

RENATO BESEN

**BUSCA DIRIGIDA POR ONTOLOGIAS E INFORMAÇÕES DE
CONTEXTO**

Florianópolis

2009

RENATO BESEN

**BUSCA DIRIGIDA POR ONTOLOGIAS E INFORMAÇÕES DE
CONTEXTO**

Relatório final do desenvolvimento do Trabalho
de Conclusão de Curso, parte dos requisitos para
aprovação na disciplina Projeto 2

Orientador:
Renato Fileto

Co-orientador:
Caio Stein D'Agostini

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

Florianópolis

2009

SUMÁRIO

Lista de Figuras	
Resumo	
1 INTRODUÇÃO.....	p. 10
1.1 JUSTIFICATIVA	p. 11
1.2 ESCOPO DO TRABALHO	p. 12
1.3 OBJETIVOS	p. 13
1.3.1 OBJETIVO GERAL	p. 13
1.3.2 OBJETIVOS ESPECÍFICOS	p. 13
1.4 ORGANIZAÇÃO DA MONOGRAFIA.....	p. 13
2 FUNDAMENTOS.....	p. 14
2.1 RECUPERAÇÃO DE INFORMAÇÃO	p. 14
2.2 WEB SEMÂNTICA	p. 15
2.2.1 ANOTAÇÃO SEMÂNTICA	p. 15
2.2.2 ONTOLOGIAS	p. 16
2.2.3 BUSCA SEMÂNTICA	p. 17
2.3 CONTEXTOS DE USUÁRIOS	p. 17
2.3.1 GRAFO DE TÓPICOS	p. 18
3 FERRAMENTAS.....	p. 20
3.1 KIM - KNOWLEDGE & INFORMATION MANAGEMENT	p. 20
3.1.1 BASE DE CONHECIMENTO	p. 20

3.1.2	KIM SERVER.....	p. 22
3.1.3	APIS DO KIM	p. 23
4	PROTÓTIPO.....	p. 26
4.1	FLUXO DA BUSCA	p. 26
4.2	ARQUITETURA	p. 27
4.2.1	PACOTE CLIENT - GERENCIAMENTO DE CONTEXTOS.....	p. 27
4.2.2	PACOTE SERVER	p. 29
4.3	RESULTADO	p. 33
5	EXPERIMENTOS.....	p. 35
5.1	EXEMPLO DE INTERAÇÃO	p. 35
5.2	PROBLEMAS ENCONTRADOS	p. 39
5.2.1	ENTIDADES DESNECESSÁRIAS	p. 39
5.2.2	FOCO EM ENTIDADES NOMEADAS	p. 39
5.2.3	NECESSIDADE DE CUSTOMIZAÇÃO DO KIM.....	p. 40
6	CONCLUSÕES.....	p. 41
6.1	TRABALHOS FUTUROS	p. 42
APÊNDICE A – ARTIGO	p. 43	
BUSCA DIRIGIDA POR ONTOLOGIAS E INFORMAÇÕES DE CONTEXTO.....	p. 43	
INTRODUÇÃO	p. 43	
FUNDAMENTOS.....	p. 44	
FERRAMENTAS E PROTÓTIPO	p. 45	
EXPERIMENTO.....	p. 46	
CONCLUSÕES E TRABALHOS FUTUROS	p. 48	
APÊNDICE B – CÓDIGO FONTE.....	p. 50	

B.1 PROJETO SYSTEM.....	p. 50
B.1.1 SYSTEM.CLIENT.CONTEXT.....	p. 50
B.1.2 SYSTEM.CLIENT.FORAGINGENGINE.....	p. 51
B.1.3 SYSTEM.CLIENT.FORAGINGENGINE.METRICS.....	p. 68
B.1.4 SYSTEM.COMMON.....	p. 69
B.1.5 SYSTEM.SERVER.DEFINITIONS.....	p. 72
B.1.6 SYSTEM.SERVER.REPOSITORY.....	p. 73
B.2 PROJETO SYSTEMIMPL.....	p. 74
B.2.1 SYSTEMIMPL.....	p. 74
B.2.2 SYSTEMIMPL.CLIENT.CONTEXT.....	p. 75
B.2.3 SYSTEMIMPL.CLIENT.GUI.....	p. 91
B.2.4 SYSTEMIMPL.SERVER.....	p. 128
REFERÊNCIAS.....	p. 139

LISTA DE FIGURAS

Figura 1	Arquitetura do projeto	12
Figura 2	Exemplo de grafo de tópicos	18
Figura 3	Arquitetura da plataforma KIM	21
Figura 4	Nível <i>top</i> da ontologia PROTON	22
Figura 5	Fluxo da busca	27
Figura 6	Estrutura da ferramenta	29
Figura 7	Screenshot do protótipo.	34
Figura 8	Primeira pesquisa por “boeing”	36
Figura 9	Segunda pesquisa por “boeing”	37
Figura 10	Pesquisa por “Europe” após pesquisa por “boeing”	38
Figura 11	Exemplos de entidades desnecessárias: ”100%”, ”15%”, ”June”.	39
Figura 12	Fluxo da busca	46
Figura 13	Screenshot do protótipo.	47

LISTA DE ALGORITMOS

3.1	Acesso à API do KIM.	p. 25
3.2	Acesso a componentes específicos.	p. 25
4.1	Possíveis denotações.	p. 30
4.2	Possíveis nomes.	p. 31
4.3	Recuperação de documentos.	p. 31
4.4	Anotações de um documento.	p. 32
B.1	Association.java.....	p. 50
B.2	NonExistingTopicException.java	p. 50
B.3	Topic.java	p. 50
B.4	TopicGraphContext.java	p. 51
B.5	Core.java	p. 52
B.6	Expansion.java	p. 65
B.7	GuiInterface.java	p. 67
B.8	Metric.java	p. 68
B.9	MetricA.java	p. 68
B.10	Result.java.....	p. 69
B.11	Definitions.java	p. 72
B.12	KBResult.java	p. 72
B.13	Repository.java.....	p. 73
B.14	CONSTANTS.java	p. 74
B.15	Launcher.java	p. 74
B.16	AssociationPrefuse.java	p. 75

B.17 PrefuseTopicGraphContext.java	p. 76
B.18 TopicPrefuse.java	p. 89
B.19 DisambDialog.java	p. 91
B.20 Gui.java	p. 94
B.21 IO.java	p. 116
B.22 LabelledEdgeLayout.java	p. 121
B.23 Painel.java	p. 122
B.24 Defs.java	p. 128
B.25 KIMFactory.java	p. 131
B.26 Rep.java	p. 134

RESUMO

Ontologias atendem às necessidades das máquinas na descrição formal e manipulação de conhecimento compartilhado por conjuntos de usuários. A comunicação humana, no entanto, depende também de informações subjetivas que podem ser coletadas e mantidas no contexto relativo a cada indivíduo, para permitir uma melhor interpretação das informações trocadas e o alinhamento das visões de quem envia e quem recebe uma mensagem. Em um sistema de busca, informações individuais ajudam na desambiguação de consultas e podem ser usadas para trazer outros resultados relevantes para o usuário, mesmo que não estejam lexicamente relacionados com as palavras-chave da consulta.

O objetivo deste trabalho é integrar um módulo de coleta e utilização de informações de contexto relativo a uma ontologia com uma plataforma de busca semântica, visando melhorar a usabilidade do sistema, além da precisão e cobertura dos resultados obtidos. O trabalho revisa os conceitos envolvidos, apresenta a arquitetura proposta para o sistema de busca semântica com informação de contexto e relata experiências obtidas na implementação das extensões para coleta e utilização de informação de contexto sobre a plataforma KIM.

1 INTRODUÇÃO

A quantidade de conteúdo na *Web* aumenta diariamente, com páginas variando de notícias até artigos científicos. Existem na *Web* bases com centenas de milhares de documentos, como por exemplo a IEEE Xplore ¹, que possui mais de 1 milhão de textos completos de artigos científicos e recebe adições constantemente. Outro exemplo é a Biblioteca Digital de Teses e Dissertações do IBICT ², que busca integrar sistemas de informação de teses e dissertações de instituições de ensino de todo o país em um único portal.

A grande maioria das bases de dados disponíveis fornecem um sistema de indexação baseado em metadados, como palavras-chave, título e autor, e também na contagem de ocorrências de palavras no texto. Porém, a simples indexação léxica do conteúdo de campos de metadados pode ser ineficaz em algumas situações. Uma destas situações ocorre quando o usuário não conhece o termo utilizado para descrever um determinado assunto, ou conhece apenas um sinônimo, tornando difícil encontrar informações relevantes sobre o assunto desejado. Suponha por exemplo que um usuário queria aprender sobre a capacidade de um programa manter informação sobre si próprio, e usar tal informação e observações do ambiente para modificar suas próprias estruturas e comportamento quando necessário. O usuário pode não saber que o termo usado para esta capacidade é "reflexão computacional", e até que descubra sobre o termo correto será difícil encontrar documentos sobre o assunto. Outro problema é caso um mesmo termo tenha vários significados (homonímia). Neste caso, a precisão pode ser prejudicada ao incluir nos resultados conteúdos descritos por termos ambíguos (ou seja, que não interessam ao usuário). Com base no mesmo exemplo, suponha que o usuário agora busque por "reflexão" em um sistema de buscas. As respostas vão incluir, além dos documentos desejados, documentos sobre "reflexão" mental, o fenômeno físico "reflexão" e sobre a transformação geométrica "reflexão" da matemática, entre outros possíveis significados desta palavra .

Na *Web Semântica*, idealizada por Tim Berners-Lee (FRAUENFELDER, 2004) , os conteúdos são descritos semanticamente, possibilitando assim que suas descrições semânticas sejam pro-

¹<http://ieeexplore.ieee.org/>

²<http://bdt2.ibict.br/>

cessadas por um computador. Uma forma de anotar os conteúdos é de acordo com metadados definidos de acordo com uma ontologia, que fornece uma definição formal de conceitos, instâncias e seus relacionamentos em um determinado domínio de interesse.

O formalismo de uma ontologia fornece interoperabilidade e permite que diferentes entidades se comuniquem. Em contrapartida, este formalismo demanda a disponibilidade de especialistas do domínio da ontologia para organizar o conhecimento nela descrito. Além disso, o usuário de um sistema de busca baseado em ontologia deve ter uma visão do domínio semelhante à visão da ontologia. No caso de ontologias extensas e complexas, o sistema pode não fornecer resultados satisfatórios para o usuário.

Este trabalho visa incorporar um mecanismo de coleta, representação e uso de informações de contexto do usuário a uma ferramenta de busca semântica, para contemplar a individualidade do usuário. Neste trabalho, o contexto, quando aplicado à recuperação de informações, pode ser descrito como uma parte do conhecimento de um usuário que afeta a comunicação entre duas entidades (MANI; SUNDARAM, 2007). As entidades correspondem ao usuário e o sistema, e a parte do conhecimento é a visão individual do usuário sobre um domínio. No nosso trabalho o contexto do usuário (seu conhecimento, modo de ver o mundo e preferências) é definido em relação a uma ontologia (conhecimento compartilhado).

1.1 JUSTIFICATIVA

A descrição fornecida pelas ontologias atende as necessidades das máquinas. Um *software*, por exemplo, pode buscar por eventos de tecnologia e montar um calendário com as datas dos eventos. Contudo, a comunicação humana depende também de informações subjetivas. Ela depende de um *feedback* entre as partes envolvidas, permitindo uma melhor interpretação das informações e conseqüentemente um alinhamento das visões de quem envia e quem recebe uma mensagem. Outros detalhes como experiência e informações secundárias também são importantes para que os significados corretos possam ser compartilhados (DEGLER; LEWIS, 2004). Ontologias, por serem descrições formais de um domínio, não são adequadas para armazenar este tipo de informação subjetiva. Porém, estas informações podem ser obtidas dinamicamente observando-se o comportamento do usuário e o contexto no qual ele está inserido.

