicita).

[4] `http://clarkparsia.com/pellet/`

2.4 Lógica de Descrição Probabilística

Em DL uma TBox descreve não só conceitos do domínio mas também relações entre conceitos recorrendo a axiomas de subsunção ou de inclusão como por exemplo $A \sqsubseteq B$, que indica que quaisquer que sejam as interpretações dos conceitos A e B, qualquer indivíduo que satisfaça A satisfaz também B. No entanto não se consegue exprimir em DL a noção de que dois conceitos são semelhantes, ou seja que partilham grande parte dos indivíduos, mas não todos.

Para endereçar esta questão surgiram a Lógica de Descrição Difusa (DDL) e a Lógica de Descrição Probabilística (PDL). Antes de se prosseguir é importante clarificar as diferenças entre a DDL e a PDL. O objectivo da DDL é lidar com conhecimento vago (e.g. o indivíduo x pertence ao conceito A de acordo com determinada função de pertença). A PDL lida com incerteza (e.g. o indivíduo x pode pertencer ao conceito A sendo que essa pertença tem uma determinada probabilidade de ocorrer). Na DDL conhece-se o grau de pertença do indivíduo x ao conceito A. Na PDL desconhece-se se o indivíduo x pertence ou não ao conceito A, apenas se sabe a probabilidade de pertencer.

Neste trabalho iremos analisar e utilizar a Lógica de Descrição Probabilística. Existem vários trabalhos que abordam o tema da PDL (**?**) no entanto existem poucas implementações. As duas implementações encontradas têm abordagens diferentes. O PR-OWL [5] (**?**) estende a DL de forma a incorporar incerteza utilizando *Multy Entity Bayesian Networks Logic* (MEBN) (**?**). O Pronto [6] (**?**) é uma implementação baseada no trabalho de (**?**) que utiliza a linguagem P-SHIQ e que segue a linha de desenvolvimento do Pellet.

[5] `http://www.pr-owl.org/`

[6] `http://pellet.owldl.com/pronto`

PR-OWL

Para compreender a implementação PR-OWL é necessário referir o conceito de *upper ontology*. Uma *upper ontology* é uma ontologia que exprime conhecimento sobre um domínio abstracto, genérico ou fundamental e que constituí um elemento base reutilizável noutras ontologias. Um exemplo de uma *upper ontology* é o de uma ontologia que descreva uma língua (Portuguesa, Inglesa ou outra) com todas as regras gramaticais associadas. O PR-OWL estabelece justamente uma destas *upper ontology* para descrever o domínio MEBN. Esta ontologia pode ser importada numa ontologia clássica permitindo definir um modelo probabilístico na ontologia clássica. A inferência probabilística é conseguida utilizando o modelo previamente definido para construir uma rede de Bayes que é depois fornecida a um motor de inferência (UnBBayes-MEBN) com capacidade para processar a rede de Bayes. O PR-OWL tem, no entanto, como limitação não permitir referências cíclicas não sendo possível realizar inferência sobre uma ontologia definida com este formalismo que origine uma rede de Bayes com ciclos (advêm do facto de utilizar redes de Bayes como formalismo de cálculo de probabilidades).

Pronto

O Pronto é uma extensão do motor de inferência Pellet com suporte para representação e inferência de conhecimento probabilístico e recorre a uma abordagem diferente do PR-OWL uma vez que não define uma *upper ontology* e não utiliza redes de Bayes para calcular as probabilidades. Outra diferença é o facto de fornecer uma API para integração quando o PR-OWL apenas fornece uma interface gráfica. O Pronto define uma ontologia probabilística como extensão da ontologia clássica. A ontologia probabilística importa a ontologia clássica e define restrições condicionais (*conditional constraints*) na forma de axiomas de subsunção probabilísticos, onde a probabilidade associada à subsunção é definida na forma de uma anotação. Uma das grandes diferenças relativamente ao PR-OWL consiste no facto de no Pronto as probabilidades serem definidas na forma de um intervalo

de probabilidade. Desta forma uma subsunção tem associado um intervalo de probabilidade e não uma probabilidade absoluta. Nesta abordagem a base de conhecimento é constituída por uma PTBox e uma PABox que estendem respectivamente os conceitos de TBox e ABox com informação probabilística.

O mecanismo de inferência consiste numa implementação do trabalho de (**?**) e recorre à noção de *Lexicographic Entailment*. Neste mecanismo as restrições condicionais definidas na ontologia probabilística são usadas para produzir conclusões probabilísticas de uma forma hierárquica. O algoritmo define conjuntos de restrições condicionais com vários níveis de especificidade de forma a poder utilizar as restrições mais especificas em detrimento das menos especificas.

Apesar de não sofrer do problema de referências cíclicas do PR-OWL o Pronto tem contudo o problema de seguir uma abordagem focada na subsunção não fornecendo formalismos para a definição e inferência de equivalência.

2.5 Motor de recuperação de informação

Existem várias soluções de recuperação de informação sendo o Lucene [7] uma das mais conhecidas. O Lucene possuí um conjunto de características importantes para este trabalho (**?**):

- *Open-source* e gratuito;
- Permite realizar pesquisa de frases;
- Solução estável com boa performance e bem documentada;

No Lucene a unidade fundamental de trabalho é o documento, sendo o critério de pesquisa ele próprio um documento. Cada documento é indexado produzindo um vector

[7] `http://lucene.apache.org/java/docs/index.html`

com uma dimensão para cada termo existente (no conjunto de todos os documentos). O vector de um documento é calculado com base na frequência dos termos no documento usando como base o modelo tf-idf.

O Lucene utiliza adaptações dos modelos booleano (BM) e de espaço vectorial (VSM) nos algoritmos que implementa para realizar as pesquisas. Numa primeira fase o modelo booleano decide sobre a inclusão ou não do documento na pesquisa, e em caso afirmativo o modelo vectorial determina uma métrica de semelhança entre os dois documentos (o critério de pesquisa e o documento alvo de processamento). O calculo da semelhança é baseada no coseno entre os vectores de cada documento.

2.6 Soluções de motores de pesquisa semântica

Foi analisado um conjunto de motores de pesquisa semântica existentes actualmente. A tabela **??** resume as soluções analisadas e o resto da secção analisa as principais características de cada uma.

O Hakia [8] é uma das soluções mais proeminentes com uma abordagem original que se baseia em 3 conceitos base — *Query Detection and Extraction (QDEX)*, *Comercial Ontology* (CO) e *SemanticRank*. Esta solução não usa um método de indexação baseado em contagem de termos (como o Lucene) mas antes uma tecnologia denominada QDEX que analisa os documentos extraindo um conjunto de possíveis critérios de pesquisa onde o documento seja relevante. A figura **??** [9] ilustra a tecnologia QDEX.

Para reduzir a dimensão do espaço de pesquisas extraídas é utilizada a ontologia CO de forma a extrair apenas pesquisas com significado. A CO corresponde a uma *upper ontology* com vários níveis onde o primeiro nível corresponde a substantivos e o segundo a

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

[9] retirado de `http://company.hakia.com/new/qdex.html`

Tabela 2.1: Motores de pesquisa semântica

Solução	Dados	Abordagem	Características
Hakia	Não estruturados	Ontologia/NLP	Ontologia proprietária (*Comercial Ontology*). Pré processamento para detectar e extrair querys (QDEX). Classificação semântica de resultados.
SenseBot	Não estruturados	*text mining* e *multidocument summarization*	Extracção de conceitos semânticos de um documento.
Powerset	Estruturados	NLP	Limitado ao Wikipedia.
Cognition	Não estruturados	Mapa semântico (NLP)	Mapa semântico da língua inglesa com 24 anos de desenvolvimento.

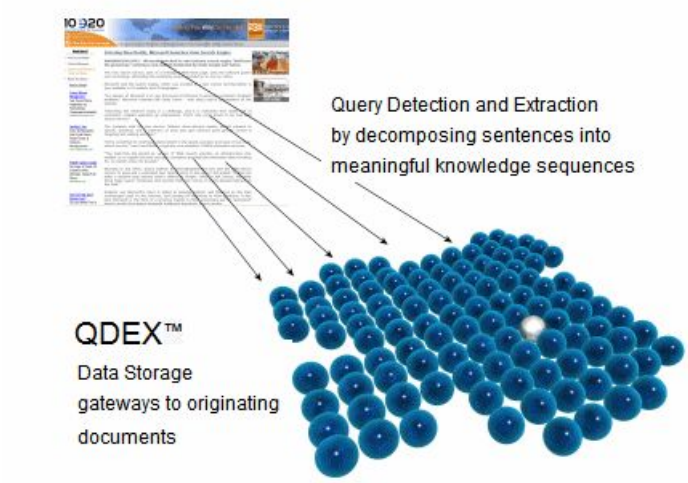

Figura 2.2: Hakia QDEX

verbos. Esta ontologia é utilizada por um algoritmo designado MAQ (*Mapping of Anticipated Queries*) (**?**) associado ao processo de extracção de pesquisas (QDEX), para limitar o conjunto das pesquisas extraídas.

Esta solução contém ainda uma tecnologia original para a classificação dos resultados. Um documento é classificado por um algoritmo baseado na analise semântica das pesquisas pré-processadas, e o critério de pesquisa introduzido pelo utilizador. Esta análise compreende várias etapas incluindo análise morfológica e sintáctica.

Toda a tecnologia existente nesta solução é proprietária não incorporando qualquer *framework open-source*.

O Sensebot [10] apresenta ao utilizador um sumário sobre o critério de pesquisa em vez de um conjunto de *links*. Este sumário é construído aplicando um algoritmo proprietário de NLP a um conjunto de documentos previamente obtidos recorrendo a um motor de pesquisa convencional.

O Powerset [11] é uma solução direccionada a domínios específicos e estruturados e actualmente apenas suporta pesquisas sobre os documentos do *site wikipedia*. Utiliza um algoritmo proprietário de NLP para tentar responder a uma pergunta ou em alternativa apresentar um conjunto de factos relevantes para a resposta.

O Cognition [12] constitui um processador de NLP para a língua inglesa. A grande vantagem desta solução reside no facto de possuir um motor de NLP com um vasto historial de investigação e aperfeiçoamento.

Todas as soluções analisadas encontram-se em desenvolvimento, estando a maior parte delas em versões beta. Destaca-se a solução Hakia que utiliza um algoritmo baseado

[10] http://www.sensebot.net/

[11] http://labs.powerset.com/

[12] http://www.cognition.com/

numa ontologia para reduzir a complexidade associada ao processamento semântico. Os exemplos apresentados baseiam-se em diferentes plataformas conceptuais mas todas têm em comum mecanismos mais ou menos evoluídos de NLP como forma de contextualizar o critério de pesquisa.

A proposta desta dissertação distingue-se das abordagens analisadas em dois aspectos fundamentais. Em primeiro lugar pretende-se utilizar técnicas simples baseadas na inferência sobre ontologias para contextualizar o critério de pesquisa e não técnicas avançadas de NLP. Em segundo lugar pretende-se utilizar um domínio específico para contextualizar o critério de pesquisa e não um domínio genérico.

Capítulo 3

Modelo

O modelo proposto baseia-se na integração de um analisador semântico com um motor de recuperação de informação segundo o diagrama da figura **??**

Para incorporar conhecimento semântico na pesquisa de informação, foi introduzido o conceito designado por Descrição Semântica (DS). A DS contém a informação do critério de pesquisa contextualizado através do conhecimento sobre o domínio existente na ontologia. Uma DS consiste num conjunto de Frases Semânticas (FS) com cada FS a conter uma lista ordenada de Termos Semânticos (TS). Um TS é uma estrutura que contém informação obtida da ontologia para um determinado termo do critério de pesquisa.

A DS é construída e em seguida expandida por aplicação de um conjunto de regras, originando uma colecção de critérios de pesquisa alternativos. Cada critério alternativo tem associado um coeficiente semântico calculado durante a expansão da DS que indica a semelhança entre o critério original e o critério alternativo. Por fim cada critério alternativo é pesquisado sintacticamente recorrendo a um motor de recuperação de informação.

Cada resultado da pesquisa tem associado um coeficiente sintáctico obtido da pesquisa. A ordenação dos resultados baseia-se na relevância,que se calcula ponderando

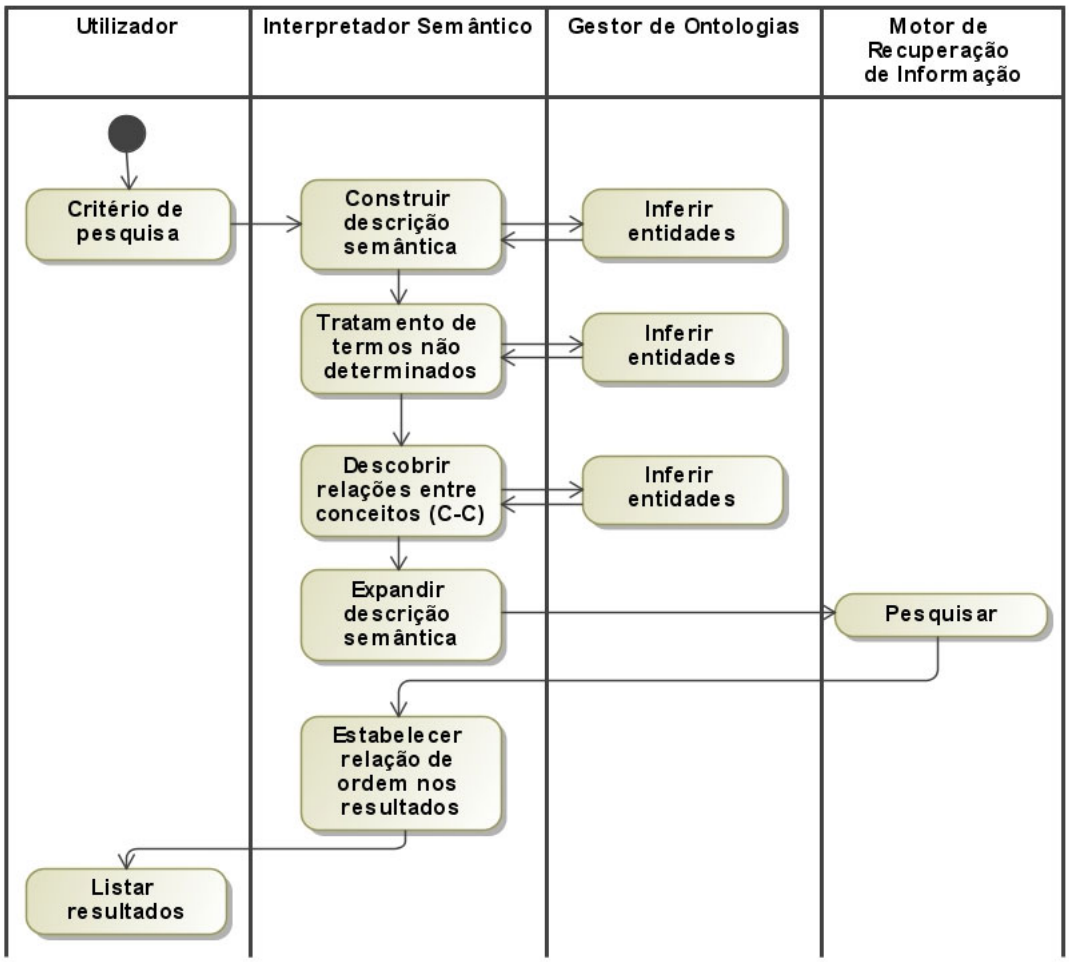

Figura 3.1: Modelo Conceptual

o coeficiente sintáctico de cada resultado de pesquisa pelo coeficiente semântico da expansão correspondente.

O modelo prevê ainda a composição de várias ontologias como meio de aumentar a abrangência do conhecimento. As ontologias são verificadas por um motor de inferência de forma a garantir a sua consistência.

3.1 Exemplo

Para ilustrar o modelo proposto considere-se o domínio 'Música', e o critério de pesquisa

Pearl Jam first concert

seja a frase retirada do documento sobre a banda *Pearl Jam* no *Wikipedia*:

The band played its first official show at the Off Ramp Café[..]

Estas duas frases tem uma relação sintáctica muito fraca (apenas o termo *first* é comum) pelo que numa pesquisa tradicional a frase apresentaria uma baixa relevância para o critério. No entanto é possível associar o conceito 'Band' ao individuo 'Pearl Jam' (com uma probabilidade associada) se o documento contiver uma frequência alta do termo 'Pearl Jam'. Recorrendo a uma ontologia pode saber-se que 'Pearl Jam' é uma 'Band' e que 'Show' e 'Concert' são conceitos equivalentes. Recorrendo ao conhecimento do domínio expresso na ontologia deve ser possível assinalar a frase como tendo alta probabilidade de satisfazer o critério de pesquisa, ainda que nenhum dos termos do critério de pesquisa (excepto first) conste na frase. Para tal é necessário inferir sinónimos e relações entre conceitos da ontologia.