Com a utilização de informação de contexto, os resultados de buscas podem ser mais precisos, completos e produzidos mais eficientemente, melhorando a experiência do usuário. As informações individuais ajudam na desambiguação de palavras-chave e podem ser usadas para trazer outros resultados relevantes para o usuário naquele contexto, mesmo que não este-

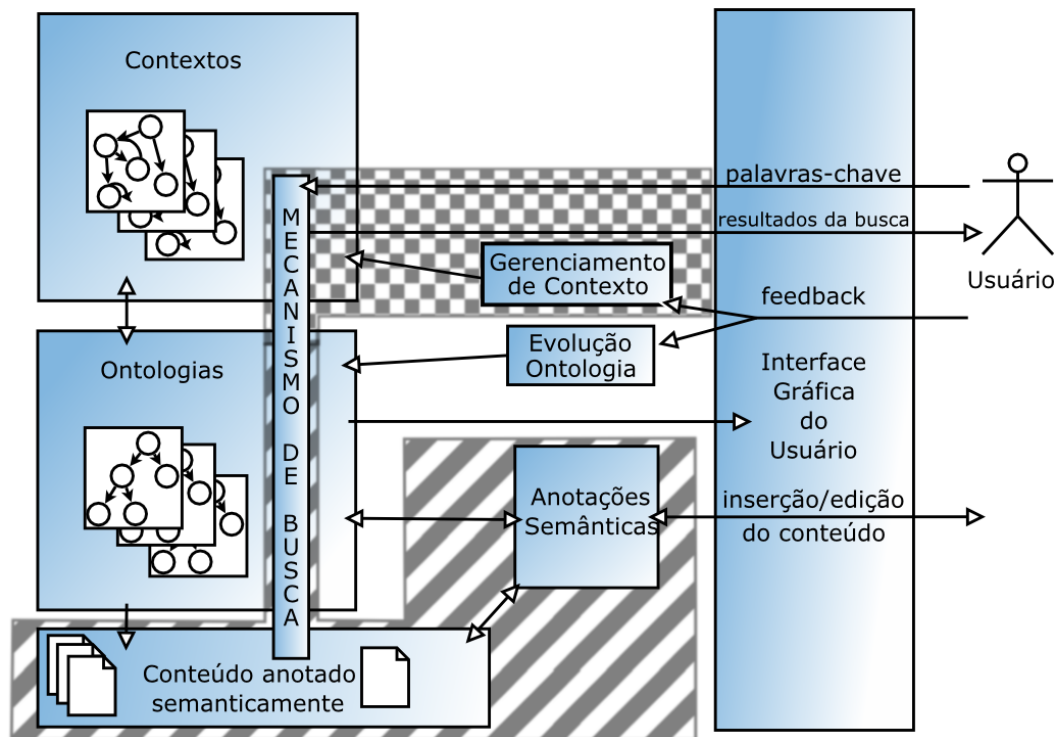


Figura 1: Arquitetura do projeto

jam diretamente associados com as palavras da busca. Além disso, como o contexto não afeta globalmente o sistema, não se perde a interoperabilidade do mesmo.

1.2 ESCOPO DO TRABALHO

Este trabalho está sendo desenvolvido no âmbito do projeto **Busca Semântica em Ambientes Distribuídos usando Informação de Contexto**, financiado pelo CNPq (processo 48139212007-6). Um dos subprojetos envolvidos visa desenvolver processos para captura e utilização de informação sobre o contexto do usuário, com a expectativa de melhorar a qualidade dos resultados e do próprio processo de busca (D'AGOSTINI et al., 2008).

A Figura 1 apresenta a arquitetura geral do sistema proposto no projeto mencionado acima. A parte destacada com o padrão xadrez mostra o foco de implementação deste trabalho, que inclui o gerenciamento de informação de contexto e o uso dessa informação no processamento de consultas. A parte com linhas diagonais referem-se a módulos relacionados ao trabalho, que devem ser integrados de outros sistemas.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Integrar um sistema de coleta e representação de informações de contexto a uma ferramenta de busca semântica, possibilitando uma melhora na precisão e revocação do processo de busca.

1.3.2 OBJETIVOS ESPECÍFICOS

1. Implementar uma API para comunicação entre o sistema de busca semântica e os mecanismos de captura e uso de informações de contexto;
2. Implementar um módulo que use informações de contexto para expandir e desambiguar buscas;
3. Classificar os resultados de acordo as denotações das palavras-chave usadas nas buscas e a relevância estimada para o usuário.

Entre os requisitos do trabalho estão: (i) o processo de busca deve fornecer um *feedback* para o sistema, na escolha de denotações de palavras-chaves e na apreciação dos resultados pelo usuário, para que o sistema possa atualizar as informações de contexto, e (ii) o sistema criado deve ser modular, para que a funcionalidade produzida possa ser integrada ao projeto maior do qual faz parte

1.4 ORGANIZAÇÃO DA MONOGRAFIA

Este trabalho está organizado da seguinte maneira: o capítulo 2 apresenta alguns fundamentos básicos para o entendimento do sistema proposto; o capítulo 3 apresenta a plataforma utilizada no desenvolvimento; o capítulo 4 descreve o protótipo criado, com alguns detalhes de sua implementação, principalmente no que se refere à integração com a plataforma discutida no capítulo 3; o capítulo 5 demonstra a utilização do protótipo e discute alguns problemas encontrados durante o seu desenvolvimento; finalmente o capítulo 6 apresenta as conclusões do trabalho, e a sugestão de alguns trabalhos futuros.

2 FUNDAMENTOS

Este capítulo apresenta alguns conceitos e tecnologias considerados essenciais para o entendimento do trabalho.

2.1 RECUPERAÇÃO DE INFORMAÇÃO

Não há muita utilidade em possuir um grande volume de dados se não é possível encontrar uma determinada informação quando se precisa dela. A recuperação de informação possui um papel fundamental em qualquer situação onde se armazene uma quantidade razoável de documentos.

A recuperação de informação consiste em, dada uma busca de um usuário, determinar os documentos relevantes para esta busca e este usuário dentro de uma coleção de documentos. O objetivo de um sistema de buscas é maximizar a **precisão** e a **revocação** (também chamada cobertura) (MANGOLD, 2007) dos resultados obtidos, onde:

$$precisao = \frac{\text{numero_de_documentos_relevantes_recuperados}}{\text{numero_de_documentos_recuperados}}$$

$$revocacao = \frac{\text{numero_de_documentos_relevantes_recuperados}}{\text{numero_de_documentos_relevantes}}$$

O valor máximo de ambas as medidas, precisão e revocação, é 1. Supondo por exemplo uma coleção com 100 documentos, se uma pesquisa "x" retorna 1 documento dentre 10 documentos relevantes dentro da coleção, temos uma precisão de 1, ou 100%, porém uma revocação de apenas 0.1, ou 10%. Outra possibilidade seria o sistema retornar os 10 documentos relevantes e mais 10 documentos sem relação com a consulta "x". Neste caso a revocação seria de 100%, enquanto a precisão seria de apenas 50%.

Um sistema de recuperação de informações depende diretamente de como o usuário vai fazer a busca e da visão lógica dos documentos no sistema. O usuário necessita traduzir a sua necessidade de informação em uma "expressão" que será analisada pelo sistema de busca.

Normalmente essa tradução é simplesmente a escolha de um conjunto de termos que transmita a semântica da necessidade de informação. Os documentos normalmente são representados no sistema com um conjunto de termos de indexação e palavras-chave, que são a visão lógica do documento no sistema (BAEZA-YATES; RIBEIRO-NETO et al., 1999).

2.2 WEB SEMÂNTICA

Atualmente boa parte dos dados disponíveis na *web* pode ser lido, porém não pode ser facilmente processados por uma máquina. Isto significa que os dados existem mas os computadores não sabem do que se tratam nem como se relacionam.

Hoje na *web* os métodos de busca se baseiam na busca sintática de documentos. Por este motivo, há uma limitação na capacidade de recuperação de documentos, pois as buscas não levam em conta significados e relações dos termos pesquisados. Por serem limitados à estrutura sintática dos documentos, os sistemas de busca necessitam de melhorias externas, como por exemplo o uso de meta-dados (e.g. autor, palavras-chave, áreas relacionadas). Este tipo de abordagem é muito adotado em bancos de artigos científicos. Outros mecanismos mais sofisticados podem ser utilizados, como contagem e relevância de *links*, utilizado pelo PageRank (BRIN; PAGE, 1998).

Na *Web Semântica* os dados são anotados semanticamente, possibilitando assim que sistemas automatizados raciocinem sobre eles. Por exemplo, acessando um calendário o computador seria capaz de saber qual o seu próximo compromisso, configurar um GPS para traçar a rota até o local, olhar a previsão de tempo do local naquele horário, enviar um e-mail de lembrete para os contatos associados àquele evento, dentre outras coisas. O computador seria capaz de interpretar os dados, e não apenas apresentá-los na tela.

2.2.1 ANOTAÇÃO SEMÂNTICA

Uma forma de dar significado à *web* é criando ontologias, e então ligando os dados de páginas aos conceitos descritos nas mesmas (BERNERS-LEE; HENDLER; LASSILA, 2001). Estas ligações entre dados e conceitos é chamada **anotação semântica**. A qualidade do processo de anotação tem impacto direto nos serviços oferecidos na *Web Semântica*.

A anotação de um documento pode ser feita manualmente, porém humanos podem gerar anotações não muito confiáveis por vários motivos, como complexidade do esquema de anotação, diferentes níveis de familiaridade com o assunto e motivação pessoal (BAYERL et

al., 2003). A anotação manual tem também outros problemas, como a dificuldade de se utilizar mais de uma ontologia para anotação (diferentes pontos de vista para um mesmo documento). Também devem ser levados em consideração o volume de dados disponível na *web* e a possível modificação de documentos e conceitos.

Contudo, anotações totalmente automatizadas também são difíceis de serem realizadas. Devido a dificuldade de um sistema identificar precisamente todas as entidades em um documento automaticamente, propostas de anotação semi-automática estão sendo desenvolvidas. Desta forma, uma plataforma de anotação faz boa parte do trabalho, mas ainda necessita de intervenção humana em algum ponto.

Com este tipo de plataforma, o problema de escalabilidade na anotação de documentos pode ser superada e o custo de anotação de novos documentos é diminuído. Outros possíveis benefícios são a capacidade de anotar consistentemente documentos com uma ontologia e o fato de se poder usar múltiplas ontologias para anotar um documento (REEVE; HAN, 2006).

2.2.2 ONTOLOGIAS

O termo ontologia vem do campo da filosofia que estuda o ser, ou a existência. Em filosofia, podemos nos referir a uma ontologia como a teoria sobre a natureza da existência. Já em ciências da computação e informação, o termo ontologia é usado para denotar um artefato projetado para representar os conhecimentos sobre um domínio (GRUBER, 2008).

Segundo Davies, Studer e Warren (2006) uma definição amplamente aceita de ontologia é “uma especificação explícita e formal da conceitualização de um domínio de interesse”. Esta definição toca em dois pontos importantes: por ser formal, máquinas de inferência podem ser usadas para raciocinar sobre a ontologia; e ontologias descrevem um certo domínio de interesse, desde domínios mais específicos até ontologias de alto nível, modelando grandes quantidades de conhecimento.

A estrutura de uma ontologia pode ser descrita como uma 4-tupla (C,R,I,A), onde C é um conjunto de conceitos (ou classes), R é um conjunto de relacionamentos (ou propriedades), I é um conjunto de instâncias e A é um conjunto de axiomas.

WEB ONTOLOGY LANGUAGE - OWL

OWL é uma linguagem de marcação semântica, desenvolvida para a publicação e compartilhamento de ontologias na *web*. OWL é uma recomendação da W3C ¹, feita como uma extensão

¹<http://www.w3.org/TR/owl-features/>

da RDF².

Existem três sub-linguagens de OWL: OWL-Lite, OWL-DL e OWL-Full, da menos expressiva para a mais expressiva, respectivamente. OWL-Lite possui restrições simples e por isso é a mais eficiente das três opções. OWL-DL pode ser usada por quem deseja uma máxima expressividade com garantia de que todas as computações feitas sobre ela são decidíveis. OWL-Full fornece uma máxima expressividade e a liberdade sintática da RDF, ao custo da garantia de decidibilidade.

2.2.3 BUSCA SEMÂNTICA

Hoje as ferramentas de busca se baseiam principalmente na contagem das palavras em um documento para determinar sua relevância. Ferramentas como o *Google*³ também se baseiam nas estruturas de *links* das páginas para aumentar a precisão das buscas (GUHA; MCCOOL; MILLER, 2003).

A busca semântica tenta tirar proveito das informações do domínio disponíveis. Quando é feita a busca por um conceito, podemos incluir documentos que não seriam retornados em uma busca comum, mas que estão relacionados semanticamente de alguma forma com os conceitos pesquisados. Caso uma busca ambígua seja feita, há a possibilidade de desambiguar ou separar os resultados por categorias.

2.3 CONTEXTOS DE USUÁRIOS

O contexto pode ser definido como um conjunto de informações que auxilia na inferência de intenções de um usuário. A comunicação humana é facilitada por este tipo de informação. Por exemplo, em uma loja de roupas, informações como a estação do ano e o sexo do cliente ajudam a inferir qual tipo de roupa o cliente procura.

Em relação à interação homem-máquina, o contexto é um conjunto de atributos que pode ser usado para tomada de decisões sem a necessidade de interromper a atividade do usuário (SATYANARAYANAN et al., 2001). Na recuperação de informações, as informações de contexto podem ser usadas para desambiguar buscas, inferindo as intenções do usuário. Por exemplo, “São Paulo” pode se referir a uma cidade, um estado ou um time de futebol. A intenção do usuário pode ser inferida a partir do contexto. Se em uma busca anterior o usuário procurou por futebol, provavelmente “São Paulo” se refere ao time de futebol. Se foram buscados documen-

²<http://www.w3.org/RDF/> - Resource Description Framework

³<http://www.google.com>

tos sobre eleições municipais, provavelmente “São Paulo” se refere à cidade (D’AGOSTINI et al., 2007).

Contextos fornecem a base para a personalização do sistema de busca. Cada interação do usuário com o sistema pode alimentar o contexto, permitindo o sistema saber quais conceitos foram pesquisados e quais documentos foram relevantes para o usuário.

2.3.1 GRAFO DE TÓPICOS

Grafos de tópicos, algumas vezes chamado de mapa de tópicos, servem para representar estruturas de conhecimento e associá-las a outros recursos. Estas representações permitem a construção de tecnologia para gerenciamento de informação. Grafos de tópicos podem ser comparados em função com glossários e tesouros, porém fornecem muito mais flexibilidade, pois permitem a capacidade de criar associações entre tópicos e recursos externos. Grafos de tópicos permitem a criação de estruturas altamente complexas para navegação em grandes quantidades de dados, mesmo com muitas interconexões.