3.2 Descrição Semântica (DS)

Antes de detalhar cada passo da construção da DS, considere-se a visão de alto nível do processo:

1. Obter uma lista de termos a partir do critério de pesquisa original;
2. Para cada termo:

   (a) Obter o conjunto dos Termos Semânticos;

   (b) Adicionar os TSs às Frases Semânticas existentes (e criar novas FS quando necessário);

3. Para cada Frase Semântica:

   (a) Tentar construir TSs determinados a partir de grupos contíguos de TS's não determinados (c.f. secção **??**);

   (b) Enriquecer os TSs com a inferência C-C (c.f. secção **??**);

3.2.1 Forma Canónica

Criar uma DS corresponde a traduzir o critério de pesquisa para um domínio descrito por uma ontologia. Para efectuar esta tradução é necessário estabelecer uma associação entre os termos e as entidades da ontologia. Associar directamente os termos do critério de pesquisa aos nomes das entidades na ontologia apresenta no entanto as seguintes limitações:

1. Podem existir diferenças de forma entre o termo e o nome da entidade (e.g. 'Band' e 'band');

2. Podem existir diferenças morfológicas entre o termo e o nome da entidade (e.g. 'Band' e 'bands');

3. O nome de uma entidade pode compreender mais do que um termo (e.g. 'LeadSinger' e 'Lead Singer');

Estas limitações podem ser ultrapassadas, usando a forma canónica tanto dos termos (no critério de pesquisa) como do nome das entidades (na ontologia). O conceito de forma canónica de uma palavra foi desenvolvido no trabalho de (**?**) e resulta de um processo de remoção de formas morfológicas comuns e inflexões no final das palavras com

vista à normalização das palavras. Esta normalização permite reduzir várias palavras semelhantes a um único termo (e.g. as palavras *connect*, *connected*, *connecting*, *connection* e *connections* apresentam todas a mesma forma canónica *connect*).

Para não impor regras na modelação das ontologias, o modelo propõe efectuar um pré-processamento na ontologia para associar a cada entidade a forma canónica do seu nome. Os nomes das entidades que sejam constituídos por mais de um termo (segundo a notação *camelCase*) são primeiramente separados em termos, depois obtida a forma canónica de cada termo e por fim concatenadas as formas canónicas usando um separador (e.g. *underscore*). Desta forma torna-se possível associar múltiplos termos a uma entidade da ontologia.

3.2.2 Frases Semânticas (FS)

A linguagem OWL exige que cada entidade numa ontologia tenha um nome único. Não obstante é necessário considerar a possibilidade de existirem múltiplas entidades com a mesma forma canónica. Esta multiplicidade pode ocorrer devido aos seguintes factores:

- Apenas existe garantia de unicidade para os nomes qualificados das entidades, sendo possível que duas entidades pertencentes a ontologias diferentes (e portanto com diferentes espaço de nomes) tenham o mesmo nome e diferentes espaço de nomes;
- Duas entidades com nomes diferentes podem apresentar a mesma forma canónica.

Nestes casos um termo do critério de pesquisa terá associado várias entidades da ontologia configurando um cenário onde o mesmo termo tem mais que um significado, devendo os diferentes significados originar diferentes critérios de pesquisa alternativos. Para lidar com este tipo de ambiguidade foi introduzido o conceito de Frase Semântica (FS).

Cada FS corresponderá a uma interpretação do conjunto dos termos do critério de pesquisa, tornando-se desta forma possível a coexistência de entidades com a mesma forma canónica. Se cada termo do critério de pesquisa apenas tiver associado uma entidade na ontologia, a DS conterá apenas uma FS com uma lista ordenada de TSs. Se um termo tiver duas entidades associadas, a DS conterá duas FSs cada uma com uma lista ordenada de TSs que diferirão apenas no TS do termo com duas entidades.

Uma FS é assim um conjunto ordenado de Termos Semânticos (TS). Um Termo Semântico contém a informação associada a um termo (classe, classes equivalentes, propriedades, etc.). A utilização de FSs permite criar contextos semânticos independentes, resolvendo o problema da ambiguidade de formas canónicas podendo também ser útil num cenário de utilização das ontologias em modo independente numa futura extensão do modelo.

3.2.3 Termos Semânticos (TS)

Cada termo, na forma canónica, dá origem a um, ou mais, TS. É possível que existam termos que não se conseguem associar a nenhuma entidade na ontologia. Cada um desses termos dá origem a um TS 'Não Determinado' sem informação semântica associada. Um TS será 'Determinado' se existir informação na ontologia associada ao termo. Cada termo origina um TS de um tipo que depende da informação existente na ontologia de acordo com a tabela **??**.

	Determinado			Não Determinado
Ontologia	Conceito	Indivíduo	Propriedade	—
TS	ClassTerm	IndividualTerm	PropertyTerm	GenericTerm

Tabela 3.1: Tipos de TS existentes numa DS

Como exemplo o critério de pesquisa '*Pearl Jam first concert*' originaria os TSs não determinados '*pearl*', '*jam*' e '*first*' e o TS determinado '*concert*'.

3.2.4 Tratamento de TSs não determinados

É possível que dois ou mais TSs não determinados representem um TS determinado (desde que apareçam juntos). Estas situações são detectadas e corrigidas após a criação das FSs. Como exemplo suponhamos a existência de uma FS com a lista de TSs $[t_0, t_1, t_2, t_3]$ e sejam t_1 e t_2 TSs não determinados. Se na ontologia existir uma entidade com a forma canónica t_1+ '_' $+t_2$ a lista de TSs da FS fica $[t_0, t_4, t_3]$ onde $t_4 = t_1+$ '_' $+t_2$ (TS determinado encontrado na ontologia).

O critério de pesquisa apresentado na secção anterior originaria os TSs determinados `IndividualTerm`(PearlJam) e `ClassTerm`(Concert) e o TS não determinado `GenericTerm`(first).

3.2.5 Informação associada aos TSs

Cada TS tem associado um conjunto de informação que depende do seu tipo e que é obtida recorrendo a um motor de inferência. Na tabela **??** cada inferência determinística e probabilística, está associada às deduções que se pretendem para cada tipo de TS. As colunas C, I, P e C-C representam `ClassTerm`, `IndividualTerm`, `PropertyTerm` e entre `ClassTerm`)

A Inferência Determinística (c.f. tabela **??** linhas 2 a 5) é usada para inferir classes equivalentes, subclasses, classes de indivíduos e propriedades equivalentes. Desta forma um TS contém um conjunto de informação que irá ser usada na fase de expansão para determinar os critérios de pesquisa alternativos. A inferência de propriedades C-C foi assim designada por se basear na ideia de inferir propriedades para pares de classes. Sejam A e B duas classes tais que A aparece antes de B na FS e p uma propriedade pertencente ao conceito

$$ObjectProperty \sqcap \exists domain.A \sqcap \exists range.B$$

			TSs			
			C	**I**	**P**	**C-C**
Inferência	**Determinística**	Classes Equivalentes	x	x		
		Subclasses	x			
		Classe		x		
		Propriedades Equi.			x	
		Propriedades				x
	Probabilística	Classes Semelhantes	x	x		

Tabela 3.2: Tipos de Inferência por TS

então p representa um verbo relacionado com o critério de pesquisa e deve ser incluído na informação semântica associada ao TS A. Desta forma são adicionadas todas as propriedades existentes entre cada par de classes. Por exemplo se uma FS contiver os TSs 'Band' e 'Show' e estiver definida na ontologia uma propriedade 'performed' entre 'Band' e 'Show', essa propriedade vai ser adicionada ao TS 'Band'.

A Inferência Probabilística é usada para modelar a incerteza e ambiguidade inerente a qualquer conhecimento. O conceito de inferência probabilística é importante porque introduz a noção de conceitos semelhantes permitindo relaxar a noção de conceitos equivalentes. Dois conceitos consideram-se semelhantes, se existir um axioma de equivalência probabilística entre os dois, e interpreta-se a probabilidade associada à equivalência como indicando o grau de semelhança entre os dois conceitos.

Para suportar inferência probabilística é necessário usar uma ontologia probabilística que é definida como uma extensão da ontologia (clássica) sobre a qual se declaram axiomas probabilísticos.

3.3 Expansão da DS

A expansão de uma DS consiste na expansão de cada uma das suas FSs. Cada FS representa o critério de pesquisa original num contexto semântico. No entanto uma vez que o modelo baseia as suas pesquisas num motor de recuperação de informação tradicional (não semântico) é necessário expandir a FS num conjunto de critérios de pesquisa alternativos. Esta expansão tem como objectivo a criação de critérios alternativos, semanticamente relacionados mas sintacticamente distintos do critério de pesquisa original. Para expandir uma FS é necessário percorrer a sua lista de TSs e construir uma árvore com as várias hipóteses semânticas de cada TS. A figura **??** descreve este processo.

Um TS do tipo `ClassTerm` conterá o nome da classe mas também as classes equivalentes e semelhantes (e as suas subclasses). Cada uma destas classes corresponderá a uma hipótese na criação de um critério de pesquisa alternativo. A informação contida em cada TS é apresentada na tabela **??**.

Uma expansão constitui um caminho na árvore de hipóteses de expansão da FS tendo um critério de pesquisa alternativo e um coeficiente semântico associado. No final da expansão a FS conterá um conjunto com todas as expansões obtidas. Cada passo na expansão da FS influência o coeficiente semântico associado à expansão. Ou seja se por exemplo uma das hipóteses na expansão implicar substituir uma classe por outra semelhante, essa substituição irá diminuir o coeficiente semântico de forma inversamente proporcional à semelhança.

3.3.1 Substituição de indivíduos por classes

Para os TSs do tipo `IndividualTerm` uma das hipóteses na construção da expansão é a de substituir o individuo pela sua classe (e.g. 'PearlJam' por 'Band'). Esta substituição implica no entanto, determinar a probabilidade da classe representar o indivíduo. Ou seja, ao pesquisar o critério '*Band first Concert*' é necessário apurar qual a probabilidade de o

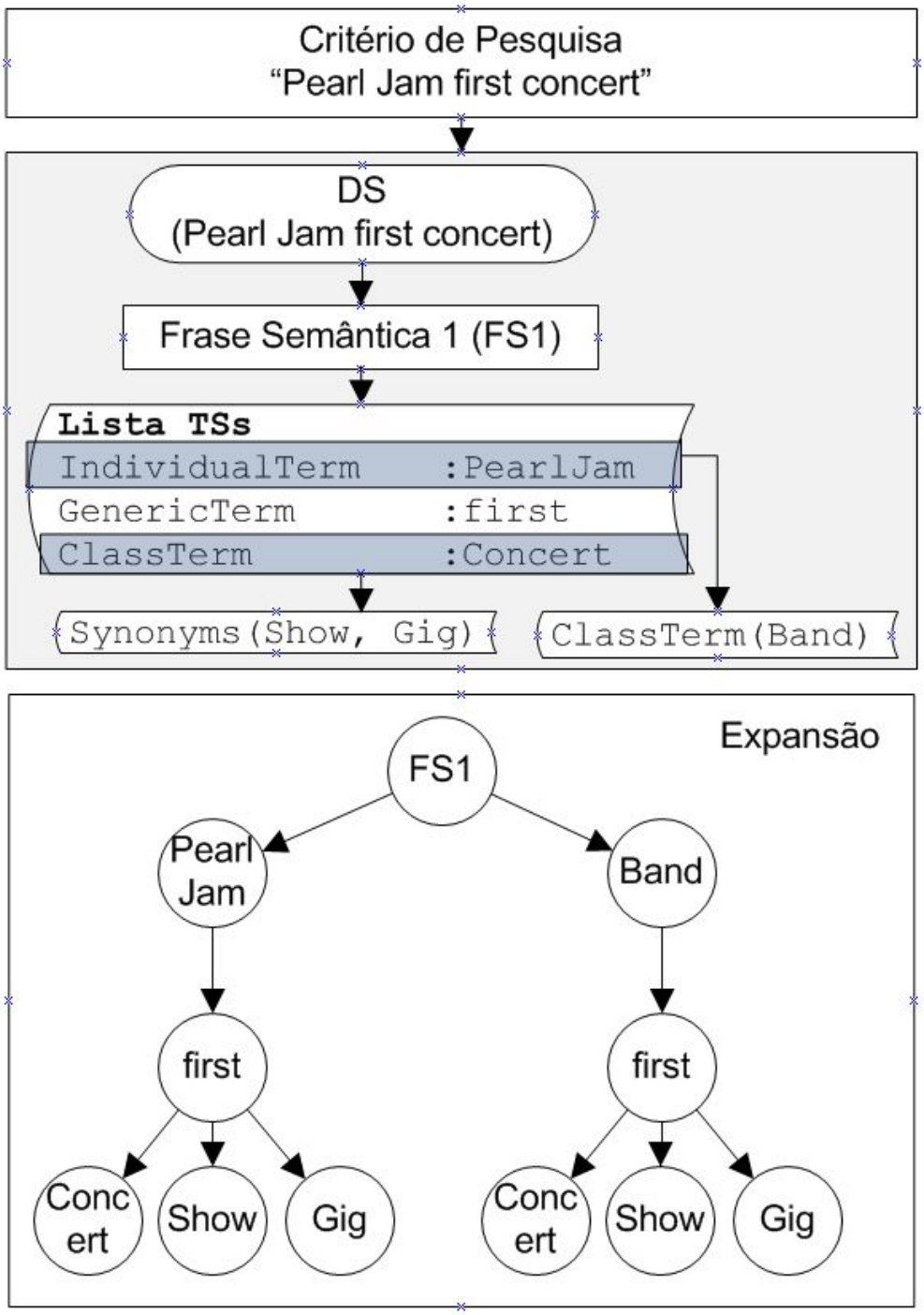

Figura 3.2: Expansão de uma Frase Semântica

Tipo de TS	Informação	Descrição
ClassTerm	Sinónimos	Lista de classes com a respectiva semelhança (1=equivalência).
	Subclasses	Lista de subclasses.
	Propriedades	Lista de propriedades obtidas pela inferência C-C.
	Individuo	Individuo que deu origem à classe.
PropertyTerm	Sinónimos	Lista de propriedades equivalentes.
	Domínio	Classe que representa o domínio da propriedade.
	Contradominio	Classe que representa o contradomínio da propriedade.
IndividualTerm	Classe	Classe do individuo.

Tabela 3.3: Informação associada a cada TS

termo 'band' se referir ao individuo 'PearlJam'. Assim foi criado o conceito de **lista de substituições** que indica para cada expansão as substituições a avaliar na fase de pesquisa. Sempre que numa expansão ocorra este tipo de substituição, é adicionada uma entrada à lista de substituições da expansão.

3.3.2 Sub-expansões

Uma vez que o motor de recuperação de informação vai pesquisar frases, se algum termo do critério de pesquisa alternativo não existir num documento, então esse documento nunca irá ser contemplado nos resultados. Ou seja, um critério de pesquisa de por exemplo 4 termos não irá considerar frases que tenham apenas 3 dos termos. Para ultrapassar esta limitação existe um processo de geração das sub-expansões, que constrói o conjunto

dos subconjuntos dos termos de uma expansão. Esta abordagem tem no entanto a desvantagem de potenciar um aumento elevado do número de critérios alternativos a serem pesquisados. Por outro lado a relevância de um subconjunto dos termos de uma expansão será tanto menor quanto menor for o número de termos, uma vez que a eliminação de termos da expansão diminui a sua semelhança semântica com o critério de pesquisa original.

A geração e pesquisa de sub-expansões permite considerar frases 'incompletas' e assim aumentar a capacidade de detecção de resultados relevantes, no entanto é necessário ter em consideração que uma sub-expansão tem sempre um maior potencial (do que a expansão que lhe deu origem) para obter uma melhor classificação. Isto resulta do facto de que qualquer resultado da pesquisa de uma expansão será sempre também um resultado da sub-expansão, não sendo o inverso verdadeiro. Ou seja, dada uma FS x tem-se que $pesquisa(expansão(x)) \subseteq pesquisa(subExpanção(X))$.

Por outro lado a classificação atribuída pelo Lucene depende do número de frases encontradas no documento e da semelhança com a frase pesquisada (c.f. secção **??** na página **??**) o que faz com que uma sub-expansão tenha sempre uma classificação melhor. Para mitigar o facto de o Lucene classificar sempre melhor as sub-expansões e tendo em conta o facto de uma sub-expansão ser sempre uma pior aproximação do critério de pesquisa original, o interpretador semântico diminuí o coeficiente semântico das sub-expansões. Esta diminuição é calculada usando a formula $(\frac{X}{Y})^{\Delta}$ em que X é o número de termos da sub-expansão, Y o número de termos da expansão e Δ é um parâmetro configurável (SUBEXPANSION-PONDERATION-FACTOR).

3.4 Pesquisa de critérios alternativos

Depois de obtido o conjunto de expansões para uma DS (expansões de todas as FSs) é necessário pesquisar nos documentos estas expansões. A pesquisa de cada expansão deverá ser feita usando um motor de recuperação de informação tradicional com capacidade

para pesquisar frases. O objectivo não é o de encontrar documentos com alta frequência dos termos existentes na expansão mas antes o de encontrar documentos com uma frase o mais próxima possível da expansão. É necessária alguma flexibilidade no critério de identificação da frase uma vez que ao considerarem-se apenas frases idênticas limita-se muito a pesquisa. É por isso necessário que o motor de recuperação de informação seja capaz de pesquisar frases e devolver uma métrica da semelhança entre a frase pesquisada e a frase encontrada. A esta métrica foi dada a designação de **coeficiente de indexação**.

Para além da pesquisa das expansões é necessário também avaliar as substituições que possam existir em cada expansão. Sempre que uma expansão contenha uma lista de substituições não vazia é necessário pesquisar os termos da lista para determinar os coeficientes de substituição.

No final da fase de pesquisa cada expansão contém uma lista de resultados de pesquisa contendo cada um, um documento, um coeficiente de indexação e um coeficiente semântico. O coeficiente semântico é calculado como o produto do coeficiente semântico da expansão pelos coeficientes de substituição apurados para a expansão.

3.5 Ordenação de resultados de pesquisa

A fase da pesquisa é orientada à expansão. Cada expansão é pesquisada originando um conjunto de resultados de pesquisa. A fase da apresentação dos resultados é orientada ao documento. Uma vez que um documento pode constar em diversos resultados de pesquisa é necessário agregar os resultados de pesquisa em função dos documentos. Esta agregação permite obter para cada documento uma lista de resultados de pesquisa.

Os resultados são apresentados na forma de uma lista de documentos ordenada por relevância. A relação de ordem entre os documentos é estabelecida pela pontuação de cada documento que é dada pelo resultado de pesquisa com maior pontuação. A pontuação

de um resultado de pesquisa é calculada pelo produto dos coeficientes de indexação e semântico.