A estrutura básica de um grafo de tópico são os tópicos, as associações e as ocorrências (PEPPER, 2000). Tópicos são os conceitos gerais no mapa, e podem representar qualquer coisa. Para o contexto de usuário, os tópicos são representações das entidades pesquisadas. Associações por sua vez representam a relação entre dois diferentes tópicos. No contexto, uma associação é criada por exemplo quando um documento é anotado com duas entidades, formando uma relação entre as mesmas. As ocorrências por sua vez representam recursos associados a um tópico, e normalmente estão presentes fora do grafo de tópicos. No contexto, as ocorrências são as próprias entidades presentes na ontologia. A figura 2 mostra um exemplo de grafo.

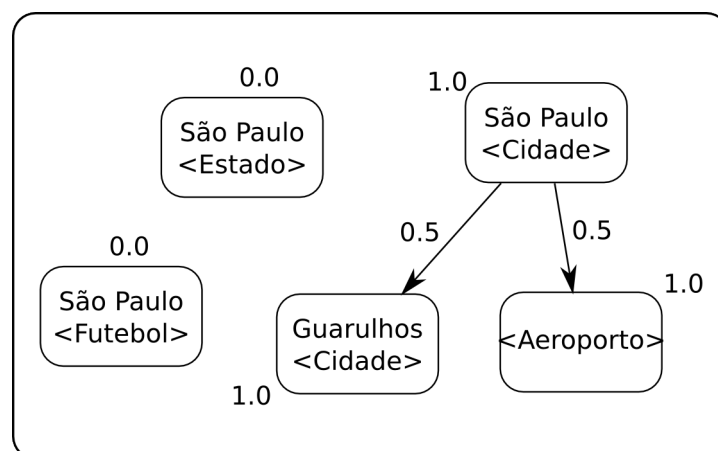


Figura 2: Exemplo de grafo de tópicos

Neste trabalho pesos são associados a cada tópico e associação, os quais podem ser utili-

zados para calcular a relevância de um documento para uma determinada busca. Estes pesos devem ser normalizados a cada iteração. Podem também ser usados valores mínimos para os pesos, eliminando tópicos em que o usuário tenha perdido o interesse.

3 FERRAMENTAS

Neste capítulo é analisada a plataforma escolhida para o desenvolvimento, verificando características que auxiliem a conclusão dos objetivos definidos para este trabalho.

3.1 KIM - KNOWLEDGE & INFORMATION MANAGEMENT

A plataforma KIM¹ prove infraestrutura e serviços para automatização de tarefas como anotação, indexação e recuperação semântica de dados. Ela consiste de uma série de componentes baseados na linguagem Java e permite a integração de aplicações customizadas (KIRYAKOV et al., 2004).

Um dos focos desta plataforma é a anotação semântica automática, ou seja, sem intervenção humana. Para isto, são usadas tecnologias em cima de linguagem humana (HLT) e extração de informações (IE). Esta característica pode facilitar o processo de teste do projeto, pois evita o passo de anotação manual dos documentos, que poderia levar a erros nos resultados devido a falta de experiência e familiaridade com o processo.

Pode-se dividir a plataforma em três partes: base de conhecimento, KIM Server e *front-ends*. Para este projeto, as duas primeiras partes são as mais importantes. Uma visão geral da plataforma pode ser vista na figura 3.1.

3.1.1 BASE DE CONHECIMENTO

Na base de conhecimento são armazenados recursos com uso do **Sesame**², uma arquitetura para armazenamento de grandes quantidades de meta-dados, descritos em RDF e RDF Schema (BROEKSTRA; KAMPMAN; HARMELEN, 2002). O *design* e implementação deste componente independe do dispositivo de armazenamento, portanto pode ser usado em uma grande diversidade de configurações.

¹<http://ontotext.com/kim/>

²<http://www.openrdf.org/>

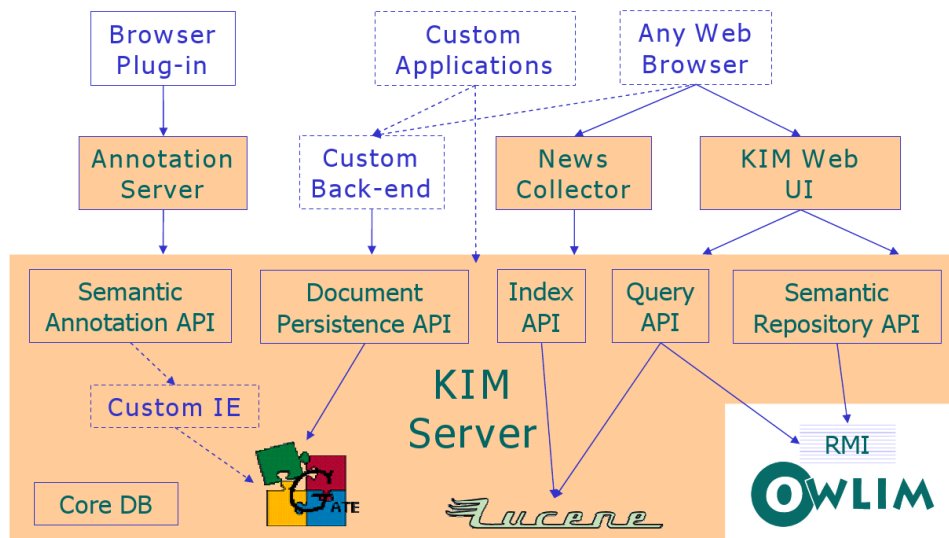


Figura 3: Arquitetura da plataforma KIM

Um dos recursos armazenados na base é a **KIMO** (KIM Ontology), uma ontologia de alto nível com cerca de 250 classes e 100 tipos de relações. A ontologia foi dividida em duas partes: **PROTON**³, uma ontologia de alto nível genérica, e os módulos específicos da plataforma KIM, **KIMSO** e **KIMLO**.

A PROTON faz algumas distinções filosóficas (e.g. diferenciando acontecimentos de objetos), e então descreve mais a fundo algumas entidades de importância geral (encontros, conflitos militares, governos e organizações, etc.). A intenção é que a ontologia seja utilizável para anotações de propósito geral, e também seja fácil de estender para domínios específicos. Uma base da ontologia pode ser vista na Figura 4.

A descrição de instâncias também é armazenada, junto com a ontologia. A plataforma vem com vários locais, organizações e pessoas pré-populados, bem como alguns relacionamentos entre essas instâncias. Uma instância pode também ser chamada de **entidade nomeada** (NE). A plataforma em geral é voltada para as entidades. O argumento para isso é que as pessoas tendem a pesquisar por coisas específicas (e.g. nomes de pessoas, lugares) e não por conceitos gerais. Esse foco tem um melhor resultado quando o conjunto de dados é sobre coisas específicas, como notícias. Em outros casos, como documentos sobre conceitos gerais (e.g. um artigo sobre "telefone"), o foco não é benéfico, pois muitos termos do documento não são conhecidos pela base.

³<http://proton.semanticweb.org/> - PROTo ONtology



Figura 4: Nível *top* da ontologia PROTON

3.1.2 KIM SERVER

KIM Server fornece uma série de APIs⁴ para anotação, indexação e recuperação de documentos. Internamente usa os *frameworks* **GATE** e **Lucene**, além do Sesame, mencionado anteriormente. Para que não haja dependência destes *frameworks* em uma aplicação customizada, são oferecidas APIs simplificadas para cada tarefa.

O *framework* GATE⁵ é usado para extração de informações, encontrando entidades nomeadas em um texto. Caso sejam identificadas entidades que não estão presentes na base de conhecimento, elas são armazenadas e gerenciadas separadamente.

O *framework* Lucene⁶ é usado para indexação e recuperação de informações. Ele foi adaptado para executar suas tarefas também em relação às entidades nomeadas, ao invés de fazer uma análise puramente sintática.

⁴Application Programming Interface, ou Interface de Programação de Aplicativos

⁵<http://www.gate.ac.uk/> - General Architecture for Text Engineering

⁶<http://lucene.apache.org/>

3.1.3 APIS DO KIM

Nesta seção serão descritas as partes mais interessantes do KIM para este trabalho. Uma visão geral de cada pacote será dada, mostrando os objetivos principais do mesmo. Algumas classes também serão mostradas, com uma pequena explicação e/ou demonstração dos principais métodos.

A estruturas de pacotes do KIM possui a seguinte estrutura:

```

com.ontotext.kim
|- client
| |- coredb
| | |- query
| |- corpora
| |- documentrepository
| |- entity
| |- inline
| |- lucene
| |- model
| |- query
| |- semanticannotation
| |- semanticrepository
| |- statistics
| |- ontology
|- util

```

Os pacotes mais importantes para este trabalho são:

corpora - Fornece os componentes básicos para o sistema, como a *KIMAnnotation* e o *KIM-Document*.

documentrepository - Fornece uma API (*DocumentRepositoryAPI*) para acesso ao repositório de documentos.

entity - Fornece uma API (*EntityAPI*) para acesso às entidades nomeadas.

CORPORA

O pacote *com.ontotext.kim.corpora* fornece os componentes básicos do sistema, como o *KIMAnnotation* e o *KIMDocument*. Por este motivo, é um dos pacotes mais usados em todo o sistema.

As *KIMAnnotations*, como o nome sugere, representam as anotações de documentos. A classe fornece métodos de comparação com outras anotações (*compareTo*), que podem ser usadas pelo próprio *Java* para ordenar listas ou construir conjuntos. Uma *KIMAnnotation* é composta basicamente de um número identificador e um mapa de Features (*FeatureMap*), que guarda as informações relativas aos documentos e entidades. As Features permitidas estão definidas na classe *FeaturesConstants*, porém a documentação não ajuda na compreensão das mesmas.

Os *KIMDocuments* representam os documentos no sistema. São compostos de um número de identificação (caso estejam armazenados em um banco de dados), um conjunto de anotações, e meta-dados sobre o documento.

DOCUMENT REPOSITORY

O pacote *documentrepository* é responsável pelo gerenciamento de documentos. A API fornece métodos para criar, recuperar e sincronizar documentos. Para carregar um documento específico por exemplo, podemos usar o método *loadDocument*, que recebe como parâmetro o número de identificação do documento.

O pacote também fornece a classe *DocumentQuery* para realizar pesquisa por documentos. Esta classe fornece a capacidade de limitar uma busca por palavras-chave, por um intervalo de tempo e por um número máximo de resultados. Outra possibilidade fornecida por *DocumentQuery* é restringir a pesquisa por documentos por entidades do repositório semântico. Esta capacidade foi utilizada no trabalho para efetuar as buscas.

ENTITY

O pacote *entity* define a *EntityDescription*, que representa uma entidade no código. Essa representação contém todos os atributos da entidade, e também suas relações.

A *EntityAPI* pode ser usada para recuperar uma *EntityDescription* específica, o que possibilita descobrir as relações da mesma. Com estas relações, podemos alimentar o grafo de tópicos com novos dados.

ACESSANDO AS APIS

A classe `KIMService` fornece métodos para acessar as APIs do servidor. O acesso ao servidor é feito através do protocolo RMI ⁷.

```
import com.ontotext.kim.client.GetService;
import com.ontotext.kim.client.KIMService;

public class Conectar {
    static KIMService serviceKim;

    public static void main(String[] args) {
        try {
            serviceKim = GetService.from();
        } catch (Exception e) {
            System.out.println("Nao foi possivel conectar
                ao servidor.");
        }
    }
}
```

Algoritmo 3.1: Acesso à API do KIM.

Após obter o acesso ao servidor, pode-se obter os componentes desejados

```
CorporaAPI apiCorpora = serviceKim.getCorporaAPI();
DocumentRepositoryAPI apiDR = serviceKim.
    getDocumentRepositoryAPI();
SemanticRepositoryAPI apiSemanticRepository = serviceKim.
    getSemanticRepositoryAPI();
```

Algoritmo 3.2: Acesso a componentes específicos.

⁷<http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp> - Remote Method Invocation

4 *PROTÓTIPO*

Como prova de conceito, foi desenvolvido um protótipo da aplicação. Neste capítulo é descrito o processo seguido, e alguns resultados obtidos com esta fase do projeto. A fase de prototipação é uma parte importante do processo, pois resultados rápidos são obtidos, fornecendo informações que não estavam disponíveis no início do projeto. Segundo Hunt e Thomas (2000), a fase de prototipação é uma experiência de aprendizado. O valor do protótipo não está no código produzido, mas sim nas lições aprendidas.

O protótipo consiste de duas camadas distintas. A primeira consiste de algumas classes gerais de uso comum para diversos componentes do sistema e interfaces, provendo a base da implementação. Esta parte é responsável principalmente pelo gerenciamento dos contextos, independentemente do sistema de busca. A segunda camada implementa uma interface gráfica e a conexão com o sistema de busca (neste caso, o KIM).

4.1 FLUXO DA BUSCA

Para começarmos com a implementação de um protótipo é preciso definir um fluxo para a busca, possibilitando assim definir uma estrutura geral para o programa. O fluxo definido para o processo de busca é o da figura 5. O primeiro passo no fluxo é a entrada da palavra-chave por parte do usuário. Em posse da palavra, é realizada uma busca no mapa de tópicos, fornecendo assim duas possibilidades: caso seja encontrada, a pesquisa por documentos é executada considerando o histórico de buscas do usuário; caso não seja encontrada, a palavra é pesquisada na ontologia. Caso mais de um tópico seja encontrado, é necessário escolher qual termo deve-se pesquisar efetivamente.

Nesta fase de desambiguação o *feedback* do usuário é recebido, sendo utilizado para a atualização do grafo de tópicos. Caso a palavra pesquisada tenha sido encontrada no grafo, sua relevância é reforçada, e caso não tenha sido encontrada, um novo tópico é inserido no grafo.

Sabendo qual termo da ontologia pesquisar (pelo histórico ou pela desambiguação), pode-

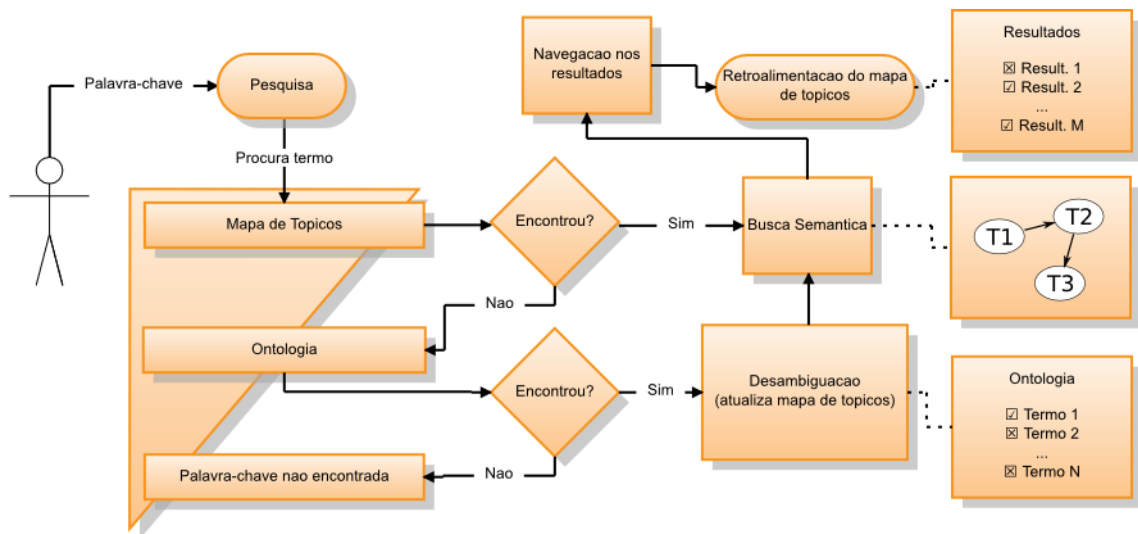


Figura 5: Fluxo da busca

se recuperar os documentos relevantes para aquela busca. Esta recuperação pode ser também expandida, utilizando os tópicos relacionados no mapa ao termo sendo pesquisado. Com os documentos em mão, o usuário tem outra chance de dar *feedback* ao sistema, escolhendo quais documentos são mais relevantes para ele na busca. Com isto, pode-se criar novas relações no mapa, pois cada documento está anotado por vários termos na ontologia.

4.2 ARQUITETURA

A arquitetura do protótipo consiste de dois pacotes principais. O primeiro pacote é o "server", responsável pela interface com o serviço de busca semântica. Os objetivos específicos deste pacote envolvem a desambiguação de termos (segundo a ontologia), acesso a documentos e anotações.

O pacote "client" é responsável pela parte de contexto do programa. O nome "client" foi utilizado no sentido de que o contexto é um cliente da busca semântica, mas ele próprio pode ser um servidor. Dentre as funções deste pacote estão receber o *feedback* do usuário e aplicar as métricas adequadas para classificar os resultados.

4.2.1 PACOTE CLIENT - GERENCIAMENTO DE CONTEXTOS

GERENCIAMENTO DE CONTEXTOS

O gerenciamento de contexto foi implementado com base nas estruturas fornecidas pela biblioteca Prefuse. Esta biblioteca fornece meios para criação e visualização de grafos e outras

estruturas. Sua escolha para o protótipo foi baseada na capacidade de resolver tanto a questão do gerenciamento e armazenamento dos dados quanto a visualização das informações.

Um grafo de tópicos foi criado para gerenciar os tópicos e relações. Uma função importante do mapa de tópico é normalizar os pesos de 0 até 1, atualizando os valores a cada iteração.

Para compor o grafo de tópicos, foram criadas as classes `Topic` e `Association`, baseadas em estruturas fornecidas pelo Prefuse. `Topic` encapsula a classe `prefuse.data.Node`, e `Association` encapsula `prefuse.data.Edge`. Com essas estruturas definidas, `TopicMap` possui um atributo `prefuse.data.Graph`, que utiliza os nodos e arestas para criação do grafo.

Uma *engine* foi criada para controlar algumas funções importantes do gerenciamento de contexto. Dentre essas funções estão o cálculo do peso das arestas, expansão das buscas segundo o contexto e a desambiguação do tópico pesquisado.

A parte de serialização e persistência de contextos não foi implementada, pois não há necessidade dessa funcionalidade para o protótipo. Devido a esta escolha, não é possível persistir e carregar um contexto para uma futura utilização do programa.

PREFUSE

Prefuse¹ é um *toolkit* para visualização de informação feito em Java. Esta ferramenta foi criada com o intuito de fornecer uma estrutura básica e customizável para que dados complexos possam ser visualizados facilmente. Para conseguir isto, o Prefuse provê abstrações que podem ser configuradas para cada domínio, permitindo que seja usada por uma grande variedade de softwares.

No âmbito deste trabalho, as informações mais interessantes para visualização são sobre o contexto do usuário. O fato de o grafo de tópicos ter a estrutura de um grafo valorado pode ser aproveitado pela visualização do Prefuse, pois o mesmo fornece as abstrações para nós e arestas.

O funcionamento das estruturas do Prefuse é baseada em tabelas. Os tópicos por exemplo são armazenados na tabela de vértices de um grafo. Cada linha desta tabela significa um tópico diferente, e cada coluna armazena um dado específico de um tópico. As associações são armazenadas na tabela de arestas do grafo. Com os dados populados nas tabelas, o Prefuse se encarrega de mostrar uma visualização gráfica dos dados.

A figura 6 mostra a estrutura básica do Prefuse. Uma fonte de dados (*Source Data*) é

¹<http://prefuse.org/>

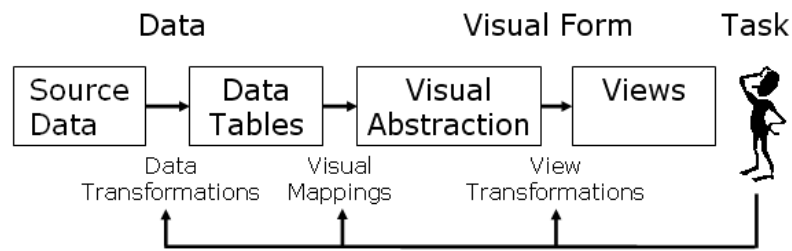


Figura 6: Estrutura da ferramenta

transformada para abstração de dados do Prefuse, as tabelas (*Data Table*). Estas tabelas, por sua vez, fornecem a base para os mapeamentos visuais (*Visual Abstraction*), onde são acrescentadas abstrações para posição, cor e geometria dos itens. O último passo é transformar a abstração visual em um gráfico na tela, o que é feito pelas *Views*.

4.2.2 PACOTE SERVER

SISTEMA DE BUSCAS

O sistema de buscas representa o nível mais baixo da arquitetura. Como mencionado no capítulo 3, o *framework* KIM foi utilizado, fornecendo todo o aparato para anotação e busca semântica.

O KIM oferece várias formas para se fazer uma busca. Para a prova de conceito, foi considerado adequado fazer a pesquisa por entidades na base de conhecimento e retornar os documentos associados à entidade. Este tipo de busca não é o mais sofisticado oferecido pelo KIM, porém a simplicidade de implementação no protótipo é um ponto chave. Outra questão considerada na implementação é o foco deste trabalho, que não está no sistemas de buscas em si, mas na parte de gerenciamento de contextos e sua integração com o sistema de buscas.

Duas classes realizam a integração da arquitetura projetada com o sistema de busca: *Definitions* e *Repository*. A classe *Repository* é responsável pelas buscas, já a classe *Definitions* é responsável pela desambiguação de termos.

A desambiguação de termos executada pela classe *Definitions* é o primeiro passo do processo de busca. Ela procura na base de conhecimento do KIM por entidades que contenham a palavra chave buscada, retornando um lista com todos os resultados possíveis. Estes resultados são representados pela classe *KBResult*, que possui uma *String* representando o rótulo principal da entidade, e uma *URI*² que representa a entidade unicamente na base. Com a lista retornada,

²Universal Resource Identification

o usuário do sistema pode ser perguntado sobre qual entidade ele quer pesquisar.

```

public List<KBResult> possibleDenotationsFor(String
keyword) {
    LinkedList<KBResult> l = new LinkedList<KBResult>();
    try {
        SemanticQuery semQuery = new SemanticQuery();
        semQuery.setReturnLabels(true);
        semQuery.addRequestedVar("Entity");
        semQuery.setClass("Entity", WKBConstants.
            CLASS_ENTITY);
        semQuery.addNameRestriction("Entity",
            SemanticQuery.COMPARE_STYLE_CONTAINS,
            keyword);
        SemanticQueryResult results = apiQuery.
            getEntities(semQuery);
        for (Object r : results) {
            SemanticQueryResultRow result = (
                SemanticQueryResultRow) r;
            l.add(new KBResult((URI) result.get(0),
                result.get(1).toString()))
                ;
        }
    } catch (KIMQueryException ex) {
        Logger.getLogger(Defs.class.getName()).log(Level.
            SEVERE, null, ex);
    }
    return l;
}

```

Algoritmo 4.1: Possíveis denotações.

O algoritmo consiste basicamente de criar uma `SemanticQuery`, que é fornecida pelo KIM, contendo uma restrição no nome da entidade. A restrição é que o nome contenha a palavra chave. Com os resultados da consulta realizada através da `SemanticQuery`, são criados `KBResults` para encapsular os dados, e então a lista é retornada.

A classe `Definitions` também é responsável por trazer os rótulos de uma determinada entidade. Por exemplo, a entidade que representa a cidade "Florianópolis" pode ter os rótulos "Floripa" e "Ilha da Magia".

```
public List<String> possibleLabelsFor(Resource entity) {
    List<String> l = new LinkedList<String>();
    EntityDescription b = null;
    try {
        b = apiEntity.getEntityDescription(entity);
    } catch (KIMQueryException ex) {
        ex.printStackTrace();
    }

    for (Literal label : b.getLabels()) {
        l.add(label.toString());
    }
    return l;
}
```

Algoritmo 4.2: Possíveis nomes.

O KIM prove uma função `getLabels()` que retorna os rótulos de uma entidade. Os rótulos então são transformados em *Strings* e retornados.

Depois da desambiguação dos termos, a classe `Repository` executa a segunda fase do processo de busca. Com a referência da entidade que o usuário deseja procurar, é possível recuperar os identificadores dos documentos anotados por essa entidade.

```
public List<Long> retrieve(Resource entity) {
    List<Long> l = new LinkedList<Long>();
    DocumentQuery q = new DocumentQuery();

    Set<String> entitiesRestriction = new HashSet<String>();
    entitiesRestriction.add(entity.toString());

    q.setCooccurringEntities(entitiesRestriction);
}
```

```

try {
    DocumentQueryResult documents = apiDocument.
        getDocuments(q);
    for (DocumentQueryResultRow doc : documents) {
        l.add(doc.getDocumentId());
    }
} catch (KIMQueryException ex) {
    ex.printStackTrace();
}
return l;
}

```

Algoritmo 4.3: Recuperação de documentos.

No KIM, todos os documentos possuem um número único de identificação. Este algoritmo busca todos os documentos associados a uma entidade da base de conhecimento, e então retorna uma lista com os identificadores dos documentos.

Para atualizar o contexto e para questões de visualização, é necessário obter as anotações dos documentos. Isto é alcançado com a função `getAnnotations()`, localizada na classe `Repository`, que recebe como argumento o identificador de um documento e retorna a lista de URIs das entidades relacionadas ao documento na base de conhecimento.

```

public List<URI> getAnnotations(long docID) {
    List<URI> l = new LinkedList<URI>();
    KIMDocument doc;
    KIMAnnotationSet annotations;
    try {
        doc = apiDocument.loadDocument(docID);
        annotations = doc.getAnnotations();

        Iterator<?> annIterator = annotations.iterator();

        while (annIterator.hasNext()) {
            KIMAnnotation kimAnnotation = (KIMAnnotation)
                annIterator.next();
            KIMFeatureMap kimFeatures = kimAnnotation.

```



```

        getFeatures();
        if (kimFeatures != null && filter(
            kimAnnotation.getType())) {
            String resourceUri = (String) kimFeatures
                .get(FeatureConstants.INSTANCE);
            l.add(new URIImpl(resourceUri));
        }
    }
} catch (KIMQueryException ex) {
    ex.printStackTrace();
} catch (KIMCorporaException ex) {
    ex.printStackTrace();
}
return l;
}

```

Algoritmo 4.4: Anotações de um documento.

Para recuperar as anotações, o primeiro passo é recuperar o documento a partir do identificador. A classe `KIMDocument` fornece a função `getAnnotations()`, que retorna um conjunto de entidades relacionadas ao documento. Para cada uma dessas entidades, adicionamos sua URI à lista.