Outra hipótese para calcular a pontuação de um documento seria considerar a média das pontuações de todos os resultados de pesquisa acima de um valor mínimo. Desta forma, quanto mais expansões considerassem o documento relevante maior seria a pontuação do documento. No entanto esta aproximação desvia-se da hipótese formulada de pesquisar frases pelo que foi preterida.

Capítulo 4

Concretização

Este capítulo descreve o protótipo desenvolvido para validar o modelo e analisa os principais desafios e propostas identificadas durante a sua implementação.

4.1 Visão geral

A aplicação é composta por 4 componentes principais:

- Descrição Semântica
- Interpretador Semântico
- Gestor de Ontologias
- Gestor de Índices

A figura **??** descreve o diagrama de classes desenvolvido e ilustra os vários componentes e as relações entre eles. Este diagrama servirá de referência durante o resto do capítulo.

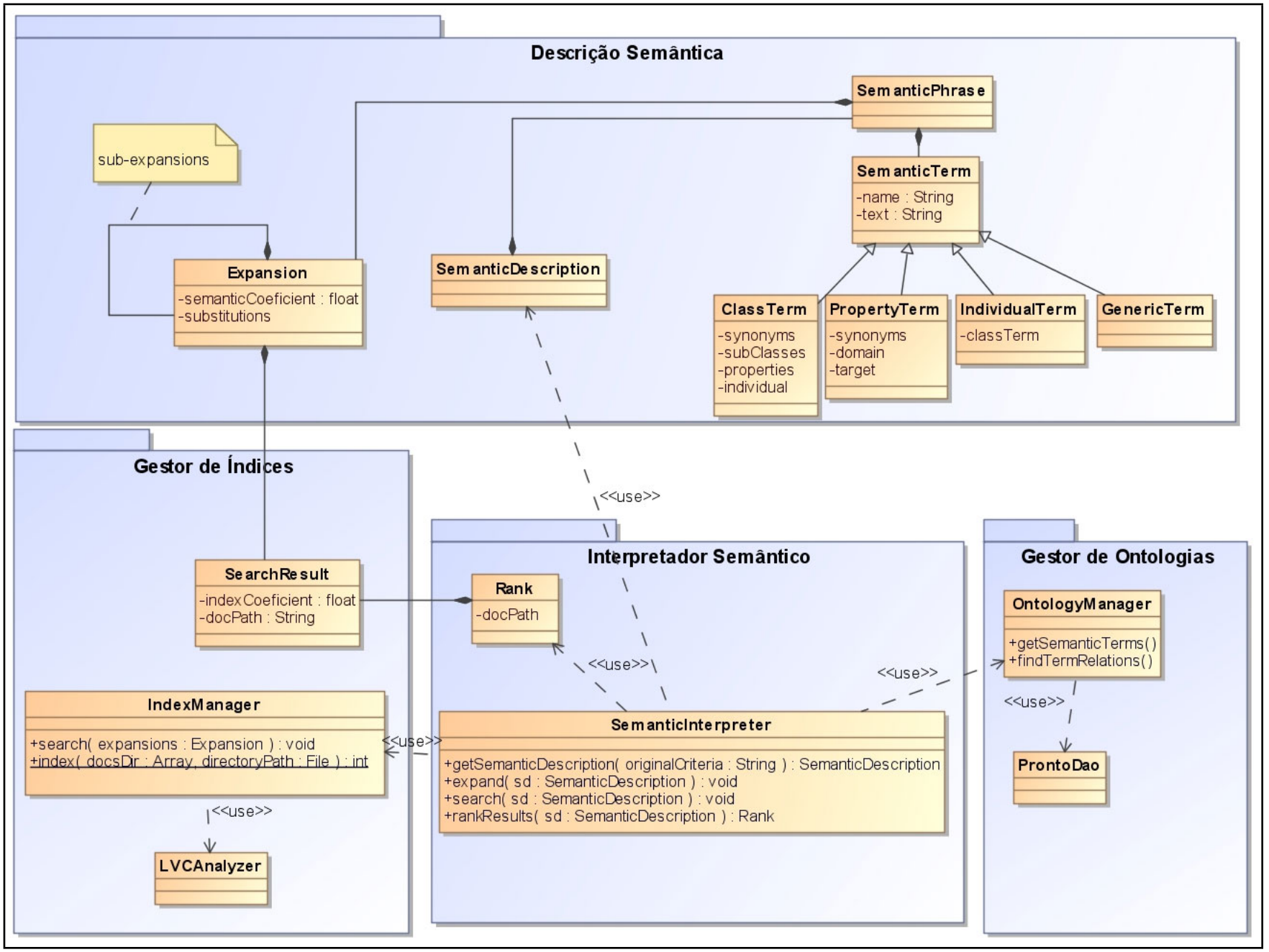

Figura 4.1: Diagrama de classes do protótipo

O Interpretador Semântico expõe uma API que permite realizar as operações necessárias à implementação do modelo. Este componente delega no Gestor de Ontologias todas as tarefas de interacção com as ontologias (pré-processamento, composição, verificação de consistência e inferência) e no Gestor de Índices toda as tarefas relacionadas com indexação e pesquisa de documentos.

O Gestor de Ontologias recorre à API OWL para manipulação das ontologias, ao motor de inferência Pellet para verificar consistência e realizar inferência determinística e ao sub-componente `ProntoDao` para realizar inferência probabilística. O `ProntoDao` é um adaptador (*wrapper*) sobre o motor de inferência probabilística Pronto. O Gestor de Índices foi implementado como um *wrapper* sobre o motor de recuperação de informação Lucene.

A execução da aplicação compreende duas fases distintas:

1. A inicialização, onde são carregadas e pré-processadas as ontologias e inicializados os componentes da aplicação;
2. A pesquisa, onde é construída a DS, efectuada a sua expansão, pesquisados os critérios alternativos e ordenados e apresentados os resultados.

4.2 Inicialização e pré-processamento

O modelo proposto utiliza ontologias determinísticas e probabilísticas como base de conhecimento para suportar a construção da DS.

Ontologias Determinísticas

Na fase de inicialização o Gestor de Ontologias carrega o conjunto de ontologias configuradas na aplicação efectuando a sua composição (classe `OWLOntologyMerger` da

API OWL). Desta composição resulta uma única ontologia que é carregada no motor de inferência Pellet sendo depois verificada a sua consistência e realizada a sua classificação. A classificação da ontologia na fase de inicialização permite inferir antecipadamente relações de subsunção, equivalência e pertença. Desta forma toda a inferência (determinística) é realizada à priori tornando a construção da DS mais eficiente.

Depois de obtida a ontologia é efectuado o pré-processamento que associa cada entidade à sua forma canónica (c.f. secção **??** na página **??**) usando o analisador `LVCAnalyzer` (c.f. secção **??** na página **??**). Nesta altura é criada na ontologia uma entidade do tipo `AnnotationProperty` (com o nome canonicForm) para efectuar a associação da forma canónica à entidade através de um axioma de asserção de anotação. Uma vez que é necessário anotar todas as entidades (classes, propriedades e indivíduos) na ontologia com a sua forma canónica, é necessário utilizar uma anotação e não uma propriedade OWL (do tipo `DataProperty` ou `ObjectProperty`) porque a linguagem OWL-DL apenas permite a associação de propriedades entre indivíduos (usando propriedades `ObjectProperty`) e a associação de literais a indivíduos (usando propriedades `DataProperty`).

Ontologias Probabilísticas

O Gestor de Ontologias delega na componente `ProntoDao` todas as tarefas relacionadas com o carregamento, pré-processamento e inferência de ontologias probabilísticas. Este componente utiliza a API Pronto para inferir classes semelhantes (equivalentes com uma probabilidade) sobre numa ontologia probabilística.

Note-se que a API Pronto se encontra em desenvolvimento e foi necessário algum esforço de integração para permitir a sua utilização para o fim proposto no modelo. Na exploração desta API foram detectadas as seguintes limitações:

1. O desempenho era aceitável apenas para um número reduzido de axiomas (cerca

de 12).

2. A abordagem estava focada na subsunção e não fornecia formalismos para a definição e inferência de equivalência.

3. Apenas eram fornecidos mecanismos de inferência da probabilidade entre duas classes não existindo mecanismos de inferência de todas as subclasses (ou classes equivalentes) de uma dada classe.

Estas limitações, assim como os objectivos deste trabalho foram expostos ao investigador responsável pelo desenvolvimento da API Pronto (Pavel Kilinov). Em consequência foi disponibilizada uma versão mais recente (ainda em desenvolvimento) com melhorias significativas de desempenho (desempenho aceitável para cerca de 700 axiomas).

O segundo problema, inesperadamente, mostrou-se de resolução mais difícil. Esta limitação poderia à primeira vista parecer negligenciável uma vez que o problema da equivalência se resume formalmente ao da subsunção (pois $A \Leftrightarrow B \equiv A \sqsubseteq B \sqcap B \sqsubseteq A$). No entanto não é possível reduzir o problema da equivalência probabilística ao da subsunção probabilística de forma directa. Para permitir esta tradução, é necessário garantir que os limites superiores dos intervalos de probabilidade correspondem sempre à certeza (c.f. detalhe no apêndice **??**). Atendendo a que o modelo não exige intervalos de probabilidade, usando sempre o limite inferior, esta restrição não limita a utilização desta API no nosso modelo. Ainda assim a API não fornecia uma maneira de declarar axiomas de equivalência probabilística, pelo que foi necessário alterar a classe `ProntoLoaderUtils` para processar axiomas de equivalência declarados na ontologia convertendo-os em dois axiomas de subsunção. A explicação do desenvolvimento efectuado encontra-se no apêndice **??**.

Para inferir as classes equivalentes foi necessário criar uma nova instância do motor de inferência Pellet onde a ontologia clássica é carregada, sendo depois adicionados axiomas de subsunção de acordo com os axiomas probabilísticos definidos na ontologia probabilística. Desta forma a inferência das classes equivalentes é efectuada pelo Pellet, sendo

depois inferidas as probabilidades associadas a cada equivalência recorrendo ao Pronto. Esta abordagem tem no entanto a desvantagem de realizar a inferência da probabilidade associada a classes equivalentes. Esta operação tem um custo considerável (quando realizada pelo Pronto) pelo que foi necessário garantir a realização da inferência apenas para as classes semelhantes. Quando são pedidas (ao ProntoDAo) as classes semelhantes, é passada uma lista de classes equivalentes que são excluídas da inferência probabilística.

Foi ainda necessário modelar a ontologia probabilística. Este passo no entanto é bastante simples bastando criar uma ontologia vazia que importa a ontologia clássica (sobre a qual pretendemos estabelecer axiomas probabilísticos) e depois escrever os axiomas de equivalência probabilísticos (c.f. figura **??** no apêndice **??**).

4.3 Construção da Descrição Semântica

O primeiro problema a resolver é o de associar os termos do critério de pesquisa às entidades da ontologia. O modelo propõe reduzir os termos à forma canónica o que pode ser conseguido recorrendo a um analisador.

4.3.1 Analisador

O motor de recuperação de informação adoptado foi o Lucene cuja API permite a construir analisadores de texto que suportam a composição de filtros e *tokenizer's* de forma muito flexível para processar texto originando termos. Para esta aplicação foi criado um novo analisador — `LVCAnalyser` apresentado na listagem **??** no apêndice **??**. Este analisador é utilizado pelo:

- Lucene no processo de indexação dos documentos,
- Interpretador Semântico para processar o critério de pesquisa,

- Gestor de Ontologias para obter a forma canónica dos nomes das entidades na ontologia (no pré-processamento).

A utilização deste analisador para pré-processar a ontologia e o critério de pesquisa antes da construção da DS permite efectuar a associação entre os termos e as entidades (c.f. secção **??** na página **??**) apresentando ainda duas outras vantagens:

1. Eliminação das *stop words* que são na sua maioria artigos (o, as, ao, por, etc.) com pouca influencia nos resultados de pesquisa (uma vez que aparecem em todos os documentos) e que por essa razão são normalmente eliminadas do critério de pesquisa. A eliminação destas *stop words* numa fase anterior à construção da DS tem a vantagem acrescida de reduzir o número de TSs existentes numa DS contribuindo para diminuir o número de expansões realizadas (com consequente diminuição no número de pesquisas sintácticas realizadas).
2. Utilizar directamente na DS formas canónicas, diminuindo o processamento a realizar pelo Lucene na fase de pesquisa dos critérios alternativos.

O analisador `LVCAnalyser` cria um `TokenStream` que gera termos e que é construído recorrendo às seguintes classes fornecidas com a distribuição do Lucene:

1. `StandardTokenizer`, utiliza um conjunto de regras gramaticais (da língua inglesa) para dividir uma sequência de caracteres em termos, realizando operações adicionais como remover caracteres de pontuação e hífenes;
2. `StandardFilter`, transforma os termos produzidos pela classe `StandardTokenizer` removendo plurais e os pontos dos acrónimos;
3. `LowerCaseFilter`, transforma os caracteres maiúsculos em minúsculos;
4. `StopFilter`, elimina as *stop words*. Este filtro necessita ser configurado com uma lista de *stop words* e neste protótipo utilizamos a lista fornecida com o Lucene para o idioma inglês;

5. `PorterStemFilter`, transforma os termos segundo o algoritmo Porter stemmer (**?**). Este algoritmo modifica os termos transformado-os na sua forma canónica (c.f. **??** na página **??**).

4.3.2 Construção de Frases Semânticas

Cada termo obtido do critério de pesquisa original é passado ao Gestor de Ontologias para este recolher a informação semântica. O Gestor de Ontologias retorna uma lista de objectos `SemanticTerm`, com um objecto para cada entidade descoberta na ontologia. Se um termo originar apenas um TS este vai ser adicionado à lista de TSs de cada FS existente. Se um termo originar mais do que um TS, vão ser criadas novas FSs de acordo com o esquema da figura **??**. No instante T_0 o termo do critério de pesquisa origina um TS pelo que é criada a FS_1 e adicionado o TS. No instante T_1 apenas é retornado um TS e este é adicionado a FS_1. No instante T_2 o termo 'record' origina dois TSs ('Record' e 'recordedFor') pelo que FS_1 é duplicada originando FS_2 e de seguida cada um dos TSs é adicionado a cada uma das frases.

Para obter a lista tss de TSs a partir de um termo t, o Interpretador Semântico invoca o Gestor de Ontologias passando-lhe a forma canónica $can(t)$ do termo t. A lista tss é construída a partir de t da seguinte forma:

1. Atribuir a tss a lista vazia;
2. Obter todas as classes cuja forma canónica seja $can(t)$;
3. Para cada classe criar um objecto c do tipo `ClassTerm`, e
   (a) Adicionar a c as classes equivalentes inferidas e a lista das suas subclasses;
   (b) Adicionar a c as classes semelhantes inferidas e a lista das suas subclasses;
   (c) Adicionar a c as subclasses inferidas;
   (d) Adicionar c a tss;

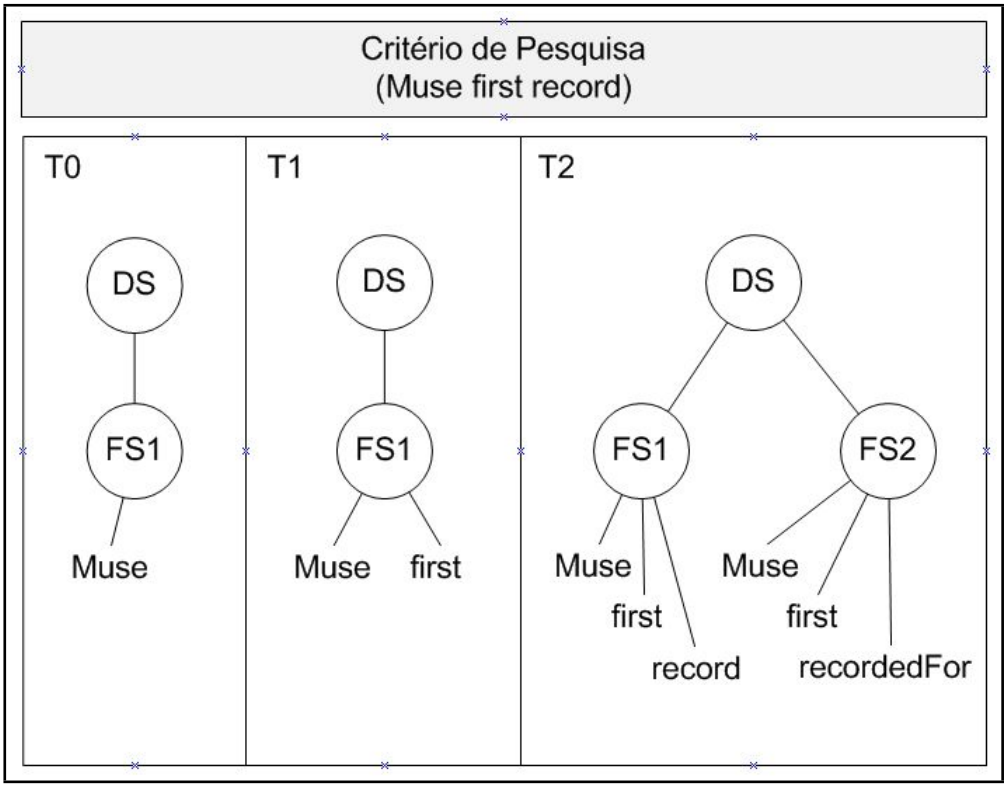

Figura 4.2: Evolução da construção de uma FS

4. Obter todas as propriedades cuja forma canónica seja $can(t)$;

5. Para cada propriedade criar um objecto p do tipo `PropertyTerm` e,

   (a) Adicionar a p as propriedades equivalentes inferidas;

   (b) Adicionar p a tss;

6. Obter todos os indivíduos cuja forma canónica seja $can(t)$;

7. Para cada indivíduo i criar um objecto do tipo `IndividualTerm` e,

   (a) Obter a classe ci de i;

   (b) Adicionar a ci as classes equivalentes inferidas;

   (c) Adicionar a ci as classes semelhantes inferidas;

(d) Adicionar i a tss;

8. Se tss for vazia criar um objecto g do tipo `GenericTerm` e adicioná-lo a tss.

A figura **??** no apêndice **??** apresenta a informação associada a cada TS.