4.3 RESULTADO

A figura 7 mostra o resultado do protótipo. A tela é dividida em três áreas principais: o grafo de tópicos, o campo de busca e os resultados.

A integração com a plataforma KIM foi feita com sucesso. Os documentos retornados e tópicos do grafo fazem parte da plataforma.

Protopto

File Edit

User's Map

Vocabulary

boeing

SEARCH

Results

Checked	Name	Annotated To	Brought By
<input type="checkbox"/>	http://docs/13:0.0681	[Newspaper_T.26, PublicCompa...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/39:0.0681	[Continent_T.4, Newspaper_T.2...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/49:0.0681	[Newspaper_T.26, PublicCompa...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/6:0.0614	[LocalCapital_T.763, CountryCa...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/31:0.054	[Airline_T.17, Newspaper_T.1, ...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/292:0.054	[Country_T.KS, PublicCompany...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/1:0.0141	[Newspaper_T.26, PublicCompa...	[[systemimpl.client.context.TopicPrefu...

Figura 7: Screenshot do protótipo.

5 EXPERIMENTOS

Não está no escopo do trabalho validar uma melhoria no sistema de busca. *Benchmarks* de sistemas de busca semântica ainda são um problema em aberto. Na busca sintática, para avaliar a melhoria de uma busca, basta verificar a presença dos termos pesquisados nos documentos retornados. Na busca semântica, a avaliação não é tão simples, pois um termo pode estar sendo usado em um documento com uma conotação diferente da pretendida na busca. Há também o uso de sinônimos, fazendo com que documentos relevantes para uma busca não sejam considerados como válidos.

A inserção de contextos na avaliação dos mecanismos de busca torna a mesma ainda mais complicada. A subjetividade torna inviável uma métrica automatizada, pois um resultado ruim para uma pessoa pode ser bom para outra. A utilização de roteiros para a pesquisa também não é satisfatória, pois podem refletir os hábitos de busca de uma pessoa, mas não de todas.

Os experimentos realizados neste capítulo tem a intenção de mostrar o funcionamento do protótipo, bem como mostrar alguns problemas encontrados durante o desenvolvimento do mesmo.

5.1 EXEMPLO DE INTERAÇÃO

Pela falta de um bom conjunto de documentos anotados é difícil simular uma interação arbitrária do usuário. Vários experimentos foram feitos, porém a relativa pequena quantidade de documentos não permite um experimento que se aproxime muito de uma interação real. Esta demonstração não busca verificar a precisão e revocação dos resultados nem a usabilidade do sistema em si, mas sim se a plataforma KIM foi capaz de cumprir os requisitos básicos do sistema, como a expansão e desambiguação dos resultados, e também se os resultados são classificados adequadamente de acordo com os pesos atribuídos no mapa de tópicos.

A figura 8 mostra uma primeira busca pela palavra "boeing". A pesquisa retornou 6 resultados, sendo os 3 primeiros documentos os seguintes:

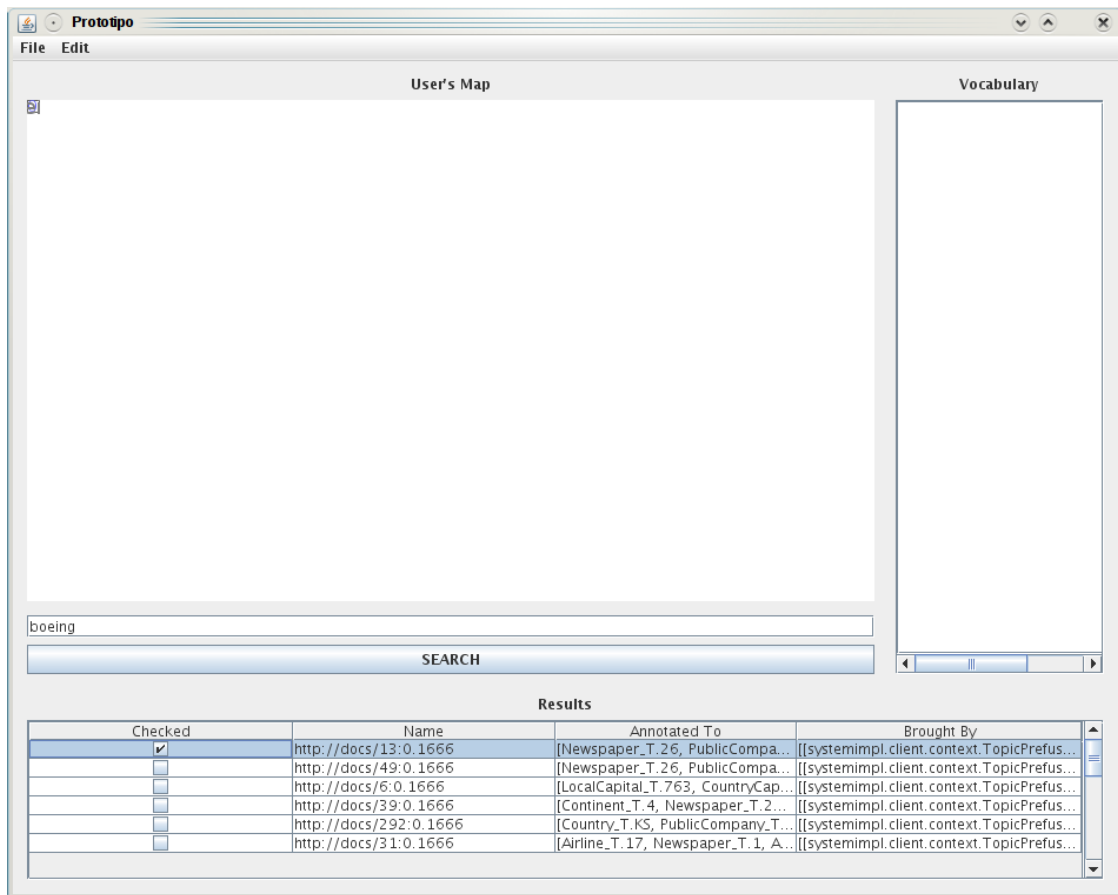


Figura 8: Primeira pesquisa por “boeing”

- *Low-cost airline puts the squeeze on Boeing as Lufthansa cuts routes* - notícia sobre vôos de baixo custo.
- *Budget airlines are proving popular with passengers and forcing traditional rivals to rethink their strategies* - notícia sobre vôos de baixo custo.
- *BP shows why it is no Shell* - notícia sobre as deferentes reações da empresa BP e da Shell durante uma crise de petróleo.

Marcando o documento 13 como relevante, o documento tem sua importância reforçada e a busca pode ser expandida com as anotações. A figura 9 mostra uma nova busca por “boeing”, mas desta vez a pesquisa retorna 78 documentos. Este aumento no número de documentos mostra que além dos documentos anotados com “boeing”, outros documentos com anotações semelhantes ao documento 13 foram recuperados. Estes outros documentos não tem uma relação direta com “boeing”, porém pelo histórico de buscas do usuário o sistema determina que eles podem ser relevantes.

A terceira busca é feita por “Europe”, e os 3 primeiros resultados retornados são os mes-

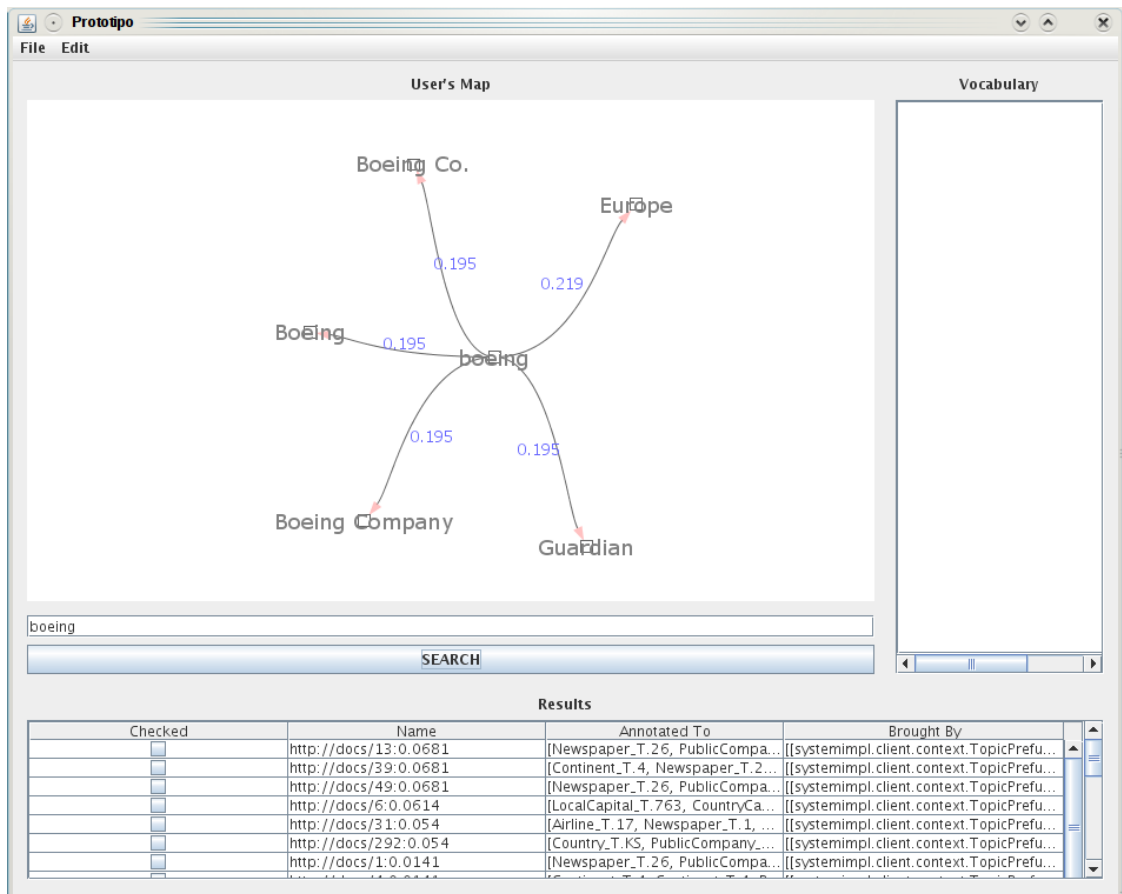


Figura 9: Segunda pesquisa por "boeing"

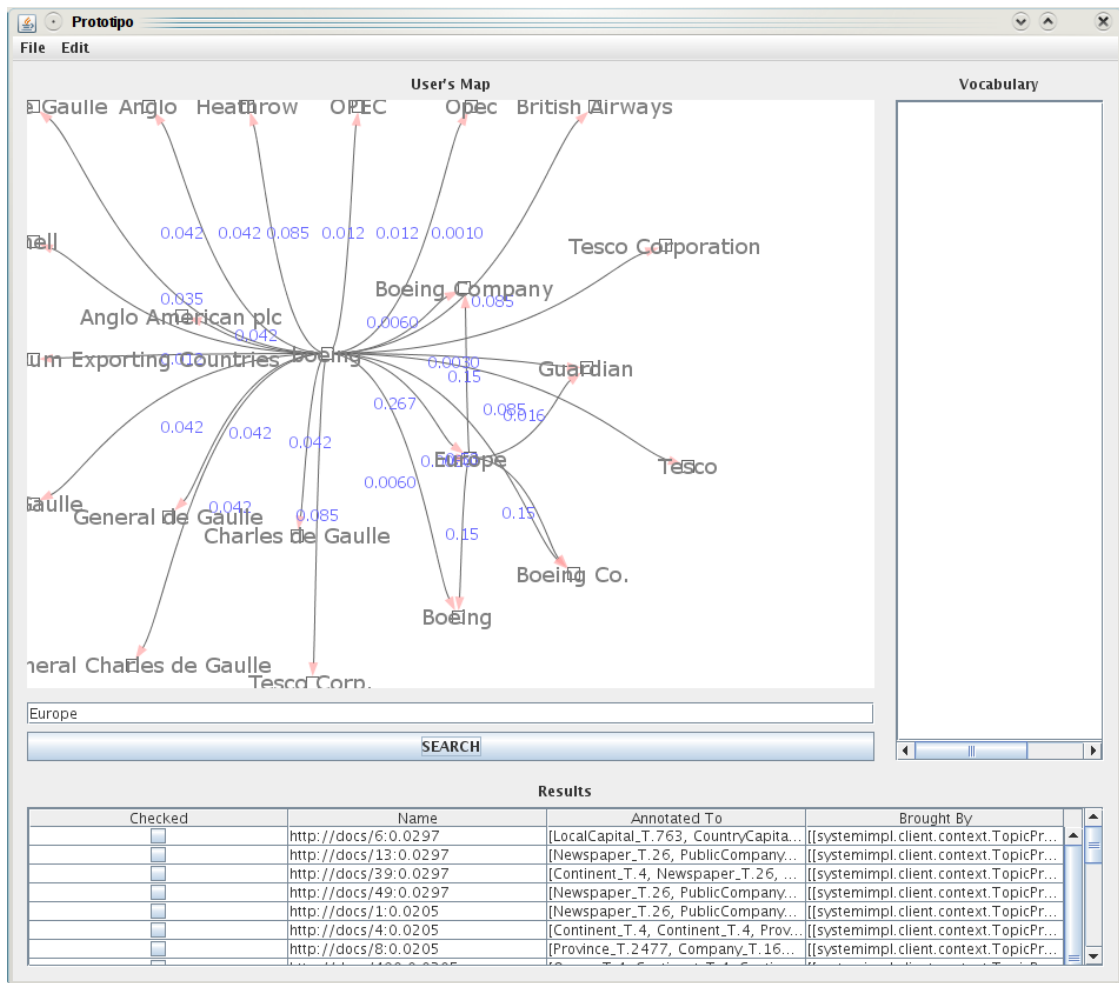


Figura 10: Pesquisa por “Europe” após pesquisa por “boeing”

mos das pesquisas anteriores. Os dois primeiros documentos retratam vôos de baixo custo na Europa, e ambos mencionam a empresa Boeing, fabricantes de aviões. A terceira notícia menciona uma companhia europeia de vôos que cancelou a compra de cinco Boeings. A relação entre “Europe” e “Boeing” fez crescer a relevância dos documentos onde ambas as anotações são encontradas.