4.3.3 Tratamento de TSs não determinados

Depois de construídas as Frases Semânticas é efectuado um processamento que resolve o problema das entidades na ontologia cujo nome tem mais do que um termo.

Uma vez que os termos do critério de pesquisa são processados um a um, não é possível detectar entidades, na ontologia, cujo nome seja constituído por mais do que um termo. Para ultrapassar esta limitação é aplicado a cada FS um algoritmo que detecta conjuntos contíguos de TSs não determinados e depois efectua combinações destes TSs para tentar detectar novos TSs determinados. O algoritmo é apresentado no apêndice **??**.

4.3.4 Inferir relações entre classes (C-C)

O último processamento efectuado na construção da DS visa enriquecer com propriedades os TSs do tipo `ClassTerm`. São adicionadas propriedades p tal que

$$ObjectProperty \sqcap \exists domain.A \sqcap \exists range.B$$

onde A e B são TSs do tipo `ClassTerm` presentes na FS e A aparece antes de B. Este processamento enriquece a DS com propriedades (verbos) semanticamente relacionadas.

Para inferir, por exemplo, todas as propriedades existentes entre as classes 'Band' e 'Show' seria necessário deduzir todas as propriedades que satisfizessem a expressão $ObjectProperty \sqcap \exists domain.Band \sqcap \exists range.Show$, no entanto o motor de inferência Pellet não permite realizar esta inferência. O problema surge do facto de 'domain' e 'range' serem propriedades definidas na linguagem RDFS não se encontrando disponíveis

Listing 4.1: Interrogação SPARQL para obter propriedades.

```
SELECT ?property ?classA ?classB
WHERE
{?property rdf:type owl:ObjectProperty .
?property rdfs:domain ?class1 .
?property rdfs:range ?class2 .
?classA rdfs:subClassOf ?class1 .
?classB rdfs:subClassOf ?class2}
```

na API OWL. A classe `PelletReasoner` (implementação de `OWLReasoner` da API OWL) define um conjunto de operações para realizar inferência sobre uma ontologia OWL e não RDFS.

Como abordagem alternativa foi considerado utilizar a linguagem de interrogação de ontologias SPARQL-DL (recorrendo à API JENA) através da interrogação apresentada na listagem **??**. No entanto existe um erro documentado (*known issue*) (**?**) que se traduz no retorno de todas as propriedades da ontologia em interrogações envolvendo 'rdfs:domain'.

Assim, para ultrapassar esta limitação foi aplicado a cada FS o seguinte algoritmo:

1. Para cada par de classes A, B na frase (B aparece depois de A).
   (a) Obter um conjunto $P1$ de propriedades cujo domínio contenha A
   (b) Obter um conjunto $P2$ de propriedades cujo contradomínio contenha B
   (c) Calcular a intersecção dos conjuntos $P = P1 \cap P2$
   (d) Adicionar ao Termo Semântico A as propriedades em P

4.4 Expansão das Frases Semânticas

Depois de construída a DS é necessário expandir cada uma das Frases Semânticas para obter um conjunto de critérios de pesquisa alternativos (c.f. secção **??** na página **??**). A expansão de uma FS é obtida por um algoritmo recursivo que recebe como parâmetros

uma lista de TSs, um coeficiente semântico, uma lista de substituições e um parâmetro de acumulação. A função vai recursivamente construindo uma expansão no parâmetro de acumulação (actualizando o coeficiente semântico e a lista de substituições) e termina quando a lista de TSs está vazia retornando a expansão apurada. Nos níveis intermédios da recursividade o algoritmo copia a lista de TSs (para poder efectuar várias invocações recursivas mantendo a lista de TSs local a cada invocação) e depois remove o primeiro termo das lista aplicando um processamento de acordo com o tipo do termo. A explicação detalhada do algoritmo encontra-se no apêndice **??**.

4.4.1 Sub-expansões

O Interpretador Semântico contém um parâmetro (USE-SUB-EXPANSIONS) que permite configurar a utilização ou não de sub-expansões e um parâmetro (SUB-EXPANSION-MIN-LENGTH) que controla o número mínimo de termos que uma sub-expansão tem de ter. Desta forma alivia-se a explosão do número de expansões e evita-se a criação de expansões com significado semântico reduzido.

Ao gerar todas as sub-expansões possíveis para cada expansão pode obter-se sub-expansões idênticas (e.g. as expansões 'Band first Show' e 'PearlJam first Show' ambas originam a sub expansão 'first show'). O Interpretador Semântico garante que no final do processo de geração das sub-expansões, não existem sub-expansões idênticas.

4.5 Gestor de Índices (Lucene)

Este componente encapsula os processos de indexação e pesquisa dos documentos e foi desenhado como um *wrapper* do Lucene permitindo um desacoplamento da aplicação em relação ao Lucene. Ou seja, alterar o motor de recuperação de informação apenas implica alterar o seu adaptador. A indexação dos documentos é realizada de forma independente da aplicação e as pesquisas sintácticas são realizadas a pedido.

4.5.1 Indexação de documentos

A Indexação dos documentos é realizada pelo Lucene (classe `IndexWriter`) com um limite fixado nos 25000 termos por campo indexado para desta forma conseguir indexar todo o texto de cada documento. São criados dois campos para cada documento, um contendo o nome do ficheiro do documento (que não é indexado), o outro, que é indexado, contendo o texto do documento. A indexação dos documentos é realizada de forma independente da aplicação e é guardada em ficheiros. Quando se inicializa o Gestor de Índices é necessário indicar uma directoria que contém os ficheiros de indexação previamente criados. O Gestor de Índices pode ser configurado para carregar os índices em memória ou utilizar directamente os ficheiros.

4.5.2 Pesquisa das expansões

A pesquisa das expansões tem como objectivo encontrar documentos com frases semelhantes à expansão. Para tal é utilizada a pesquisa de frases (classe `PhraseQuery`), em vez de termos isolados, permitindo controlar alguns parâmetros importantes de pesquisa.

O parâmetro PHRASE-SLOP (Lucene) permite controlar o critério usado para comparar frases. Este parâmetro indica o número máximo de passos permitidos para converter uma frase noutra sendo que um passo corresponde a uma troca de termos. Desta forma o Lucene aplica passos sucessivos que trocam a ordem dos termos de forma a construir a frase pesquisada. A frase no documento tem de conter os termos da frase pesquisada embora não precisem aparecer na mesma ordem em que aparecem no critério de pesquisa. Para além disso, desta forma, um documento com uma frase que contenha termos intermédios, não existentes no critério de pesquisa, não é automaticamente excluído dos resultados. Assim, é possível considerar duas frases semelhantes e atribuir um grau de semelhança (quanto mais passos são necessários na transformação menor a semelhança) obtendo uma métrica gradual e não puramente binária (de igual ou diferente). Uma vez que os passos necessários para transformar uma frase noutra aumenta (potencialmente)

com o número de termos da frase, o parâmetro PHRASE-SLOP não é estático mas sim uma função do tamanho da frase pesquisada. O Gestor de Índices contém o parâmetro PHRASE-SLOP-LENGTH-FACTOR que é usado para calcular o PHRASE-SLOP a usar na pesquisa de cada expansão de acordo com a formula $criteriaLength\times$ PHRASE-SLOP-LENGTH-FACTOR. O Gestor de Índices contém ainda o parâmetro PHRASE-SLOP-LENGTH-MIN que controla o valor mínimo de PHRASE-SLOP.

4.6 Ordenação de resultados

Depois de criar a DS, efectuar as expansões e pesquisar os critérios alternativos, é necessário processar os resultados de forma a ordenar os documentos. Esta ordenação é obtida na forma de uma lista ordenada de objectos do tipo `Rank`. Cada objecto deste tipo contém o nome de um documento e uma lista de objectos `SearchResult` que reúne todos os resultados obtidos para o documento na fase de pesquisa (em todas as FSs englobando as expansões e sub-expansões). A relação de ordem é calculada com base na pontuação de cada documento (`Rank`). A pontuação do documento é obtida percorrendo todos os resultados e determinando aquele com maior pontuação individual. A pontuação individual de cada resultado é calculada pelo produto dos coeficientes semânticos e de indexação. Cada resultado contém um coeficiente semântico calculado durante a fase de expansão e um coeficiente de indexação obtido na fase de pesquisa.

Capítulo 5

Validação

Este capítulo apresenta os testes realizados e os processos e resultados da validação do modelo.

5.1 Processo de Validação

A validação do modelo proposto foi realizada com base na aplicação desenvolvida tendo sido avaliadas cada uma das hipóteses formuladas. Recorda-se (c.f. secção **??**) que as hipóteses são:

Hipótese 1 - A adopção de ontologias suporta a composição modular de conceitos e relações permitindo endereçar diferentes domínios;

Hipótese 2 - Recuperar informação incorporando relações semânticas permite obter resultados que transcendam a mera semelhança sintáctica;

Hipótese 3 - Uma pesquisa baseada em frases que incorpore inferência sobre ontologias permite construir um modelo com uma baixa complexidade (e.g., em relação ao NLP).

Para validar a primeira hipótese foram utilizadas duas ontologias, uma criada fora do âmbito deste trabalho e outra criada no âmbito deste trabalho. Foi avaliada a capacidade da aplicação para verificar a consistência da ontologia obtida pela composição das duas e utilizá-la para realizar os processos de inferência subjacentes ao processamento semântico. O endereçamento de diferentes domínios foi validado recorrendo a uma terceira ontologia para verificar o correcto funcionamento da aplicação, nomeadamente na construção da DS, num domínio/ontologia diferente da utilizada na fase de desenvolvimento.

Para validar a segunda e a terceira hipóteses foi reunido um conjunto de documentos, sobre o domínio explorado (música) e foram elaborados alguns critérios de pesquisa de forma a efectuar pesquisas utilizando a aplicação desenvolvida. Os resultados obtidos foram comparados com os resultados obtidos numa pesquisa tradicional (quer para pesquisa de termos quer de frases) com o motor de pesquisa Lucene. A comparação dos resultados permitiu obter conclusões relativamente aos ganhos de precisão obtidos pela conjugação da pesquisa de frases e da incorporação de informação semântica no critério de pesquisa.

A terceira hipótese foi ainda validada efectuando uma análise sobre o desempenho e escalabilidade da aplicação desenvolvida para diferentes configurações do modelo. Esta análise avalia o impacto dos vários componentes no desempenho da aplicação nomeadamente a dimensão da ontologia usada e os vários parâmetros de configuração.

Neste trabalho foi escolhido o universo da música como domínio e a ontologia utilizada representa conhecimento sobre bandas, concertos e trabalhos musicais (*albums*, CDs, etc).

5.2 Hipótese 1 - adopção de ontologias

'*A adopção de ontologias suporta a composição modular de conceitos e relações permitindo endereçar diferentes domínios.*'

Para validar a composição modular de conceitos e relações foram utilizadas duas ontologias, a MO - Music Ontology (**?**), e a LVCMO - LVC Music Ontology desenvolvida no âmbito deste trabalho. Para validar o endereçamento de diferentes domínios foi utilizada a ontologia MO - Movie Ontology (**?**).

5.2.1 Verificação de consistência

É necessário validar que a verificação de consistência é correctamente realizada para a composição de ontologias pois duas ontologias consistentes podem gerar, quando compostas, uma ontologia inconsistente. Para efectuar esta validação foram construídas três ontologias.

- A ontologia M1 define uma hierarquia simples de classes ($Album \sqsubseteq \top, \quad Artist \sqsubseteq \top, \quad Singer \sqsubseteq Artist, \quad NewSinger \sqsubseteq Singer$),
- A ontologia M2 (figura **??**) importa a ontologia M1 e adiciona o axioma $Artist \sqsubseteq recorded.Album$,
- A ontologia M3 (figura **??**) importa a ontologia M1 e adiciona o axioma $NewSinger \equiv \neg\exists recorded.Album$,

Os dois axiomas isoladamente (nas ontologias M2 e M3) não causam nenhuma inconsistência nas ontologias respectivas no entanto quando combinados tornam o conceito `NewSinger` impossível de satisfazer.

Existem dois tipos de verificação de consistência que se podem realizar sobre uma KB. A verificação da TBox e a verificação da ABox relativamente à TBox. Em lógica de descrição uma ontologia é consistente se 'cada conceito na ontologia admite pelo menos um individuo' (**?**), no entanto o Pellet não realiza a verificação da consistência de conceitos mas sim de indivíduos, sendo verificado se a 'ABox é consistente relativamente à TBox' (**?**). Isto significa que uma ABox vazia é sempre consistente. Assim, a composição

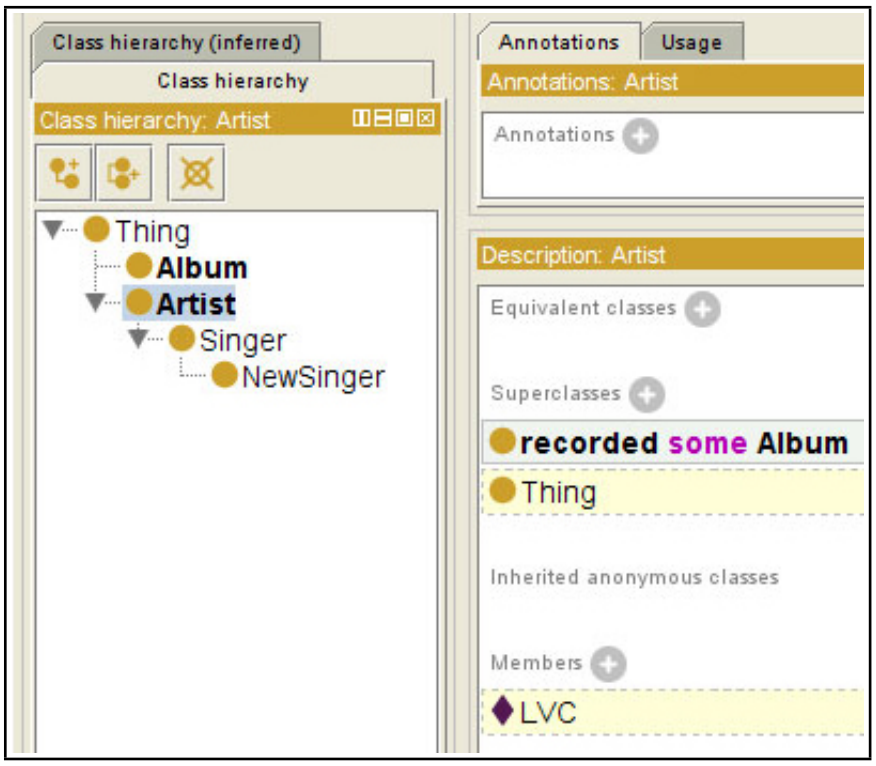

Figura 5.1: Ontologia M2

das duas ontologias é considerada consistente. Adicionando um individuo ao conceito `NewSinger` foi obtido o resultado de inconsistência para a composição das ontologias.

5.2.2 Composição modular de conceitos e relações

Ao configurar a aplicação com as ontologias MO e LVCMO, e fornecendo o critério de pesquisa 'english groups', obteve-se a DS apresentada na figura **??**. Esta DS contém duas FSs uma vez que o termo 'groups' no critério de pesquisa dá origem a dois TSs provenientes de cada uma das ontologias.

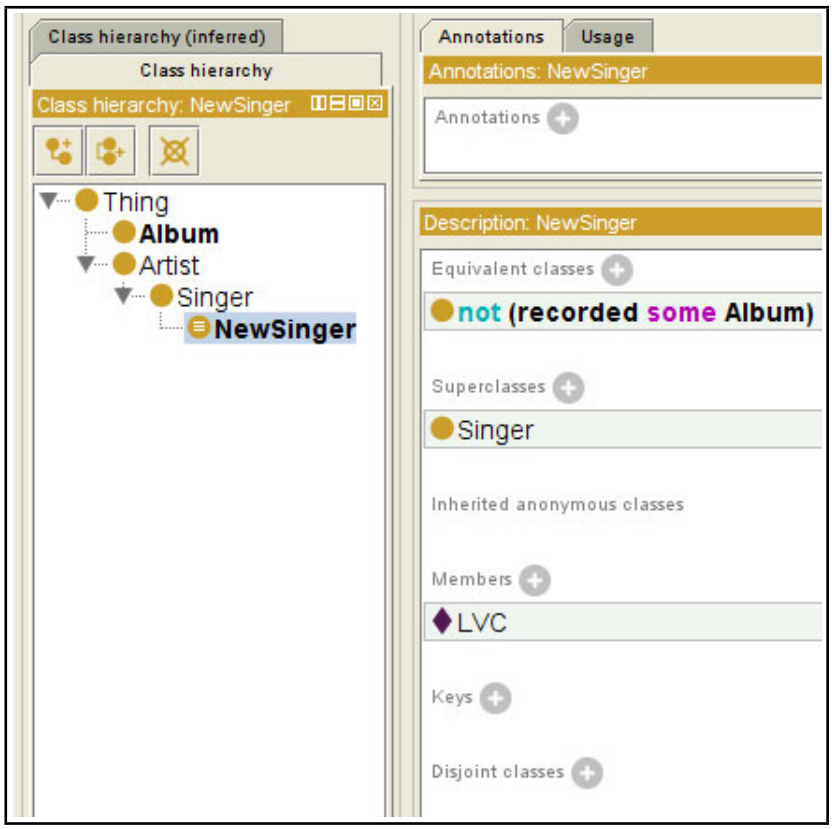

Figura 5.2: Ontologia M3

5.2.3 Endereçar domínios diferentes

A aplicação desenvolvida utiliza as ontologias de forma modular sendo possível a substituição da ontologia para endereçar um domínio diferente. As ontologias a utilizar são passadas como parâmetros à aplicação que as carrega e utiliza de forma totalmente agnóstica ao seu conteúdo. Os resultados da aplicação dependem da estrutura e da riqueza da ontologia utilizada mas a aplicação não necessita que nenhum requisito especial seja cumprido pela ontologia (para além de respeitar a linguagem OWL/RDF). Para validar este desacoplamento a aplicação foi testada com a ontologia MO, Movie Ontology (**?**) e com o critério de pesquisa 'romance movies' obtendo-se a DS apresentada na figura **??**.

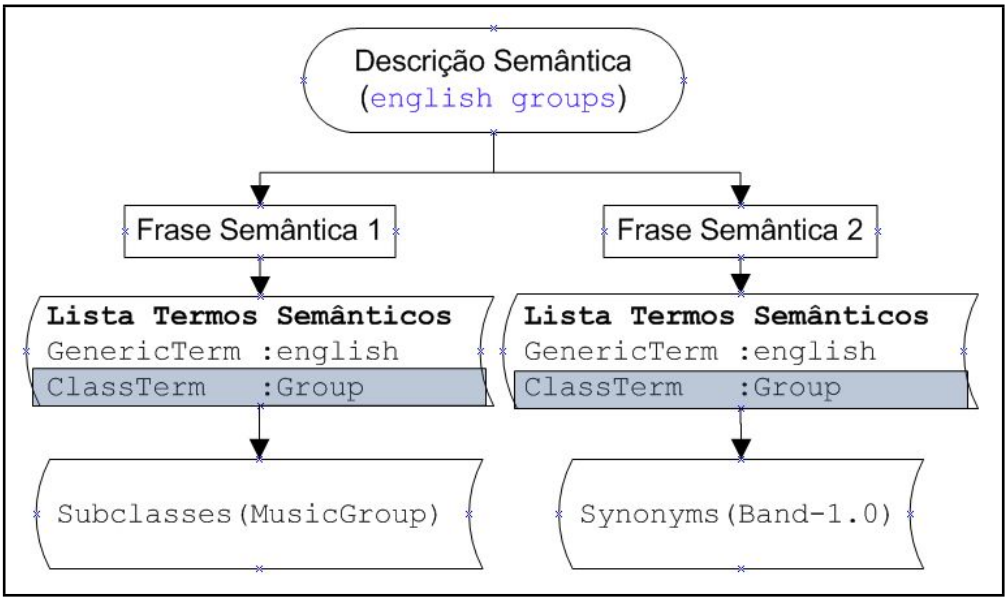

Figura 5.3: Descrição Semântica com múltiplos conceitos

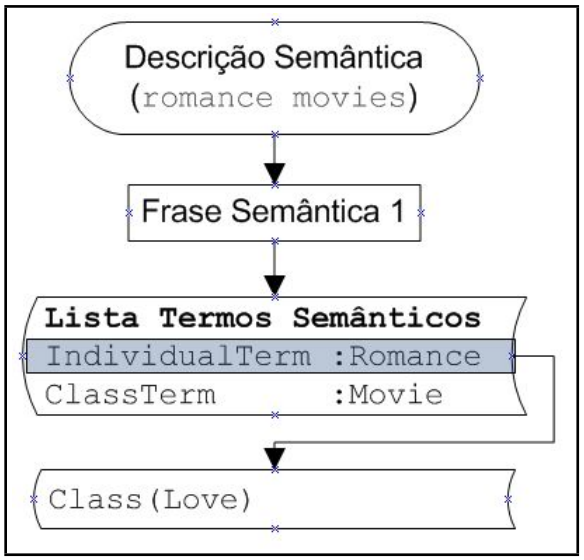

Figura 5.4: Descrição Semântica para Movie Ontology

5.3 Hipótese 2 - incorporar relações semânticas

'*Recuperar informação incorporando relações semânticas permite obter resultados que transcendam a mera semelhança sintáctica.*'

Para validar esta hipótese foram avaliados os resultados obtidos para três critérios de pesquisa. As secções seguintes descrevem a metodologia utilizada nos testes, as condições em que foram realizados e um resumo dos resultados obtidos. Por fim é explicado em detalhe a avaliação de cada critério de pesquisa incluindo a DS resultante e as expansões e resultados obtidos.

5.3.1 Metodologia dos testes

Para efectuar os testes foi reunido um conjunto de 18 documentos sobre diversos temas relacionados com música obtidos a partir de páginas disponíveis na *internet*. Os textos foram copiados integralmente do conteúdo das páginas não tendo sofrido qualquer alteração. Apenas o texto principal foi copiado não tendo sido copiados menus, separadores ou *banners*. Depois foram definidos três critérios de pesquisa com base em frases existentes nos documentos. As frases foram seleccionadas de acordo com os seguintes critérios:

- Frases com termos com diferentes significados em diferentes domínios,
- Frases com termos genéricos cuja substituição por termos mais específicos permita reduzir o âmbito da pesquisa,
- Frases com conceitos constituídos por múltiplos termos,
- Frases com termos relativos a indivíduos específicos passiveis de serem substituídos por um conceito base,

- Frases com termos passiveis de serem substituídos por termos equivalentes ou semelhantes.

A análise dos resultados focou-se em três aspectos:

- Avaliação do funcionamento da aplicação relativamente ao modelo proposto (avaliação da construção da DS e das expansões),
- Precisão da aplicação na obtenção dos documentos relevantes,
- Comparação entre os resultados obtidos com a aplicação e com o Lucene avaliando as métricas de precisão e cobertura (*recall*).

A tabela **??** no apêndice **??** descreve a configuração da aplicação usada na realização dos testes de validação do modelo.

5.3.2 Resumo dos resultados

A tabela **??** contém uma listagem dos documentos usados e a sua relevância para cada critério de pesquisa. A tabela **??** contém a lista dos principais documentos relevantes bem como a frase que os torna relevantes.

As tabelas **??** e **??** apresentam um resumo dos resultados obtidos (apenas os três primeiros resultados) pela aplicação e pelo Lucene respectivamente.

Documentos	Relevância por critério		
	Critério 1	Critério 2	Critério 3
sales-drop-01	x		
korn-01	x		
greatest-singer		x	
muse-01		x	
pearl-jam-01		x	
msg-biggest-01			x
sales-drop-02			
msg-03			
msg-mj-02			
yeah-yeah-yeah			
adidas-01			
incubus-01			
korn-02			
korn-03			
korn-adidas-01			
pearl-jam-02			
single-01			
yield-01			

Tabela 5.1: Relevância dos documentos por critério

Tabela 5.2: Documentos relevantes

Documento	Frase relevante
sales-drop-01	*Downloads up as album sales drop*
korn-01	*The band has blamed Internet piracy for the drop in sales, as an unmastered version of the album had leaked three months prior to its official release date*
greatest-singer	*Sarah is the best singer in the world*
muse-01	*Muse the best live band in the world*
pearl-jam-01	*In April 2006, Pearl Jam was awarded the prize for "Best Live Act*
msg-biggest-01	*In 2010, Madison Square Garden chose Michael Jackson's 1988 concert during the Bad World Tour as the greatest concert ever held at its venue"*

Tabela 5.3: Resumo dos resultados da aplicação

Modelo Proposto			
Critério Original	Classif.	Critério Pesquisado	Documento
record sales drop	1	album sale drop	sales-drop-01
	2	record sale drop	sales-drop-02
	3	record sale	adidas-01
best artist in the world	1	best singer world	greatest-singer
	2	best artist	msg-biggest-01
	3	artist world	yeah-yeah-yeah
greatest sHow at madisoN SQuare Garden	1	greatest concert held Venue Venue=MadisonSquareGarden	msg-biggest-01
	2	show madison_squar_garden	msg-03
	3	show madison_squar_garden	msg-mj-02

Tabela 5.4: Resumo dos resultados do Lucene

Lucene		
Critério Original	Classif.	Documento
record sales drop	1	sales-drop-02
	2	sales-drop-01
	3	korn-01
best artist in the world	1	msg-biggest-01
	2	yeah-yeah-yeah
	3	msg-biggest-01
greatest sHow at madisoN SQuare Garden	1	msg-03
	2	msg-biggest-01
	3	msg-mj-02

Para avaliar as métricas de precisão e cobertura foi necessário calcular primeiro os seguintes valores para cada critério (com a aplicação e com o Lucene):

- **tp** — *true positive*, nº de documentos relevantes classificados como relevantes (obtidos no resultado da pesquisa)
- **fp** — *false positive*, nº de documentos não relevantes classificados como relevantes (obtidos no resultado da pesquisa)

- **fn** — *false negative*, nº de documentos relevantes classificados como não relevantes (não obtidos no resultado da pesquisa)
- **tn** — *true negative*, nº de documentos não relevantes classificados como não relevantes (não obtidos no resultado da pesquisa)

As tabelas **??** e **??** apresentam os valores apurados para cada um dos critérios, para a aplicação e o Lucene respectivamente.

Tabela 5.5: Valores das métricas da aplicação

	tp	fp	fn	tn
Critério 1	2	4	0	12
Critério 2	3	2	0	13
Critério 3	1	4	0	13

Tabela 5.6: Valores das métricas do Lucene

	tp	fp	fn	tn
Critério 1	2	12	0	4
Critério 2	3	12	0	3
Critério 3	1	15	0	2

A precisão e a cobertura foram calculadas usando as formulas:

$$precisão = \frac{tp}{tp + fp}$$

$$cobertura = \frac{tp}{tp + fn}$$

Os gráficos das figuras **??**, **??** e **??** apresentam os valores da precisão na aplicação e no Lucene para as pesquisas dos três critérios. Os valores da cobertura não são apresentados uma vez foi obtido o valor 1 em todas as situações.

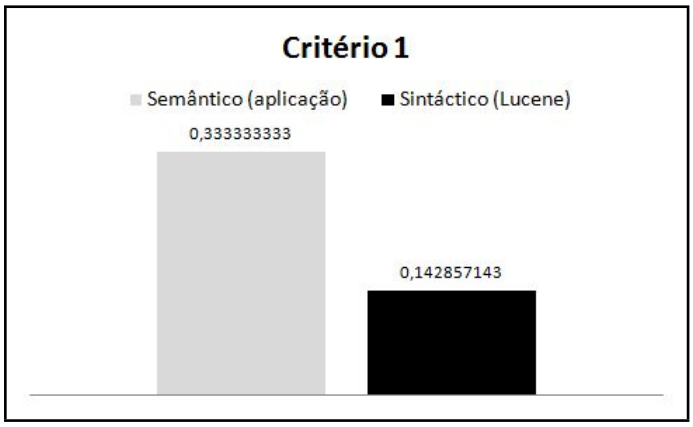

Figura 5.5: Comparação da precisão no critério 1

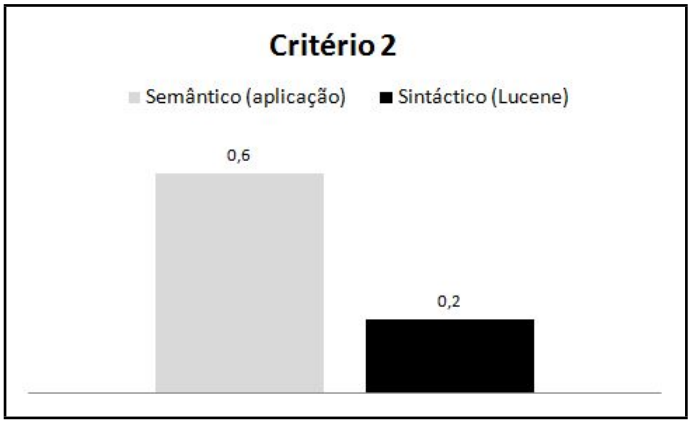

Figura 5.6: Comparação da precisão no critério 2

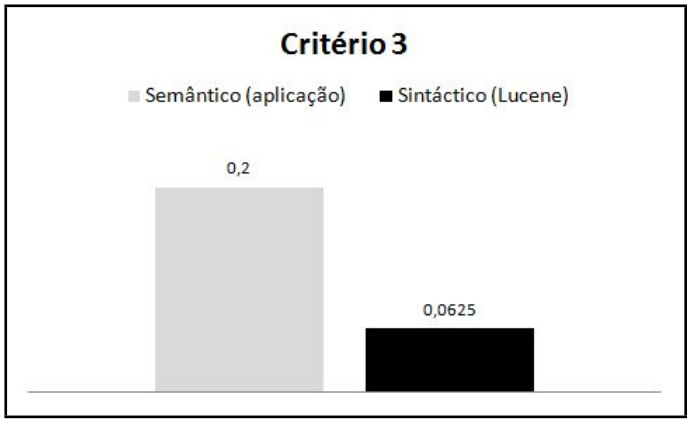

Figura 5.7: Comparação da precisão no critério 3

O valor da cobertura é sempre 1 uma vez que os documentos relevantes foram sempre recuperados, no entanto a precisão é sempre melhor na aplicação quando comparado com o Lucene. O ganho de precisão deve-se ao facto de se usar pesquisa de frases o que restringe o número de resultados. No entanto, e não obstante a restrição no número de resultados, a aplicação recuperou todos os resultados relevantes.

Foi também efectuada a pesquisa dos critérios no Lucene configurado para realizar pesquisa de frases (com PHRASE-SLOP=20) obtendo-se apenas o documento 'sales-drop-02' para o critério 1 e nenhum documento para os critérios 2 e 3.

5.3.3 Critério 1

'*record sales drop*'

Este critério de pesquisa dá origem à DS apresentada na figura **??**. A DS apresenta duas FSs uma vez que o termo 'record' existe como classe e como propriedade na ontologia. Na primeira FS a classe 'Record' tem associada a informação inferida da ontologia (classes equivalentes e descendentes). De notar que a lista de sinónimos apresenta não só as classes equivalentes mas também as classes semelhantes uma vez que para a DS não é relevante se a classe é inferida por via de um axioma de equivalência definido na ontologia LVCMO ou se é inferida por via de uma axioma de equivalência probabilística definido na ontologia PLVCMO. O valor associado indica o grau de semelhança (neste caso 1.0 indica equivalência). A partir desta DS são geradas as expansões apresentadas na listagem **??**. O termo 'record' é substituído pela classe 'Record' e por cada uma das subclasses e classes equivalentes. De notar que de acordo com o descrito na secção **??** (página **??**) não existem sub-expansões repetidas.

Na listagem **??** pode observar-se que as expansões apresentam todas um coeficiente semântico unitário. Este resultado deve-se ao facto dos sinónimos terem coeficiente unitário e o parâmetro SUB-CLASS-FACTOR ter também o valor 1. O valor unitário

neste parâmetro indica que se considera não haver degradação semântica quando uma classe é substituída por uma subclasse. Este parâmetro assume este valor pelo facto de se considerar que este tipo de substituição especializa o critério de pesquisa reduzindo o número potencial de resultados mas não degrada a semântica do critério. Ou seja se $B \sqsubset A$ então qualquer resultado com B será um resultado com A. As sub-expansões apresentam uma degradação do coeficiente semântico (de acordo com a formula apresentada na secção **??** na página **??**) uma vez que a eliminação de termos causa um acentuar da divergência com o critério de pesquisa original.

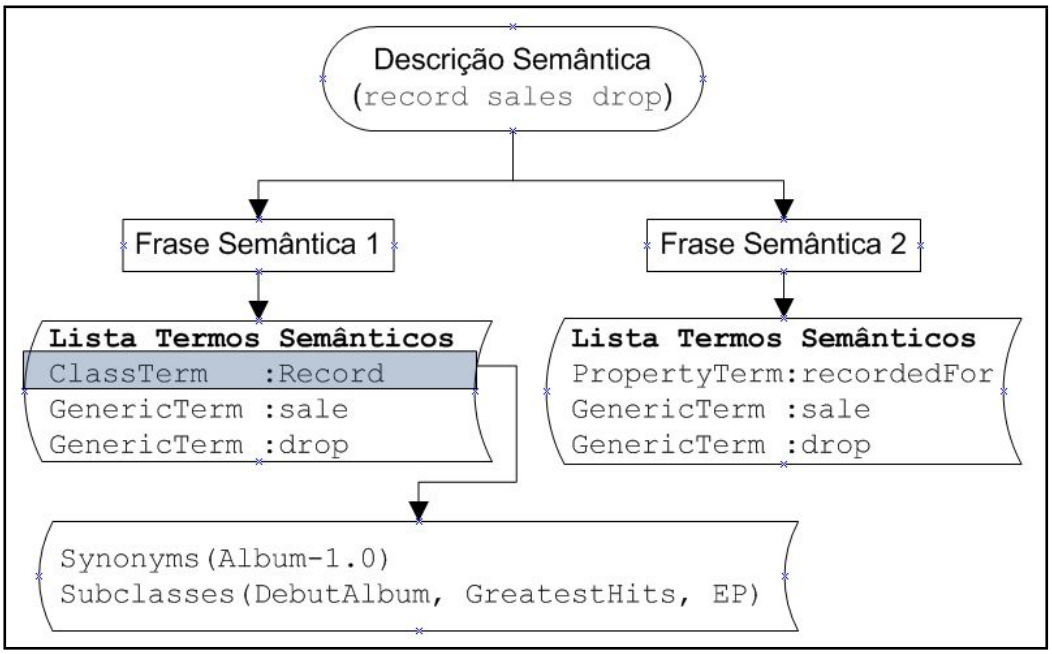

Figura 5.8: Descrição Semântica para critério 1

Os resultados da pesquisa são apresentados na tabela **??** podendo observar-se três aspectos interessantes.