Para efeitos de comparação, uma busca por “Europe” sem considerar as relações (ou seja, a primeira busca de uma interação), entre os três primeiros documentos, dois são diferentes:

- *Vodafone proves tech millstone* - notícia sobre uma baixa nas ações da Vodafone.
- *Square Mile slump hits Guardian iT* - notícia anuncia que a empresa *Guardian iT* receberia menos pedidos de clientes grandes nos próximos meses.

Outro ponto a se notar é que a pesquisa por “Europe” necessitou ser desambiguada quando executada sem contar com as relações. Quando feita depois de se pesquisar por “boeing” e

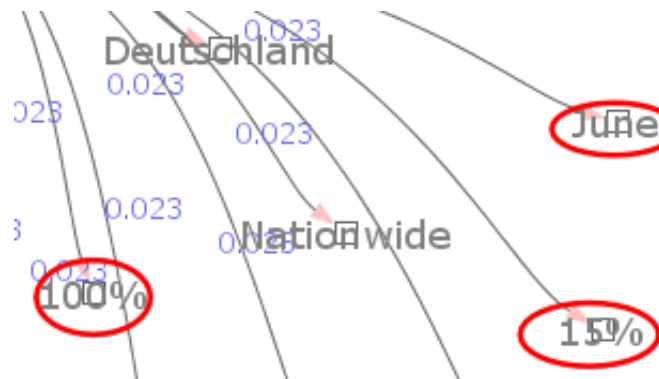


Figura 11: Exemplos de entidades desnecessárias: "100%", "15%", "June".

marcar documentos relevantes, a inferência sobre o tópico foi feita automaticamente.

5.2 PROBLEMAS ENCONTRADOS

Durante o desenvolvimento do protótipo, alguns problemas foram identificados. Nas próximas seções os principais deles são descritos.

5.2.1 ENTIDADES DESNECESSÁRIAS

Um dos primeiros problemas encontrados na utilização do KIM foram algumas anotações com entidades que não contribuem com a melhoria do processo de busca. Um documento sobre uma transação financeira, por exemplo, é anotado no KIM com os valores monetários, porcentagens envolvidas, e outras anotações que não fornecem auxílio concreto. Isso causa uma poluição do grafo de tópicos, que além de atrapalhar a visualização do mesmo, tem o potencial de causar lentidão no sistema.

Para amenizar este problema, foi implementado um filtro de entidades. Ao recuperar as anotações de um documento, as anotações que estão em categorias consideradas desnecessárias para o contexto são filtradas, deixando assim um contexto mais limpo e eficiente.

5.2.2 FOCO EM ENTIDADES NOMEADAS

O foco em entidades nomeadas do KIM parece ser uma abordagem interessante para documentos que não sejam conceituais. Notícias, por exemplo, são bastante adequadas a esse tipo de anotação, pois tratam normalmente de pessoas, lugares e acontecimentos específicos. Infelizmente, outros tipos de documentos podem não ser adequados para essa abordagem. Um artigo de uma enciclopédia sobre um conceito específico, por exemplo, normalmente não possui

muitas entidades nomeadas associadas a ele, o que faz com que a anotação do documento seja incompleta.

Este problema infelizmente não pode ser contornado sem mudanças drásticas na plataforma, o que nos leva ao ponto descrito na seção 5.2.3.

5.2.3 NECESSIDADE DE CUSTOMIZAÇÃO DO KIM

A utilização do KIM foi justificada pela utilização de uma ontologia de alto-nível, além da capacidade de anotar documentos automaticamente. Para os documentos que vem junto com a plataforma, ambos os fatores se mostraram satisfatórios, porém com documentos externos, a anotação foi um fracasso.

Para a utilização efetiva da plataforma, ajustes finos devem ser feitos. A ontologia, por ser generalista, fornece a possibilidade de ser estendida. Com isso, domínios mais específicos podem ser melhor atendidos pela plataforma.

Outra possibilidade de modificação é na extração de informação (entidades nomeadas presentes no repositório) utilizada pelo KIM durante a anotação semântica. Esta é uma solução mais complicada, e os problemas devem ser identificados primeiro, para verificar a viabilidade e eficácia da solução.

A documentação sobre customização da plataforma se motrou escassa e confusa. Como não faz parte do escopo deste trabalho, não foram investidos esforços nestas áreas.

6 CONCLUSÕES

Este trabalho apresentou uma resenha sobre sistemas de buscas semânticas, captura e uso de informação de contexto relativa a ontologias em tais sistemas e relatou experiências obtidas na implementação de extensões para coleta e utilização de informação de contexto sobre a plataforma KIM. O objetivo geral do trabalho foi atingido, integrando a plataforma KIM com um sistema de coleta e utilização de informações de contexto. O sistema é capaz de expandir e desambiguar as buscas, bem como classificar os resultados. Todavia, as APIs de integração do módulo de gerenciamento de contexto com os módulos de anotação e busca semântica do KIM ainda precisam ser refinadas.

A plataforma KIM se mostrou promissora para o suporte básico a buscas semânticas com informação de contexto, porém as suas características mais importantes para este trabalho necessitam de ajustes finos para um funcionamento satisfatório. Sua ontologia generalista pode ser interessante na anotação de notícias ou outros documentos deste gênero, mas há necessidade de estender a mesma para áreas específicas para possibilitar melhores anotações. O foco do KIM em entidades nomeadas também é um ponto importante, pois a anotação semântica depende da base de conhecimento estar populada com as entidades que aparecem nos documentos. Se uma entidade está em um documento, mas não está na base de conhecimento, a anotação do documento fica prejudicada.

A utilização básica do KIM se mostrou relativamente simples. Porém o uso de funcionalidades avançadas é dificultado pela documentação limitada e um tanto confusa. A documentação das classes é extremamente superficial, o que dificulta o entendimento da plataforma. Os exemplos de uso presentes na documentação também são superficiais, não mostrando a verdadeira capacidade da plataforma.

A maioria dos problemas encontrados durante o desenvolvimento tem alguma relação com a parte de anotação semântica. Isto sugere que a área precisa de mais pesquisas. Estas pesquisas são extremamente importantes, pois todos os outros passos necessários para atingir a Web semântica como foi imaginada precisam de boas anotações. Anotações manuais são inviáveis

devido ao volume de dados, portanto a automatização do processo é imprescindível.

Outra área que precisa de avanços é a área de avaliação de sistemas de busca semântica. Sem isso, as pesquisas para melhoria desses sistemas ficam limitadas, pois é difícil comparar os resultados e benefícios de diferentes abordagens, quando os resultados corretos para uma busca dependem da intenção do usuário quanto a denotação de palavras-chaves utilizadas nas buscas. Métricas de melhoria da qualidade dos resultados, como precisão e revocação são importantes para avaliar diferentes abordagens, mas difíceis de aferir na presença de ambiguidades. Faltam *benchmarks* que levem em consideração o significado específico de termos usados nas consultas e a sequência de interações com o usuário. Os benefícios do sistema de busca semântica com informação de contexto dependem desta sequência de interações fornecendo *feedback* para o sistema capturar a informação de contexto a ser utilizada no processamento de buscas subsequentes.

6.1 TRABALHOS FUTUROS

Os trabalhos futuros sugeridos visam corrigir os problemas encontrados durante o desenvolvimento deste trabalho. As linhas de trabalho sugeridas são:

- Extensão da ontologia PROTON, adaptando a mesma a domínios específicos.
- Estudo e possível customização do mecanismo de anotação do KIM. A plataforma provê alternativas para isso, que pode propiciar melhoria nos resultados.
- Pesquisa e desenvolvimento de métodos de avaliação de sistemas de busca semântica. Isto possibilitaria a comparação entre vários sistemas, e permitiria verificar a melhoria oferecida pelos mesmos.
- Integração do mecanismo de contexto com outras plataformas.

APÊNDICE A – ARTIGO

BUSCA DIRIGIDA POR ONTOLOGIAS E INFORMAÇÕES DE CONTEXTO

Renato Besen ³Departamento de Informática e Estatística Universidade Federal de Santa Catarina – Florianópolis, SC – Brazil renatob@inf.ufsc.br

Abstract. *Ontologies serve the need of machines to manipulate and formally describe knowledge shared among users. Human communication, however, depends deeply in subjective information, that can be collected and stored in a context related to each user. The objective of this work is to integrate a module that collects and uses context information with a platform that has semantic search capabilities, leveraging the usability of the system in addition to the increased precision and recall of the results.*

Resumo. *Ontologias atendem às necessidades das máquinas na descrição formal e manipulação de conhecimento compartilhado por conjuntos de usuários. A comunicação humana, no entanto, depende também de informações subjetivas, que podem ser coletadas e mantidas no contexto relativo a cada indivíduo. O objetivo deste trabalho é integrar um módulo de coleta e utilização de informações de contexto com uma plataforma de busca semântica, visando melhorar a usabilidade do sistema, além da precisão e cobertura dos resultados obtidos.*

INTRODUÇÃO

A grande maioria das bases de dados disponíveis fornecem um sistema de indexação baseado em metadados, como palavras-chave, título e autor, e também na contagem de ocorrências de palavras no texto. Porém, a simples indexação léxica do conteúdo de campos de metadados pode ser ineficaz em algumas situações.

Na Web Semântica, idealizada por Tim Berners-Lee [FRAUENFELDER 2004], os conteúdos são descritos semanticamente, possibilitando assim que suas descrições semânticas sejam pro-

cessadas por um computador. Uma forma de anotar os conteúdos é de acordo com metadados definidos de acordo com uma ontologia, que fornece uma definição formal de conceitos, instâncias e seus relacionamentos em um determinado domínio de interesse.

O formalismo de uma ontologia fornece interoperabilidade e permite que diferentes entidades se comuniquem. Em contrapartida, este formalismo demanda a disponibilidade de especialistas do domínio da ontologia para organizar o conhecimento nela descrito. Além disso, o usuário de um sistema de busca baseado em ontologia deve ter uma visão do domínio semelhante à visão da ontologia.

Este trabalho visa incorporar um mecanismo de coleta, representação e uso de informações de contexto do usuário a uma ferramenta de busca semântica, para contemplar a individualidade do usuário.

Este artigo está organizado da seguinte forma: na segunda seção alguns fundamentos são apresentados; na terceira seção alguns comentários sobre as ferramentas e o protótipo criado são apresentados; na quarta seção uma pequena demonstração é feita; na quinta seção são apresentadas as conclusões.

FUNDAMENTOS

Hoje na web os métodos de busca se baseiam na busca sintática de documentos. Por este motivo, há uma limitação na capacidade de recuperação de documentos, pois as buscas não levam em conta significados e relações dos termos pesquisados. Na Web Semântica os dados são anotados semanticamente, possibilitando assim que sistemas automatizados raciocinem sobre eles. Com isso um computador seria capaz de interpretar os dados, e não apenas apresentá-los na tela.

Uma forma de dar significado à web é criando ontologias, e então ligando os dados de páginas aos conceitos descritos nas mesmas [BERNERS-LEE; HENDLER; LASSILA, 2001]. Estas ligações entre dados e conceitos é chamada anotação semântica. A qualidade do processo de anotação tem impacto direto nos serviços oferecidos na Web Semântica.

Segundo Davies, Studer e Warren (2006) uma definição amplamente aceita de ontologia é “uma especificação explícita e formal da conceitualização de um domínio de interesse”. Esta definição toca em dois pontos importantes: por ser formal, máquinas de inferência podem ser usadas para raciocinar sobre a ontologia; e ontologias descrevem um certo domínio de interesse, desde domínios mais específicos até ontologias de alto nível, modelando grandes quantidades de conhecimento.

Hoje as ferramentas de busca se baseiam principalmente na contagem das palavras em um documento para determinar sua relevância. A busca semântica tenta tirar proveito das informações do domínio disponíveis. Quando é feita a busca por um conceito, podemos incluir documentos que não seriam retornados em uma busca comum, mas que estão relacionados semanticamente de alguma forma com os conceitos pesquisados. Caso uma busca ambígua seja feita, há a possibilidade de desambiguar ou separar os resultados por categorias.

O contexto pode ser definido como um conjunto de informações que auxilia na inferência de intenções de um usuário. Em relação à interação homem-máquina, o contexto é um conjunto de atributos que pode ser usado para tomada de decisões sem a necessidade de interromper a atividade do usuário [SATYANARAYANAN et al., 2001].

Grafos de tópicos servem para representar estruturas de conhecimento e associá-las a outros recursos. Estas representações permitem a construção de tecnologia para gerenciamento de informação. Grafos de tópicos podem ser comparados em função com glossários e tesouros, porém fornecem muito mais flexibilidade, pois permitem a capacidade de criar associações entre tópicos e recursos externos.

FERRAMENTAS E PROTÓTIPO

A plataforma KIM1 prove infraestrutura e serviços para automatização de tarefas como anotação, indexação e recuperação semântica de dados. Ela consiste de uma série de componentes baseados na linguagem Java e permite a integração de aplicações customizadas (KIRYAKOV et al., 2004).

Um dos focos desta plataforma é a anotação semântica automática, ou seja, sem intervenção humana. Para isto, são usadas tecnologias em cima de linguagem humana (HLT) e extração de informações (IE). Esta característica pode facilitar o processo de teste do projeto, pois evita o passo de anotação manual dos documentos, que poderia levar a erros nos resultados devido a falta de experiência e familiaridade com o processo.

Como prova de conceito, foi desenvolvido um protótipo da aplicação. O protótipo consiste de duas camadas distintas. A primeira consiste de algumas classes gerais de uso comum para diversos componentes do sistema e interfaces, provendo a base da implementação. Esta parte é responsável principalmente pelo gerenciamento dos contextos, independentemente do sistema de busca. A segunda camada implementa uma interface gráfica e a conexão com o sistema de busca (neste caso, o KIM).

A implementação do sistema é baseada no fluxo apresentado na figura 1.

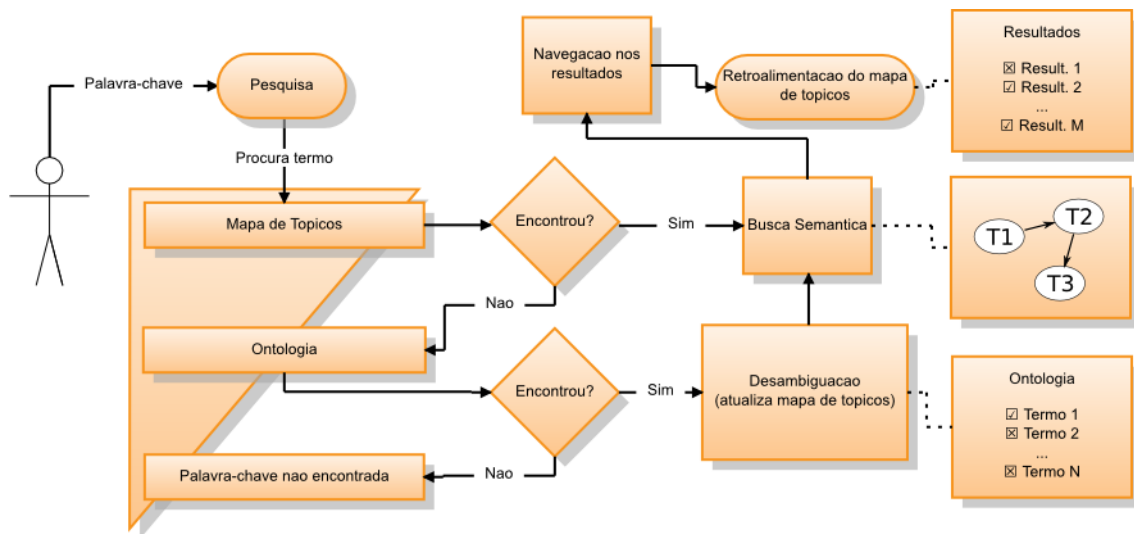


Figura 12: Fluxo da busca

O primeiro passo no fluxo é a entrada da palavra-chave por parte do usuário. Em posse da palavra, é realizada uma busca no mapa de tópicos, fornecendo assim duas possibilidades: caso seja encontrada, a pesquisa por documentos é executada considerando o histórico de buscas do usuário; caso não seja encontrada, a palavra é pesquisada na ontologia. Caso mais de um tópico seja encontrado, é necessário escolher qual termo deve-se pesquisar efetivamente.

Nesta fase de desambiguação o feedback do usuário é recebido, sendo utilizado para a atualização do grafo de tópicos. Caso a palavra pesquisada tenha sido encontrada no grafo, sua relevância é reforçada, e caso não tenha sido encontrada, um novo tópico é inserido no grafo.

Sabendo qual termo da ontologia pesquisar, pode-se recuperar os documentos relevantes para aquela busca. Esta recuperação pode ser também expandida, utilizando os tópicos relacionados no mapa ao termo sendo pesquisado. Com os documentos em mão, o usuário tem outra chance de dar feedback ao sistema, escolhendo quais documentos são mais relevantes para ele na busca. Com isto, pode-se criar novas relações no mapa, pois cada documento está anotado por vários termos na ontologia.

A figura 2 mostra o resultado do protótipo. A tela é dividida em três áreas principais: o grafo de tópicos, o campo de busca e os resultados.

EXPERIMENTO

Pela falta de um bom conjunto de documentos anotados é difícil simular uma interação arbitrária do usuário. Vários experimentos foram feitos, porém a relativa pequena quantidade de documentos não permite um experimento que se aproxime muito de uma interação real. Esta

Protopto

File Edit

User's Map

Vocabulary

```

graph TD
    boeing((boeing)) -- 0.195 --> BoeingCo[Boeing Co.]
    boeing -- 0.219 --> Europe[Europe]
    boeing -- 0.195 --> Boeing[Boeing]
    boeing -- 0.195 --> BoeingCompany[Boeing Company]
    boeing -- 0.195 --> Guardian[Guardian]
  
```

boeing

SEARCH

Results

Checked	Name	Annotated To	Brought By
<input type="checkbox"/>	http://docs/13:0.0681	[Newspaper_T.26, PublicCompa...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/39:0.0681	[Continent_T.4, Newspaper_T.2...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/49:0.0681	[Newspaper_T.26, PublicCompa...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/6:0.0614	[LocalCapital_T.763, CountryCa...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/31:0.054	[Airline_T.17, Newspaper_T.1, ...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/292:0.054	[Country_T.KS, PublicCompany...	[[systemimpl.client.context.TopicPrefu...
<input type="checkbox"/>	http://docs/1:0.0141	[Newspaper_T.26, PublicCompa...	[[systemimpl.client.context.TopicPrefu...

Figura 13: Screenshot do protótipo.

demonstração não busca verificar a precisão e revocação dos resultados nem a usabilidade do sistema em si, mas sim se a plataforma KIM foi capaz de cumprir os requisitos básicos do sistema.

Com uma primeira busca pela palavra "boeing", A pesquisa retornou 6 resultados, sendo os 3 primeiros documentos os seguintes:

- Low-cost airline puts the squeeze on Boeing as Lufthansa cuts routes - notícia sobre vôos de baixo custo.
- Budget airlines are proving popular with passengers and forcing traditional rivals to rethink their strategies - notícia sobre vôos de baixo custo.
- BP shows why it is no Shell - notícia sobre as deferentes reações da empresa BP e da Shell durante uma crise de petróleo.

Marcando o primeiro documento como relevante, o mesmo tem sua importância reforçada e a busca pode ser expandida com as anotações. Em uma nova busca por "boeing" a pesquisa retorna 78 documentos. Este aumento no número de documentos mostra que além dos documentos anotados com "boeing", outros documentos com anotações semelhantes ao primeiro documento foram recuperados. Estes outros documentos não tem uma relação direta com "boeing", porém pelo histórico de buscas do usuário o sistema determina que eles podem ser relevantes.

Em uma terceira busca, o termo "Europe" é pesquisado, e os 3 primeiros resultados retornados são os mesmos das pesquisas anteriores. A relação entre "Europe" e "Boeing" fez crescer a relevância dos documentos onde ambas as anotações são encontradas. Para efeitos de comparação, uma busca por "Europe" sem considerar as relações (ou seja, a primeira busca de uma interação), entre os três primeiros documentos dois são diferentes.

Um ponto a se notar é que a pesquisa por "Europe" necessitou ser desambiguada quando executada sem contar com as relações. Quando feita depois de se pesquisar por "boeing" e marcar documentos relevantes, a inferência sobre o tópico foi feita automaticamente.

CONCLUSÕES E TRABALHOS FUTUROS

A plataforma KIM se mostrou promissora para o suporte básico a buscas semânticas com informação de contexto, porém as suas características mais importantes para este trabalho necessitam de ajustes finos para um funcionamento satisfatório. Sua ontologia generalista pode ser

interessante na anotação de notícias ou outros documentos deste gênero, mas há necessidade de estender a mesma para áreas específicas para possibilitar melhores anotações. O foco do KIM em entidades nomeadas também é um ponto importante, pois a anotação semântica depende da base de conhecimento estar populada com as entidades que aparecem nos documentos. Se uma entidade está em um documento, mas não está na base de conhecimento, a anotação do documento fica prejudicada.

A maioria dos problemas encontrados durante o desenvolvimento tem alguma relação com a parte de anotação semântica. Isto sugere que a área precisa de mais pesquisas. Estas pesquisas são extremamente importantes, pois todos os outros passos necessários para atingir a Web semântica como foi imaginada precisam de boas anotações. Anotações manuais são inviáveis devido ao volume de dados, portanto a automatização do processo é imprescindível.

Os trabalhos futuros sugeridos visam corrigir os problemas encontrados durante o desenvolvimento deste trabalho. As linhas de trabalho sugeridas são:

- Extensão da ontologia PROTON, adaptando a mesma a domínios específicos.
- Estudo e possível customização do mecanismo de anotação do KIM. A plataforma provê alternativas para isso, que pode propiciar melhoria nos resultados.
- Pesquisa e desenvolvimento de métodos de avaliação de sistemas de busca semântica. Isto possibilitaria a comparação entre vários sistemas, e permitiria verificar a melhoria oferecida pelos mesmos.
- Integração do mecanismo de contexto com outras plataformas.

APÊNDICE B – CÓDIGO FONTE

B.1 PROJETO SYSTEM

B.1.1 SYSTEM.CLIENT.CONTEXT

```
package system.client.context;

public interface Association {
    public Topic getTargetTopic();
    public Topic getOriginTopic();
    public double getWeight();
}
```

Algoritmo B.1: Assosiation.java

```
package system.client.context;

public class NonExistingTopicException extends Exception
{
    private static final long serialVersionUID = 1L;
}
```

Algoritmo B.2: NonExistingTopicException.java

```
package system.client.context;

import org.openrdf.model.URI;

public interface Topic {
    public URI getOccurence();
}
```

```

    public double getWeight();
}

```

Algoritmo B.3: Topic.java

```

package system.client.context;

import java.util.HashSet;
import java.util.List;

import org.openrdf.model.URI;

public interface TopicGraphContext {
    public void cleanAssociations();
    public void cleanTopics();
    public Topic createOrRecoverTopic(String name, URI
        occurrence, double weight);
    public Association createOrRecoverAssociation(Topic
        topic1, Topic topic2);
    public List<Topic> listHomonyms(String name);
    public List<Association> listOutgoingAssociations(
        Topic topic1);
    public void addInteractionWeight(Topic topic1, double
        value);
    public void addInteractionWeight(Association
        association, double value);
    public void normalize();
    public void passTimeUnusedAssociations(HashSet<
        Association> usedAssociations);
    public void passTimeUnusedTopics(HashSet<Topic>
        usedTopics);
}

```

Algoritmo B.4: TopicGraphContext.java

B.1.2 SYSTEM.CLIENT.FORAGINGENGINE

```
package system.client.foragingEngine;

import java.rmi.RemoteException;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

import org.openrdf.model.URI;

import system.client.context.Association;
import system.client.context.Topic;
import system.client.context.TopicGraphContext;
import system.common.Result;
import system.server.definitions.Definitions;
import system.server.definitions.KBResult;
import system.server.repository.Repository;

public class Core {

    Definitions defs;
    TopicGraphContext context;
    private double incr;
    public static double PRECISION = 10000d;
    private Repository r;
    private double tThreshold = 0.0d;
    private double aThreshold = 0.0d;
    private double sThreshold = 0.05d;
    private static final int maxDepth = 2;
    GuiInterface gui;
```

```

public Core(final TopicGraphContext tm, final
    Repository rep,
        final Definitions d, final GuiInterface gui)
    {
        context = tm;
        r = rep;
        defs = d;
        incr = 1.0d;
        this.gui = gui;
    }

public TopicGraphContext getContext() {
    return context;
}

public List<Result> search(final String[] keywords) {

    final List<Expansion> expanded = new LinkedList<
        Expansion>();
    List<Result> results;
    final List<Expansion> searches = new LinkedList<
        Expansion>();
    for (final String keyword : keywords) {
        Topic t;
        if (context.listHomonyms(keyword).size() ==
            0) { // vai ser sempre
                // verdadeiro

                final List<KBResult> denotations =
                    disambiguateKeyword(keyword);
                final double weight = 0; // 1 /
                    denotations.size();
                for (final KBResult result : denotations)
                    {

```

```

        t = context.createOrRecoverTopic(
            keyword, result.getUri(),
            weight);

        context.createOrRecoverAssociation(t,
            t);

    }
}
searches.addAll(lookup(keyword, getTThreshold
    ()));

}

expanded.addAll(deepExpand(searches, maxDepth,
    getAThreshold()));

results = recover(expanded);
System.out.println(results);
Collections.sort(results);
return results;

}

List<Expansion> lookup(final String keyword, final
    Double tThreshold) {

    final List<Expansion> searches = new LinkedList<
        Expansion>();
    final List<Topic> homonyms = context.listHomonyms
        (keyword);
    double size = 0d;
    try {
        for (final Topic topic : homonyms) {
            if (topic.getWeight() >= tThreshold) {