- O Lucene apresenta uma lista de resultados muito maior, o que está de acordo com o esperado uma vez que a aplicação procura frases e o Lucene procura termos. Ou seja a aplicação é mais restritiva nas pesquisas efectuadas.

Listing 5.1: Expansões para critério 1

```
Expansions for Phrase 1

ep sale drop - 1.0
        ep sale - 0.19753088
        sale drop - 0.19753088
        ep drop - 0.19753088
debut_album sale drop - 1.0
        debut_album drop - 0.19753088
        debut_album sale - 0.19753088
record sale drop - 1.0
        record drop - 0.19753088
        record sale - 0.19753088
greatest_hit sale drop - 1.0
        greatest_hit sale - 0.19753088
        greatest_hit drop - 0.19753088
album sale drop - 1.0
        album sale - 0.19753088
        album drop - 0.19753088
```

- O Lucene indica como resultado mais relevante o documento 'sales-drop-02' cujo conteúdo foi retirado de uma página de um canal de noticias (foxnews) reportando sobre a crise económica e a queda nas vendas de imóveis. Na aplicação o mesmo documento é classificado na segunda posição sendo considerado como documento mais relevante o documento 'sales-drop-01' cujo conteúdo foi retirado também de uma página de um canal de noticias (bbcnews) mas que fala da queda das vendas de *albums* e do aumento dos downloads.

- A substituição de 'record' por 'album' permite contextualizar a pesquisa,

Este cenário configura um exemplo de desambiguação semântica com base na ontologia. Para o Lucene o termo 'record' não tem significado semântico e por isso, aparecendo no documento 'sales-drop-02', contribuí para que este seja classificado como o mais relevante. A aplicação no entanto possuí um contexto semântico (música) e é orientada à frase e por isso o documento 'sales-drop-01' que contém a frase '*Downloads up as album sales drop*' é melhor classificado do que o documento 'sales-drop-02' com a frase '*sales drop to lowest level on record*'.

Tabela 5.7: Resultados para critério 1

Documento	Critério Pesquisado	Pontuação
Resultados Aplicação		
sales-drop-01	album sale drop	0.415
sales-drop-02	record sale drop	0.165
adidas-01	record sale	0.036
korn-01	album sale drop	0.024
muse-01	album sale drop	0.010
yield-01	album sale drop	0.005
Resultados Lucene		
sales-drop-02	record sales drop	0.399
sales-drop-01	record sales drop	0.353
korn-01	record sales drop	0.101
adidas-01	record sales drop	0.095
yield-01	record sales drop	0.049
pearl-jam-01	record sales drop	0.036
msg-biggest-01	record sales drop	0.033
incubus-01	record sales drop	0.032
muse-01	record sales drop	0.032
pearl-jam-02	record sales drop	0.020
korn-02	record sales drop	0.017
korn-adidas-01	record sales drop	0.014
yeah-yeah-yeah	record sales drop	0.011
greatest-singer	record sales drop	0.006

5.3.4 Critério 2

'*best artist in the world*'

Este critério de pesquisa produz a DS apresentada na figura **??** com apenas uma FS uma vez que cada termo origina apenas um TS. As expansões geradas são apresentadas na listagem **??** e a tabela **??** apresenta a frequência dos termos nos três documentos considerados mais relevantes (tanto pelo Lucene como pela aplicação).

Os resultados da pesquisa são apresentados na tabela **??** e volta a ser evidenciada a

vantagem de conjugar a pesquisa de frases com a incorporação de informação semântica no critério de pesquisa. o Lucene relega para terceiro lugar o documento 'greatest-singer' por não conter o termo 'artist' e destaca os outros dois documentos por conterem um melhor equilíbrio dos termos pesquisados mas a aplicação destaca este documento como o mais relevante uma vez que a substituição semântica lhe permite efectuar pesquisas alternativas. A frase '*best artist in the world*' também é pesquisada pela aplicação o que levaria a supor que o documento 'yeah-yeah-yeah' poderia competir pelo primeiro lugar. É aqui que a pesquisa de frases se mostra relevante já que o documento 'greatest-singer' inclui a frase '*Sarah is the best singer in the world*' e no documento 'yeah-yeah-yeah' os termos aparecem dispersos.

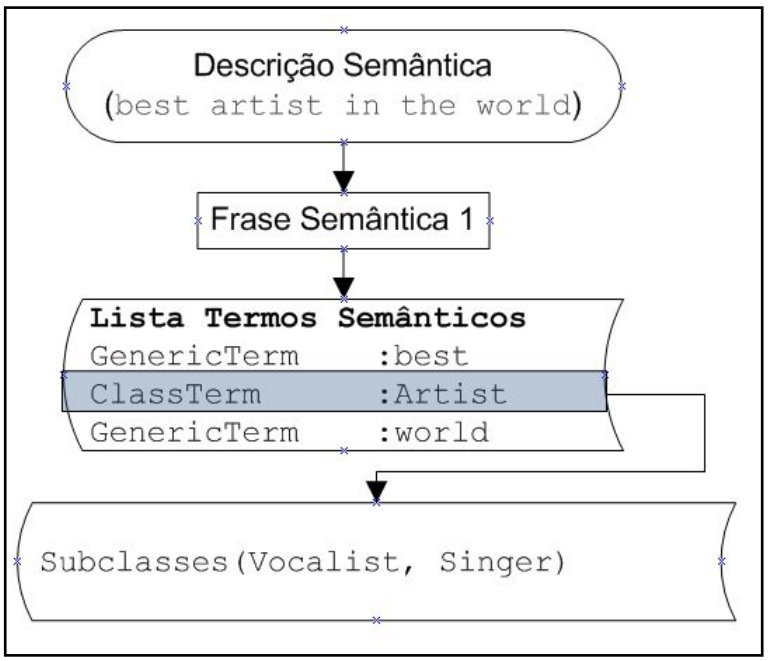

Figura 5.9: Descrição Semântica para critério 2

Listing 5.2: Expansões para critério 2

```
Expansions for Phrase 1

best vocalist world - 1.0
        best vocalist - 0.19753088
        vocalist world - 0.19753088
        best world - 0.19753088
best singer world - 1.0
        best singer - 0.19753088
        singer world - 0.19753088
best artist world - 1.0
        artist world - 0.19753088
        best artist - 0.19753088
```

Tabela 5.8: Resultados para critério 2

Documento	Critério Pesquisado	Pontuação
Resultados Aplicação		
greatest-singer	best singer world	0.147
msg-biggest-01	best artist	0.007
yeah-yeah-yeah	artist world	0.006
muse-01	best world	0.004
pearl-jam-01	best artist	0.003
Resultados Lucene		
msg-biggest-01	best artist in the world	0.132
yeah-yeah-yeah	best artist in the world	0.131
greatest-singer	best artist in the world	0.091
pearl-jam-01	best artist in the world	0.083
muse-01	best artist in the world	0.081
korn-01	album sales drop	0.070
msg-mj-02	best artist in the world	0.065
incubus-01	best artist in the world	0.056
sales-drop-01	best artist in the world	0.055
korn-02	best artist in the world	0.045
adidas-01	best artist in the world	0.043
korn-adidas-01	best artist in the world	0.036
yield-01	best artist in the world	0.031
pearl-jam-02	best artist in the world	0.017
korn-03	best artist in the world	0.010

Tabela 5.9: Frequência de termos

	Termos			
	best	artist	singer	world
greatest-singer	13	0	12	2
msg-biggest-01	4	2	0	6
yeah-yeah-yeah	10	4	0	4

5.3.5 Critério 3

'*greatest sHow at madisoN SQuare Garden*'

Neste critério de pesquisa pode observar-se a utilização da versão *stemmed* dos termos para realizar inferência sobre a ontologia uma vez que o termo 'sHow' é correctamente interpretado como a classe 'Show'. Pode também confirma-se a correcta interpretação de conceitos multi termo. Os termos 'madisoN', 'SQuare' e 'Garden' são correctamente interpretados como o individuo 'MadisonSquareGarden' e não como três TSs genéricos. A DS obtida para este critério contém uma FS e é apresentada na figura **??**. As expansões geradas são apresentadas na listagem **??**.

Neste critério de pesquisa observa-se ainda o efeito da componente probabilística. Na ontologia PLVCMO está definido um axioma de equivalência probabilística entre as classes 'Show' e 'Gig' o que resulta na classe 'Gig' aparecer na DS como sinónimo de 'Show' (com probabilidade 0.7). Neste cenário é validado também o conceito de substituição de um individuo pela sua classe. Ao substituir o individuo 'MadisonSquareGarden' pela sua classe 'Venue' a aplicação consegue construir um critério de pesquisa alternativo, sintacticamente bastante diferente do critério de pesquisa original, mas com uma grande relação semântica. A aplicação atribuí uma pontuação à substituição do individuo pela sua classe permitindo uma avaliação da relevância da substituição no contexto do documento.

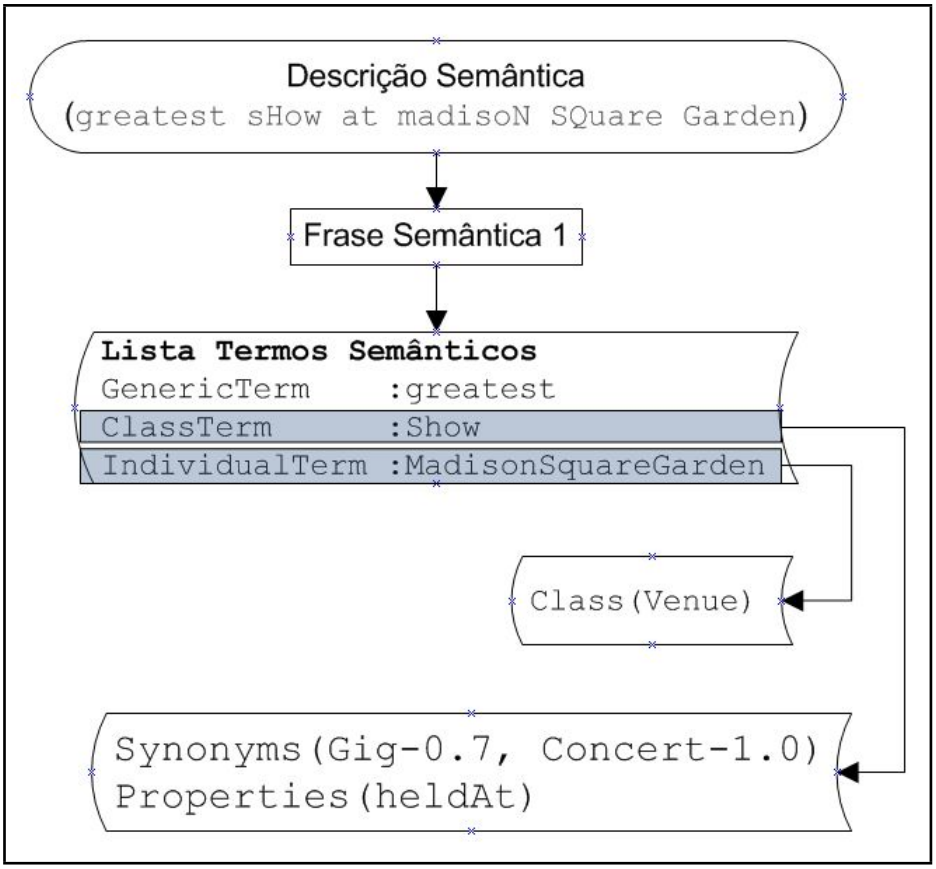

Figura 5.10: Descrição Semântica para critério 3

Listing 5.3: Expansões para critério 3

```
Expansions for Phrase 1

greatest Gig madison_squar_garden - 0.7
        greatest Gig - 0.13827161
        greatest madison_squar_garden - 0.13827161
        Gig madison_squar_garden - 0.13827161
greatest Gig Venue - 0.7
        greatest Venue - 0.13827161
        Gig Venue - 0.13827161
greatest Gig held madison_squar_garden - 0.7
        greatest held madison_squar_garden - 0.22148438
        held madison_squar_garden - 0.04375
        Gig held madison_squar_garden - 0.22148438
        greatest Gig held - 0.22148438
        greatest held - 0.04375
        Gig held - 0.04375
greatest Gig held Venue - 0.7
        greatest held Venue - 0.22148438
        held Venue - 0.04375
        Gig held Venue - 0.22148438
greatest show madison_squar_garden - 1.0
        show madison_squar_garden - 0.19753088
        greatest show - 0.19753088
greatest show Venue - 1.0
        show Venue - 0.19753088
greatest show held madison_squar_garden - 1.0
        greatest show held - 0.31640625
        show held - 0.0625
        show held madison_squar_garden - 0.31640625
greatest show held Venue - 1.0
        show held Venue - 0.31640625
greatest Concert madison_squar_garden - 1.0
        greatest Concert - 0.19753088
        Concert madison_squar_garden - 0.19753088
greatest Concert Venue - 1.0
        Concert Venue - 0.19753088
greatest Concert held madison_squar_garden - 1.0
        Concert held madison_squar_garden - 0.31640625
        greatest Concert held - 0.31640625
        Concert held - 0.0625
greatest Concert held Venue - 1.0
        Concert held Venue - 0.31640625
```

Tabela 5.10: Resultados para critério 3

Documento	Critério Pesquisado	Pontuação
Resultados Aplicação		
msg-biggest-01	greatest concert held Venue Venue=MadisonSquareGarden 0.7564937	0.105
msg-03	show madison_squar_garden	0.028
msg-mj-02	show madison_squar_garden	0.029
pearl-jam-01	show madison_squar_garden	0.010
muse-01	concert held	0.003
Resultados Lucene		
msg-03	greatest sHow at ...Garden	0.498
msg-biggest-01	greatest sHow at ...Garden	0.477
msg-mj-02	greatest sHow at ...Garden	0.117
pearl-jam-01	greatest sHow at ...Garden	0.074
muse-01	greatest sHow at ...Garden	0.067
greatest-singer	greatest sHow at ...Garden	0.047
korn-01	album sales drop	0.039
korn-02	greatest sHow at ...Garden	0.016
incubus-01	greatest sHow at ...Garden	0.014
korn-adidas-01	greatest sHow at ...Garden	0.011
sales-drop-01	greatest sHow at ...Garden	0.005
korn-03	greatest sHow at ...Garden	0.005
yeah-yeah-yeah	greatest sHow at ...Garden	0.003
pearl-jam-02	greatest sHow at ...Garden	0.003
sales-drop-02	greatest sHow at ...Garden	0.003
yield-01	greatest sHow at ...Garden	0.002

5.4 Hipótese 3 - baixa complexidade do modelo

'*Uma pesquisa baseada em frases que incorpore inferência sobre ontologias permite construir um modelo com uma baixa complexidade (e.g., em relação ao NLP)*'

Para validar esta hipótese é necessário analisar a complexidade do modelo desenvolvido

relativamente a dois aspectos:

1. A facilidade de configuração e manutenção do modelo.
2. A escalabilidade do modelo, ou seja a maneira como evolui o desempenho em função do aumento da complexidade.

Relativamente ao primeiro aspecto o modelo utiliza ontologias de forma modular e sem impor quaisquer tipo de regras na sua modelação, o que torna a sua utilização extremamente fácil. Para configurar a aplicação ou alterar o domínio semântico basta fornecer uma ontologia OWL-DL.

Relativamente ao segundo aspecto é necessário em primeiro lugar analisar os factores que maior impacto tem no desempenho da aplicação.

1. Número e tamanho dos documentos indexados,
2. Valor do parâmetro PHRASE-SLOP,
3. Tipo de estrutura de indexação usada (memória ou ficheiro),
4. Complexidade da(s) ontologia(s) usadas,
5. Número e tipo de termos utilizados no critério de pesquisa original,
6. Número de Frases Semânticas,
7. Número de expansões,
8. Número de sub-expansões,
9. Número de axiomas probabilísticos envolvidos na Descrição Semântica

Os três primeiros pontos apresentados estão relacionados directamente com o Lucene. O tamanho e o número de documentos utilizados deverá ter um impacto significativo na

fase de indexação, no entanto este processamento é realizado de forma independente da aplicação não afectando o seu desempenho. Já na fase de pesquisa é expectável que o número e tamanho dos índices criados tenham um impacto significativo no desempenho da aplicação. Por outro lado é também expectável que o valor do parâmetro PHRASE-SLOP afecte o desempenho da pesquisa uma vez que quantos mais passos forem permitidos na pesquisa de frases maior o processamento envolvido (c.f. **??** na página **??**). Por fim o tipo de estrutura de indexação deve ter também um impacto no desempenho da pesquisa sendo expectável um ganho de desempenho quando os índices são mantidos em memória relativamente à opção de trabalhar com os índices a partir de ficheiro.

Os pontos 4 a 8 estão ligadas ao processamento semântico. A complexidade das ontologias usadas deverá ter impacto no seu carregamento, composição e na verificação de consistência e de inferência. Por outro lado o aumento na complexidade das ontologias usadas deverá potenciar a criação de DSs mais elaboradas com previsível aumento nos tempos de criação e no número de expansões criadas. Uma DS mais elaborada com maior número de expansões obrigará por sua vez à realização de mais pesquisas com impacto no tempo total de pesquisa.

Por fim o último ponto prende-se com a utilização do Pronto no cálculo de classes semelhantes recorrendo a lógica de descrição probabilística. Também aqui um aumento do número de inferências deverá causar um aumento deste tempo de processamento.

Para analisar o desempenho começou-se por registar os tempos associados a uma pesquisa simples[1]. A tabela **??** apresenta a configuração usada e a listagem **??** apresenta a descrição semântica obtida. A listagem **??** apresenta os tempos do processamento em milisegundos.

Os resultados indicam tempos baixos em todos os factores com um tempo total de pesquisa de 169 ms para 17 pesquisas.

O cenário 2 descrito na tabela **??** origina a DS apresentada na listagem **??** e o de-

[1]Processador: quad-core 2.4GHz; RAM: 4Gb, 64-bit Vista

Tabela 5.11: Descrição do cenário 1

Parâmetro	Valor
Nº Docs.	19
Tempo de indexação	424 ms
Ontologias	LVCMO
Critério	record sales drop
Inferências Prob.	0
Nº expansões	17
Índices	memoria

Listing 5.4: Descrição Semântica para cenário 1

```
Phrase Count: 2

Semantic Terms for phrase 1:
ClassTerm          :Record;
          Synonyms(Record-1.0,Album-1.0);
          Subclasses(DebutAlbum,GreatestHits,EP)
GenericTerm        :sale
GenericTerm        :drop

Semantic Terms for phrase 2:
PropertyTerm       :recordedAt
GenericTerm        :sale
GenericTerm        :drop
```

Tabela 5.12: Descrição do cenário 2

Parâmetro	Valor
Nº Docs.	4864
Tempo de indexação	19183 ms
Ontologias	LVCMO, MO e PLVCMO
Critério	frontman of rock bands
Inferências Prob.	3
Nº expansões	1380
Índices	memoria

Listing 5.5: Desempenho para cenário 1

```
Desempenho (ms)

Carregamento de ontologias                  : 354
Composição de ontologias                    : 6
Verificação de consistência                 : 65
Clasificação de ontologia                   : 29
Carregamento de ontologia prob.             : 0
Total carr. gestor de ontologias            : 2163
Total carr. aplicação                       : 2218
-------------------------------------------------

Inferência probabilísitica                  : 0
Total da descrição semântica                : 27
-------------------------------------------------

Expansão da descrição semântica             : 4
-------------------------------------------------

Número de pesquisas realizadas              : 17
Tempo total de pesquisa                     : 169
-------------------------------------------------

Classificação                               : 1
```

Listing 5.6: Descrição Semântica para cenário 2

```
Phrase Count: 1

Semantic Terms for phrase 1:
ClassTerm          :Frontman;
          Synonyms(Lead Vocalist -0.9,Vocalist -0.72,
                   Lead Singer -0.9);
          Properties(theme,fundedBy,logo)
ClassTerm          :Rock;
          Subclasses(Alternative,Metal,Punk,Grunge);
          Properties(theme,fundedBy,logo)
ClassTerm          :Band;
          Synonyms(Group -1.0)
```

Listing 5.7: Desempenho para cenário 2

```
Desempenho (ms)


Carregamento de ontologias                : 5008
Composição de ontologias                  : 31
Verificação de consistência               : 95
Clasificação de ontologia                 : 109
Carregamento de ontologia prob.           : 7838
Total carr. gestor de ontologias          : 14859
Total carr. aplicação                     : 15077
-----------------------------------------------------

Inferência probabilísitica                : 2189
Total da descrição semântica              : 2313
-----------------------------------------------------

Expansão da descrição semântica           : 156
-----------------------------------------------------

Número de pesquisas realizadas            : 1380
Tempo total de pesquisa                   : 999
-----------------------------------------------------

Classificação                             : 0
```

sempenho descrito na listagem **??**. Observa-se um aumento do tempo de carregamento devido em grande parte ao carregamento das ontologias. Relativamente ao desempenho da aplicação após o seu carregamento, nota-se um tempo significativo na inferência probabilística mas um tempo muito baixo na pesquisa das expansões. O Lucene realizou 1380 pesquisas em 4864 documentos em menos de um segundo. É expectável que este valor aumente com o número de documentos sendo neste caso necessário controlar o número de expansões que neste cenário foi deixado propositadamente elevado. De qualquer forma é um resultado muito bom por parte do Lucene. Alterando-se a estrutura de indexação de memória para ficheiro este tempo não se alterou significativamente.

Com estes resultados é possível analisar as três fases de processamento na aplicação. O carregamento, a construção semântica e a pesquisa. O carregamento ocorre apenas uma vez e não é relevante durante a fase de pesquisa da aplicação. A pesquisa por parte

Listing 5.8: Descrição Semântica para cenário 3

```
Phrase Count: 1

Semantic Terms for phrase 1:
ClassTerm          :C1;
          Synonyms(C2-0.7,C3-0.49,C4-0.34,
                   D1-1.0,D2-0.7,D3-0.34,D4-0.49);
          Properties(theme,fundedBy,logo)
ClassTerm          :C2;
          Synonyms(C1-0.7,C3-0.7,C4-0.49,
                   D1-0.7,D2-0.49,D3-0.34,D4-0.24);
          Properties(theme,fundedBy,logo)
ClassTerm          :C3;
          Synonyms(C1-0.49,C2-0.7,C4-0.7,
          D1-0.49,D2-0.34,D3-0.24,D4-0.17,)
```

do Lucene é extremamente rápida. O número de expansões tem tendência para explodir sendo necessário controlar o número permitido de expansões. Ainda assim o Lucene consegue realizar centenas de pesquisas em espaços de tempo muito curtos. Quanto à construção semântica pode observar-se que o verdadeiro congestionamento se encontra na inferência probabilística. Para apurar melhor o desempenho deste componente recorreu-se a um terceiro cenário com uma inferência probabilística de complexidade acrescida. Para o efeito utilizou-se as subclasses da classe 'ProntoTest' (c.f. figura **??** no apêndice **??**) e o critério 'C1 C2 C3'. A listagem **??** apresenta a DS obtida e a listagem **??** apresenta os resultados.

Observa-se um tempo mais elevado mas um crescimento moderado tendo em conta que o número de inferência probabilísticas passa de 3 para 20 e o tempo aumenta para pouco menos do dobro. Este cenário possibilita também avaliar o Lucene em condições mais extremas uma vez que são produzidas 10973 expansões. O Lucene volta a surpreender uma vez que apesar do aumento do tempo de pesquisa o tempo médio por pesquisa diminuiu significativamente. Neste cenário, se o Lucene trabalhar directamente com os ficheiros de índices (sem os carregar em memória) obtém-se um tempo de pesquisa de 4586 ms.

Listing 5.9: Desempenho para cenário 3

```
Desempenho (ms)

Carregamento de ontologias                  : 5476
Composição de ontologias                    : 31
Verificação de consistência                 : 95
Clasificação de ontologia                   : 109
Carregamento de ontologia prob.             : 7196
Total carr. gestor de ontologias            : 14654
Total carr. aplicação                       : 14872
-------------------------------------------------

Inferência probabilísitica                  : 3873
Total da descrição semântica                : 3998
-------------------------------------------------

Expansão da descrição semântica             : 483
-------------------------------------------------

Número de pesquisas realizadas              : 10973
Tempo total de pesquisa                     : 3635
-------------------------------------------------

Classificação                               : 16
```

Capítulo 6

Conclusões

Nesta dissertação é proposto um modelo que utiliza informação sobre um domínio para expandir um critério de pesquisa num conjunto de critérios alternativos semanticamente relacionados. A informação sobre o domínio é fornecida por uma ontologia sobre a qual se realiza inferência para, a partir do critério de pesquisa, construir uma Descrição Semântica que contextualize o critério no domínio. A Descrição Semântica é uma estrutura que expressa relações entre conceitos, propriedades e indivíduos permitindo a construção de critérios alternativos de pesquisa. A pesquisa dos critérios alternativos é orientada à frase e não ao termo para aumentar a precisão dos resultados.

Esta abordagem permite obter resultados de pesquisa sintacticamente diferentes mas semanticamente semelhantes. A utilização de um domínio especifico permite desambiguar conceitos potenciando o aumento da precisão dos resultados. Por outro lado a utilização de ontologias numa lógica de *plugin* torna o modelo muito flexível. Neste trabalho é ainda usada lógica de descrição probabilística como forma de modelar a noção de semelhança entre conceitos (equivalentes com uma probabilidade). Foram obtidos bons resultados nos testes conduzidos quer ao nível do aumento de precisão quer ao nível do desempenho da aplicação desenvolvida para validar o modelo proposto. Ao nível do desempenho foi possível identificar a inferência probabilística como o componente mais dispendioso mas ainda assim com tempos bastante promissores. Ao nível da precisão os

resultados permitiram validar as hipóteses formuladas tendo sido demonstrada a capacidade da aplicação para desambiguar termos e produzir critérios de pesquisa semanticamente relacionados e sintacticamente diferentes.

6.1 Importância da ontologia

O modelo proposto não impõem nenhum conjunto de regras a uma ontologia para que esta possa ser utilizada (tendo apenas de respeitar o formato OWL-DL). No entanto, a modelação da ontologia desempenha um papel vital nos resultados conseguidos pelo modelo. A ontologia deve ser modelada com noções claras sobre o que são classes o que são indivíduos e o que são propriedades. Existem regras importantes a ter em consideração ao desenhar uma ontologia (**?**) no entanto muitas das ontologias disponíveis na *internet* apresentam ainda uma modelação descuidada e pouco coerente (Veja-se o caso da MO - Music Ontology onde o conceito 'Album' é modelado como um individuo apesar de representar uma classe abstracta de indivíduos).

6.2 Cenários de utilização

Uma empresa que opere num ramo de actividade com uma gíria própria, com grande abundância de termos técnicos ou específicos à sua actividade, pode beneficiar deste modelo pois este possibilita a um utilizador efectuar pesquisas em linguagem comum. A modelação do conhecimento específico ao contexto da empresa numa ontologia e a sua utilização pelo Interpretador Semântico permitirá traduzir critérios de pesquisa no contexto da empresa aumentando a precisão da pesquisa. Na realidade qualquer domínio com um vocabulário próprio que necessite de ser pesquisado por pessoas com pouco conhecimento desse vocabulário é um bom candidato à utilização com vantagem deste modelo.

6.3 Trabalho futuro

Uma vez que o modelo suporta a composição de ontologias é possível incluir uma ontologia genérica sobre um determinado idioma para melhorar a contribuição de adjectivos (e.g. *best*, *greatest*, *biggest*).

Uma variação interessante do modelo pode passar por substituir o Lucene pelo Google! Ou seja, em vez de utilizar o Lucene para pesquisar um conjunto de documentos segundo uma lista de critérios alternativos, utilizar o Google para pesquisar a *internet* segundo o mesmo conjunto de critérios alternativos. Neste caso, seria no entanto necessário obter um coeficiente de indexação do Google.

Apêndice A

Ontologias

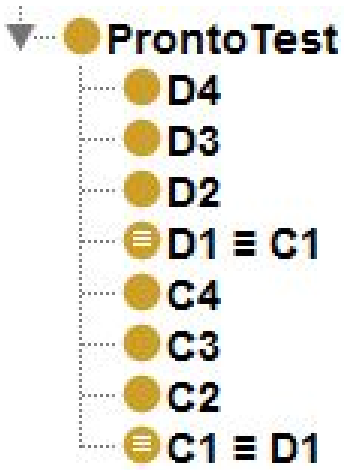

Figura A.1: Ontologia LVCMO (ProntoTest)

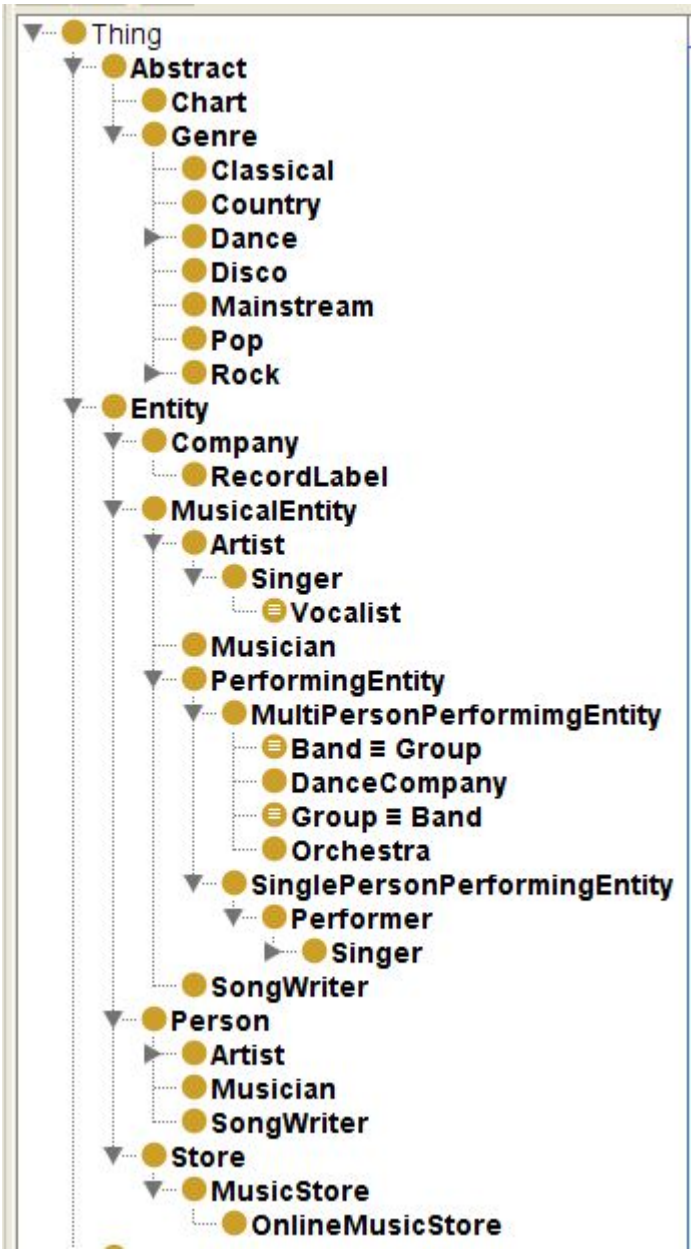

Figura A.2: Ontologia LVCMO (Classes 1)

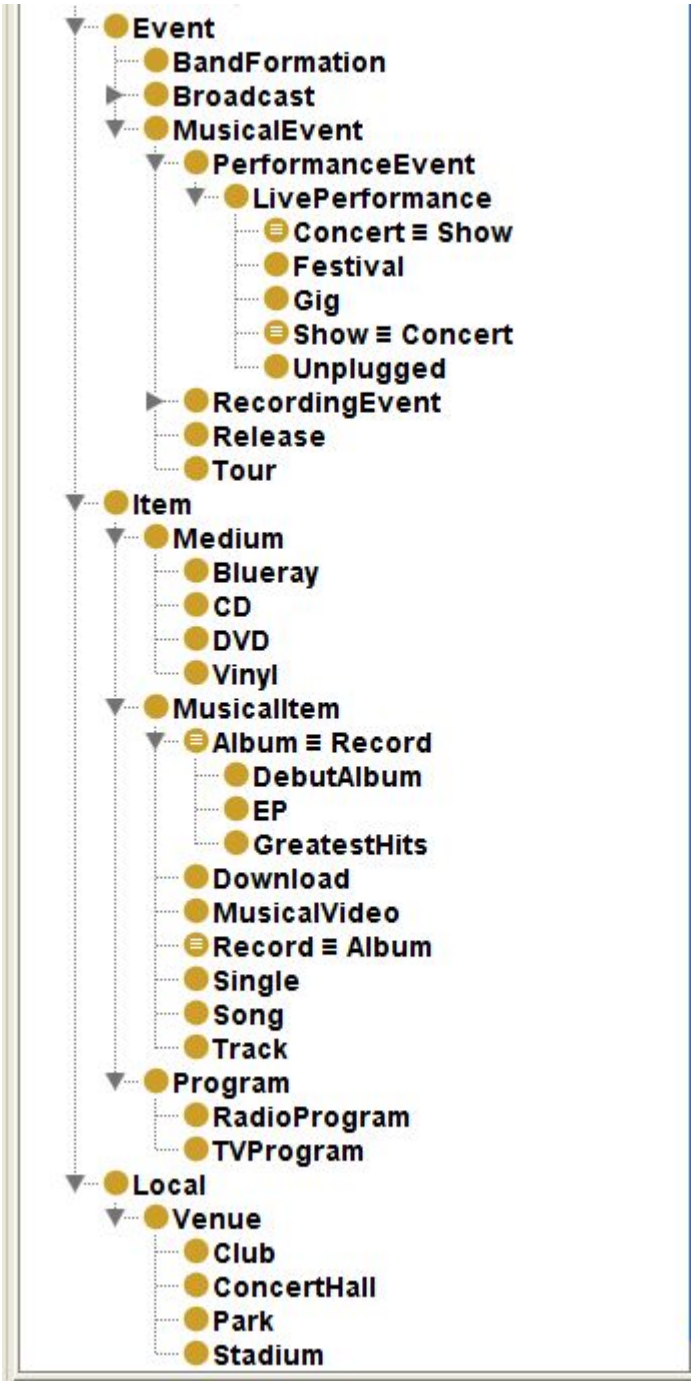

Figura A.3: Ontologia LVCMO (Classes 2)

Figura A.4: Ontologia LVCMO (Propriedades)

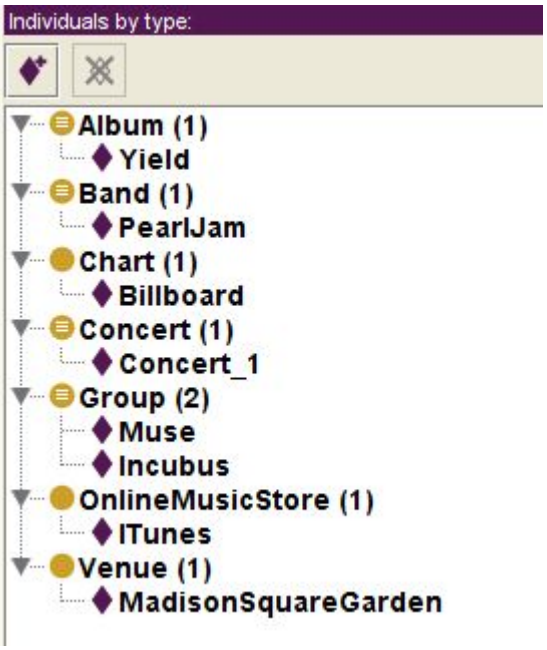

Figura A.5: Ontologia LVCMO (Individuos)

```
- <owl11:Axiom>
    <rdf:object rdf:resource="http://lvc.pt/music.owl#C1" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#C2" />
    <pronto:certainty>0.7;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:object rdf:resource="http://lvc.pt/music.owl#C2" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#C3" />
    <pronto:certainty>0.7;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:object rdf:resource="http://lvc.pt/music.owl#C3" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#C4" />
    <pronto:certainty>0.7;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:object rdf:resource="http://lvc.pt/music.owl#D1" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#D2" />
    <pronto:certainty>0.7;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:object rdf:resource="http://lvc.pt/music.owl#D2" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#D3" />
    <pronto:certainty>0.7;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:object rdf:resource="http://lvc.pt/music.owl#D3" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#D4" />
    <pronto:certainty>0.7;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#Gig" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:object rdf:resource="http://lvc.pt/music.owl#Show" />
    <pronto:certainty>0.7;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#Frontman" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:object rdf:resource="http://lvc.pt/music.owl#LeadSinger" />
    <pronto:certainty>0.9;1</pronto:certainty>
  </owl11:Axiom>
- <owl11:Axiom>
    <rdf:subject rdf:resource="http://lvc.pt/music.owl#LeadSinger" />
    <rdf:predicate rdf:resource="http://www.w3.org/2002/07/owl#equivalentClass" />
    <rdf:object rdf:resource="http://lvc.pt/music.owl#Vocalist" />
    <pronto:certainty>0.8;1</pronto:certainty>
  </owl11:Axiom>
```

Figura A.6: Ontologia PLVCMO

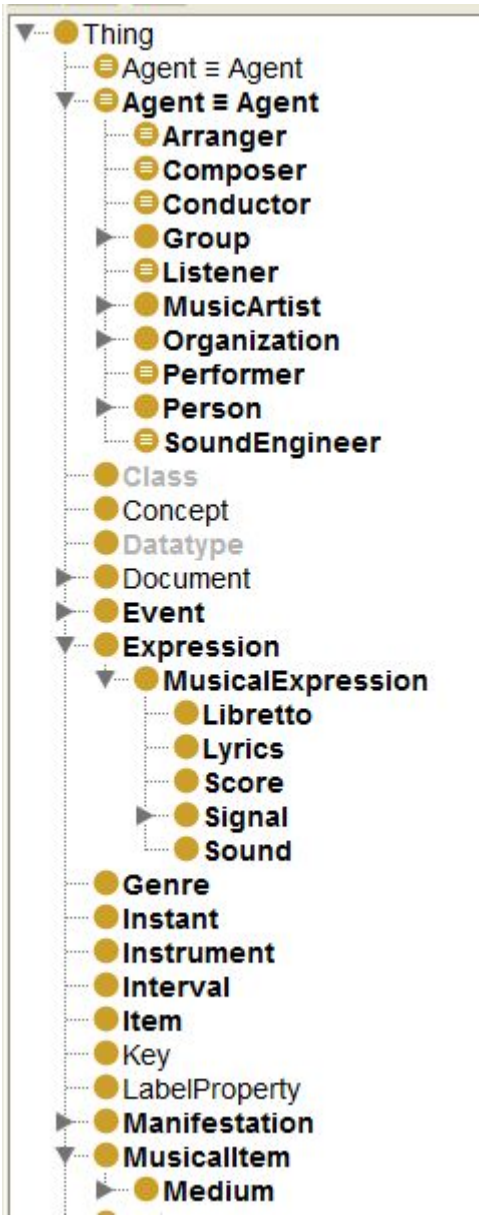

Figura A.7: Ontologia MO - Music Ontology

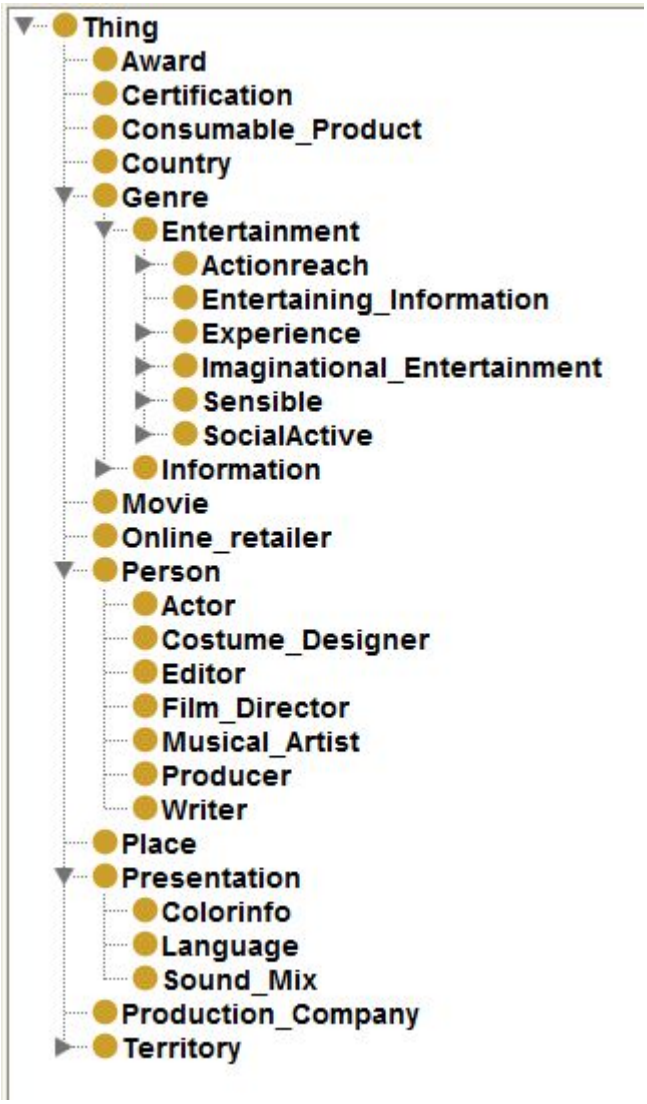

Figura A.8: Ontologia MO - Movie Ontology

Apêndice B

Subsunção e equivalência probabilística

Os contactos mantidos com o investigador da API Pronto (P. Kilinov), permitiram apurar as causas subjacentes à limitação que implica utilizar a certeza como limite superior de probabilidade na realização de inferência de equivalência probabilística.

Na teoria das probabilidades (ToP) a probabilidade condicional é dada por $\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$ (**?**). No entanto, no Pronto esta probabilidade é calculada assumindo $\mathbb{P}(B) = 1$ e calculado apenas $\mathbb{P}(A \cap B)$. O Pronto assume esta simplificação para possibilitar a herança de propriedades probabilísticas (que não é suportada em ToP). Como exemplo considere-se a restrição condicional $(Voa|Passaro)[0.9, 1]$ e o axioma $PassaroAmarelo \sqsubseteq Passaro$. Em ToP tem-se que $(Voa|PassaroAmarelo)[0, 1]$ o que não permite concluir nada em relação à probabilidade de $PassaroAmarelo$ voar. Usando a formula de cálculo do Pronto no entanto tem-se que $(Voa|PassaroAmarelo)[0.9, 1]$. Esta abordagem tem no entanto efeitos colaterais.

Ao considerarmos $(A|B)[0.5, 0.5] \sqcap (B|A)[0.5, 0.5]$, qualquer tentativa de calcular a probabilidade de $(C|A)$ ou $(C|B)$ (para qualquer classe C) resultará numa inconsistência uma vez que $(A|B)[0.5, 0.5]$, $(B|A)[0.5, 0.5]$ são inconsistentes com $\mathbb{P}(B) = 1$ ou $\mathbb{P}(A) = 1$. Assim é necessário optar entre suportar herança de propriedades (com os efeitos se-

cundários descritos) ou, não suportar herança e ter inferências inconclusivas.

No Pronto a opção escolhida foi a de suportar herança. Para ultrapassar esta limitação e assim conseguir realizar inferência de equivalência é necessário que o limite superior da probabilidade seja sempre 1. Desta forma tem-se que $(A|B)[0.5, 1]$, $(B|A)[0.5, 1]$ são consistentes para $\mathbb{P}(B) = 1$ ou $\mathbb{P}(A) = 1$.

Apêndice C

Extensão do Pronto para simplificar a descrição probabilística

A API Pronto contém o método `loadDefaultConstraintsFromOWL` (na classe `ProntoLoaderUtils`) que percorre todos os axiomas do tipo SUBCLASS-OF [1] existentes na ontologia, criando para cada um uma restrição condicional com as classes envolvidas na subsunção e a probabilidade associada. O algoritmo de *Lexicographic Entailment* utiliza depois estas restrições condicionais para realizar a inferência probabilística de subsunção. Para suportar a modelação de equivalência probabilística na ontologia, este método foi alterado para percorrer também todos os axiomas do tipo EQUIVALENT-CLASSES [2], criando para cada um duas restrições condicionais com as subsunções associadas à equivalência (usando para cada uma a probabilidade associada à equivalência). Realizar a inferência probabilística de equivalência resume-se a efectuar duas inferências probabilísticas de subsunção. No entanto, a obtenção do conjunto de classes semelhantes (equivalentes com uma probabilidade) de determinada classe é realizada em dois passos. Primeiro obtêm-se o conjunto de classes equivalentes e depois infere-se a probabilidade de cada equivalência. Desta forma pode-se optimizar a inferência probabilística de equivalência efectuando apenas uma inferência probabilística de subsunção.

[1] `http://www.w3.org/2000/01/rdf-schema#subClassOf`
[2] `http://www.w3.org/2002/07/owl#equivalentClass`

Apêndice D

Analisador LVCAnalyzer

Listing D.1: LVCAnalyser

```
public class LVCAnalyser extends Analyzer {
  private String name;
  private final Version matchVersion;

  public LVCAnalyser(Version matchVersion, String name) {
                this.name = name;
                this.matchVersion = matchVersion;
        }

  @Override
  public TokenStream tokenStream(String fieldName,
                                 Reader reader) {
    TokenStream result = new StandardTokenizer(matchVersion,
                                               reader);
    result.addAttribute(TermAttribute.class);
    result = new StandardFilter(result);
    result = new LowerCaseFilter(result);
    result = new StopFilter(true, result,
                 StopAnalyzer.ENGLISH_STOP_WORDS_SET);
    result = new PorterStemFilter(result);
    return result;
  }
}
```

Apêndice E

Informação dos TSs

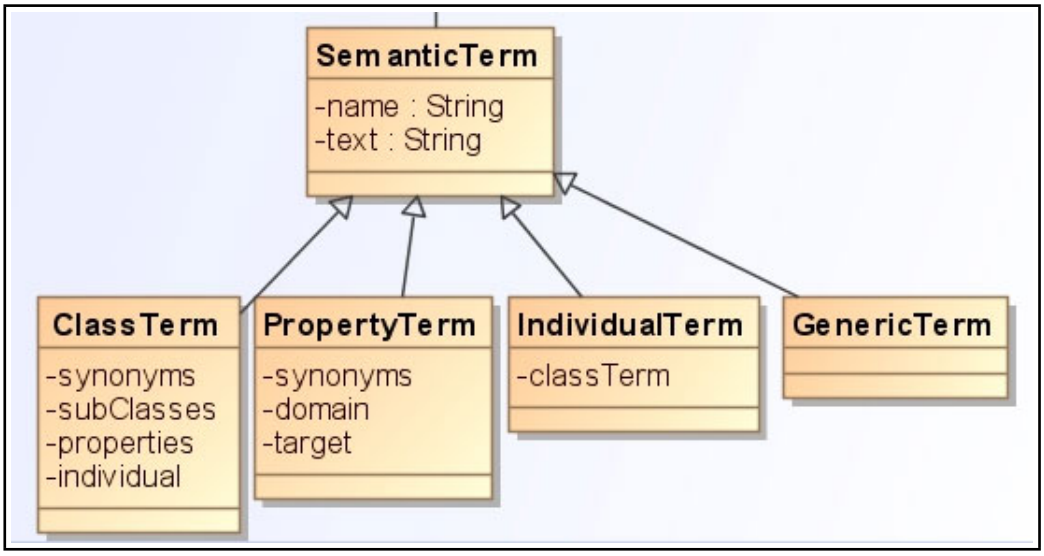

Figura E.1: Informação por Termo Semântico

Apêndice F

Algoritmo para tratamento de TSs não determinados

1. Percorrer a lista de TSs da FS, criando grupos de TSs, contíguos, não determinados;
2. para cada grupo encontrado:
   (a) tentar obter um TS determinado, tsd, a partir dos TSs existentes no grupo;
   (b) se existir tsd ou o grupo for vazio passar ao próximo passo, caso contrário remover o último TS do grupo e voltar ao passo anterior;
   (c) se existir tsd substituir na lista de TSs da FS os termos que originaram tsd por tsd;

Apêndice G

Algoritmo para expandir uma FS

A expansão de uma FS é obtida por um algoritmo recursivo que recebe como parâmetros uma lista de TSs, um coeficiente semântico, uma lista de substituições e um parâmetro de acumulação. A função vai recursivamente construindo uma expansão no parâmetro de acumulação (actualizando o coeficiente semântico e a lista de substituições) e termina quando a lista de TSs está vazia retornando a expansão apurada. Nos níveis intermédios da recursividade o algoritmo copia a lista de TSs (para poder efectuar várias invocações recursivas mantendo a lista de TSs local a cada invocação) e depois remove o primeiro termo das lista aplicando um processamento de acordo com o tipo do termo.

Se o TS for do tipo `IndividualTerm` tem-se:

1. Adicionar o nome do individuo à expansão,
2. Invocar recursivamente obtendo uma lista de expansões,
3. Caso existam propriedades na classe do individuo, para cada propriedade,
   (a) Duplicar a lista de TSs de forma a obter uma lista independente,
   (b) Adicionar a propriedade ao inicio da lista de TSs,
   (c) Invocar recursivamente obtendo uma lista de expansões,

(d) Juntar o resultado à lista de expansões,

4. Adicionar a classe do individuo ao inicio da lista de TSs,
5. Invocar recursivamente obtendo uma lista de expansões,
6. Juntar o resultado à lista de expansões,
7. Retornar a lista de expansões.

Se o TS for do tipo `ClassTerm` tem-se:

1. Obter um dicionário com a classe, todas as classes equivalentes (e suas subclasses), com o respectivo coeficiente de equivalência,
2. Para cada classe no dicionário,
   (a) Adicionar o nome da classe à expansão
   (b) Se a classe tiver associado um individuo,
      i. Duplicar a lista de substituições de forma a obter uma lista independente,
      ii. Adicionar o individuo ao dicionário associando-lhe a classe,
   (c) Invocar recursivamente actualizando o coeficiente semântico (cs = cs * coeficiente de equi.) obtendo uma lista de expansões,
   (d) Caso existam propriedades associadas à classe, para cada propriedade,
      i. Duplicar a lista de TSs de forma a obter uma lista independente,
      ii. Adicionar a propriedade ao inicio da lista de TSs,
      iii. Invocar recursivamente actualizando o coeficiente semântico (cs = cs * coeficiente de equi.) obtendo uma lista de expansões,
      iv. Juntar o resultado à lista de expansões,
3. Retornar a lista de expansões.

Se o TS for do tipo `PropertyTerm` tem-se:

1. Obter uma lista com a propriedade e propriedades equivalentes,
2. Para cada propriedade na lista,
   (a) Adicionar o nome da propriedade à expansão,
   (b) Invocar recursivamente obtendo uma lista de expansões,
   (c) Retornar a lista de expansões.

Se o TS for do tipo `GenericTerm` tem-se:

1. Adicionar o nome do termo à expansão,
2. Invocar recursivamente obtendo uma lista de expansões,
3. Retornar a lista de expansões.

De notar que a utilização da forma canónica dos termos na expansão não representa um problema uma vez que na fase de pesquisa é aplicado o analisador `LVCAnalyzer` ao critério de pesquisa resultando a eliminação dos caracteres *underscore*.

Apêndice H

Configuração da aplicação nos testes

Tabela H.1: Configuração da aplicação nos testes

Parâmetro	Descrição	Valor
Ontologias	LVCMO - LVC Music Ontology PLVCMO - Probabilistic LVC Music Ontology	-
USE-SUB-EXPANSIONS	Controla o uso ou não de sub-expansões	true
SUB-EXPANSION-MIN-LENGTH	Número mínimo de termos em qualquer sub-expansão.	2
INCLUDE-SUBCLASSES	Indica se devem ser consideradas as subclasses na construção das expansões.	true
SUB-CLASS-FACTOR	Determina a maneira como o coeficiente semântico é afectado quando uma classe é substituída por uma subclasse.	1
USE-PRONTO	Indica se é usada ou não a componente probabilística para inferir conceitos semelhantes.	true
SUBEXPANSION-PONDERATION-FACTOR	Controla a ponderação do coeficiente semântico nas sub-expansões (c.f. **??** na página **??**).	4
LUCENE-DIRECTORY	Directoria de indexação do Lucene utilizada para guardar a indexação dos documentos. Pode ser baseada em ficheiro(s) ou memória	memória
PHRASE-SLOP-LENGTH-FACTOR	Controla o valor do parâmetro PHRASE-SLOP em função do critério de pesquisa (c.f. **??** na página **??**).	4
PHRASE-SLOP-LENGTH-MIN	Indica o valor mínimo que o parâmetro PHRASE-SLOP admite.	10