```

```

        searches.add(new Expansion(topic));
    }
    size++;
}
} catch (final Exception e) {
    e.printStackTrace();
}
return searches;
}

List<Expansion> deepExpand(final List<Expansion>
searches,
    final Integer maxDepth, final Double
    aThreshold) {
    final List<Expansion> expanded = new LinkedList<
    Expansion>();
    List<Expansion> temp = new LinkedList<Expansion
    >();
    final List<Expansion> answer = new LinkedList<
    Expansion>();
    final HashMap<Topic, Expansion> visited = new
    HashMap<Topic, Expansion>();
    try {

        for (final Expansion s : searches) {
            temp.addAll(expand(s, aThreshold));
            expanded.addAll(temp);
            visited.put(s.getExpanded(), s);
        }

        Expansion s;

        Integer depth = 0;
        while (depth < maxDepth) {
            temp = new LinkedList<Expansion>();

```

```

        while (!expanded.isEmpty()) {
            s = expanded.remove(0);
            if (s.getWeight() > sThreshold) {
                if (visited.containsKey(s.
                    getExpanded())) {
                    s = compare(visited.get(s.
                        getExpanded()), s);
                }
                temp.addAll(expand(s, aThreshold)
                    );
                visited.put(s.getExpanded(), s);
            }
        }
        expanded.addAll(temp);

        depth++;
    }

} catch (final Exception e) {
    e.printStackTrace();
}

answer.addAll(visited.values());

return answer;
}

List<Expansion> expand(final Expansion s, final
    Double aThreshold) { // ,

    final List<Expansion> expanded = new LinkedList<
        Expansion>();

```



```
Expansion es = null;
Association a;
try {
    final Topic t = s.getDisambiguatedTopic();
    final Iterator<Association> associated =
        context
            .listOutgoingAssociations(t).iterator
                ();
    System.out.println("associated = " +
        associated);
    Double size = 0d;
    if (associated.hasNext()) {
        while (associated.hasNext()) {
            a = associated.next();
            final Topic topicDestiny = a.
                getTargetTopic();
            final double associationWeight = a.
                getWeight();
            if (associationWeight >= aThreshold)
            {
                es = new Expansion(s,
                    topicDestiny,
                    associationWeight);
                expanded.add(es);
            }
            size++;
        }
    } else {
        context.createOrRecoverAssociation(t, t);
        es = new Expansion(s, t, 1d);
        expanded.add(es);
    }
}
```

```
    } catch (final Exception e) {
        e.printStackTrace();
    }
    Collections.sort(expanded);
    return new LinkedList<Expansion>(expanded);
}

public List<Result> recover(final List<Expansion>
expanded) {
    double total = 0.d;
    final Hashtable<Long, Result> rt = new Hashtable<
    Long, Result>();
    for (final Expansion es : expanded) {
        final URI occurrence = es.getExpanded().
        getOccurrence();
        try {
            final List<Long> rl = r.retrieve(
            occurrence);
            for (final Long resource : rl) {
                Result res;
                if (rt.containsKey(resource)) {
                    res = rt.get(resource);
                } else {
                    res = new Result(resource);
                    rt.put(resource, res);
                }
                res.add(es, es.getWeight());
                total = total + es.getWeight();
                System.out.println(res + " " + es.
                getWeight());
            }
        } catch (final RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

```
        }
    }

    if (total == 0) {
        for (final Result result : rt.values()) {
            result.add(null, 1);
            result.normalize(rt.values().size());
        }
    } else {
        for (final Result result : rt.values()) {
            result.normalize(total);
        }
    }
    return new LinkedList<Result>(rt.values());
}

public void updateContext(final List<Result> feedback
    ,
        final List<Result> results) {

    final HashMap<URI, Integer> numberOfAnnotations =
        new HashMap<URI, Integer>();

    final HashSet<Topic> usedTopics = new HashSet<
        Topic>();
    final HashSet<Association> usedAssociations = new
        HashSet<Association>();

    final List<Result> rejected = new LinkedList<
        Result>();
    rejected.addAll(results);

    try {
```

```

for (final Result res : results) {
    for (final URI annot : r.getAnnotations(
        res.getDocId())) {
        if (numberOfAnnotations.containsKey(
            annot)) {
            numberOfAnnotations.put(annot,
                numberOfAnnotations
                    .get(annot) + 1);
        } else {
            numberOfAnnotations.put(annot, 1)
                ;
        }
    }
}
} catch (final RemoteException e) {
    e.printStackTrace();
}
for (final Result res : feedback) {

    Topic origin;
    Topic destiny;
    Topic trackTopicOrigin;
    Topic trackTopicDestiny;
    Association a;
    for (final Expansion es : res.
        getRelatedExpandedSearches()) {
        origin = es.getDisambiguatedTopic();

        context.addInteractionWeight(origin, incr
            );
        usedTopics.add(origin);
        final List<Topic> track = es.backTrack();
        trackTopicDestiny = track.get(0);
        double value;
        try {

```

```
for (final URI occurrence : r
     .getAnnotations(res.getDocId
                     ())) {

    final List<String> labels = defs
        .possibleLabelsFor(
            occurrence);
    value = incr /
        numberOfAnnotations.get(
            occurrence);

    for (int i = track.size() - 1; i
        > 1; i--) {
        trackTopicOrigin = track.get(
            i);
        trackTopicDestiny = track.get
            (i - 1);

        a = context.
            createOrRecoverAssociation
            (
                trackTopicOrigin,
                trackTopicDestiny)
            ;
        context.addInteractionWeight(
            a, value);
        usedAssociations.add(a);
    }

    a = context.
        createOrRecoverAssociation(
            trackTopicDestiny, track.
                get(0));
    usedAssociations.add(a);
```

```
        for (final String label : labels)
        {
            destiny = context.
                createOrRecoverTopic(label
                    ,
                        occurrence, 0);
            context.addInteractionWeight(
                destiny, value);
            a = context.
                createOrRecoverAssociation
                    (origin,
                        destiny);
            context.addInteractionWeight(
                a, value);
            usedAssociations.add(a);
        }
    }

    } catch (final RemoteException e) {
        e.printStackTrace();
    }
}

endInteration(usedTopics, usedAssociations);
}

public Repository getRepository() {
    return r;
}

public void setRepository(final Repository r) {
    this.r = r;
}

public Definitions getDefinitions() {
```

```
        return defs;
    }

    public void setDefinitions(final Definitions defs) {
        this.defs = defs;
    }

    public double getIncr() {
        return incr;
    }

    public void setIncr(final double incr) {
        this.incr = incr;
    }

    public Double getTThreshold() {
        return tThreshold;
    }

    public void setTThreshold(final Double tThreshold) {
        this.tThreshold = tThreshold;
    }

    public Double getAThreshold() {
        return aThreshold;
    }

    public void setAThreshold(final Double aThreshold) {
        this.aThreshold = aThreshold;
    }

    public Double getSThreshold() {
        return sThreshold;
    }
}
```

```
public void setSThreshold(final Double sThreshold) {
    this.sThreshold = sThreshold;
}

public List<KBResult> disambiguateKeyword(final
String keyword) {
    try {
        final List<KBResult> occurrences = defs
            .possibleDenotationsFor(keyword);
        return gui.disambiguateKeyword(occurrences,
            keyword);
    } catch (final Exception e) {
        e.printStackTrace();
        return null;
    }
}

public void endInteration(final HashSet<Topic>
usedTopics,
    final HashSet<Association> usedAssociations)
{
    context.passTimeUnUsedAssociations(
        usedAssociations);

    context.passTimeUnUsedTopics(usedTopics);

    context.normalize();
    context.cleanAssociations();
    context.cleanTopics();
}

public Expansion compare(final Expansion e1, final
Expansion e2) {
```



```

        if (e1.getWeight() > e2.getWeight()) {
            return e1;
        } else {
            return e2;
        }
    }

    public double weightPath(final double topicWeight,
        final double associationWeight) {
        return topicWeight * associationWeight;
    }
}

```

Algoritmo B.5: Core.java

```

package system.client.foragingEngine;

import java.util.LinkedList;
import java.util.List;

import system.client.context.Topic;

public class Expansion implements Comparable<Expansion> {

    private final Topic disambiguatedTopic;
    private final Topic expandedTopic;
    private Expansion previous = null;
    private Double pathWeight = 1d;

    public Expansion(final Topic topic) {
        disambiguatedTopic = topic;
        expandedTopic = topic;
    }

    public Expansion(final Expansion previous, final
        Topic expandedTopic,

```

```

        final double pathWeight) {
    this.previous = previous;
    this.expandedTopic = expandedTopic;
    disambiguatedTopic = previous.
        getDisambiguatedTopic();
    this.pathWeight = this.pathWeight * pathWeight;
}

public Topic getDisambiguatedTopic() {
    return disambiguatedTopic;
}

public Topic getExpanded() {
    return expandedTopic;
}

public double getWeight() {

    return pathWeight;
}

@Override
public boolean equals(final Object obj) {
    if (obj instanceof Expansion) return ((Expansion)
        obj).getExpanded()
            .equals(getExpanded());
    else throw new RuntimeException("Argumento
        invalido");
}

public List<Topic> backTrack() {
    final List<Topic> answer = new LinkedList<Topic
        >();
    Expansion prev = this;
    int i = 0;

```

```

        while (prev.previous != null) {

            i++;
            answer.add(prev.expandedTopic);
            prev = prev.previous;
        }
        answer.add(prev.expandedTopic);

        return answer;
    }

    @Override
    public int compareTo(final Expansion o) {
        return pathWeight.compareTo(o.getWeight());
    }

    @Override
    public int hashCode() {
        return (getExpanded()).hashCode();
    }

    @Override
    public String toString() {
        return backtrack().toString();
    }
}

```

Algoritmo B.6: Expansion.java

```

package system.client.foragingEngine;

import java.util.List;

import system.server.definitions.KBResult;

public interface GuiInterface {

```

```

    public List<KBResult> disambiguateKeyword(List<
        KBResult> options,
        String keyword);
}

```

Algoritmo B.7: GuiInterface.java

B.1.3 SYSTEM.CLIENT.FORAGINGENGINE.METRICS

```

package system.client.foragingEngine.metrics;

import system.client.foragingEngine.Expansion;

public interface Metric {

    public Expansion compare(Expansion s1, Expansion s2);

    public double weightPath(double topicWeight, double
        associationWeight);

}

```

Algoritmo B.8: Metric.java

```

package system.client.foragingEngine.metrics;

import system.client.foragingEngine.Expansion;

public class MetricA implements Metric {

    public Expansion compare(final Expansion s1, final
        Expansion s2) {
        if (s1.getWeight() > s2.getWeight()) {
            return s1;
        } else {

```

```

        return s2;
    }
}

public double weightPath(final double topicWeight,
    final double associationWeight) {
    return topicWeight * associationWeight;
}
}

```

Algoritmo B.9: MetricA.java

B.1.4 SYSTEM.COMMON

```

package system.common;

import java.util.LinkedList;
import java.util.List;

import org.openrdf.model.URI;
import org.openrdf.model.impl.URIImpl;

import system.client.foragingEngine.Core;
import system.client.foragingEngine.Expansion;

public class Result implements Comparable<Result> {

    private final URI content;
    private double unNormalizedAccumulatedRelevance;
    private double normalizedAccumulatedRelevance;
    private final List<Expansion> expansions;
    private Long docId;

    public Result(final URI content) {
        this.content = content;
        unNormalizedAccumulatedRelevance = 0d;
    }
}

```

```
        normalizedAccumulatedRelevance = 0d;
        expansions = new LinkedList<Expansion>();
    }

    public Result(final Long docId) {
        content = new URIImpl("http://docs/" + docId);
        this.docId = docId;
        unNormalizedAccumulatedRelevance = 0d;
        normalizedAccumulatedRelevance = 0d;
        expansions = new LinkedList<Expansion>();
    }

    public void add(final Expansion expansion, final
        double weight) {
        unNormalizedAccumulatedRelevance =
            unNormalizedAccumulatedRelevance
                + weight;
        if (expansion != null) {
            expansions.add(expansion);
        }
    }

    public double getNormalizedAccumulatedRelevance() {
        return normalizedAccumulatedRelevance;
    }

    public double normalize(final double total) {
        normalizedAccumulatedRelevance =
            unNormalizedAccumulatedRelevance
                / total;
        unNormalizedAccumulatedRelevance = 0d;
        return normalizedAccumulatedRelevance;
    }

    @Override
```

```
public String toString() {
    return content.toString() + ":"
        + (int) (
            getNormalizedAccumulatedRelevance() *
            Core.PRECISION)
        / Core.PRECISION;
}

public List<Expansion> getRelatedExpandedSearches() {
    return expansions;
}

public URI getContent() {
    return content;
}

@Override
public boolean equals(final Object obj) {
    if (obj == null || !(obj instanceof Result))
        return false;
    return content.equals(((Result) obj).getContent());
}

@Override
public int hashCode() {
    return content.hashCode();
}

@Override
public int compareTo(final Result o) {
    return Double.compare(
        normalizedAccumulatedRelevance, (o)
            .getNormalizedAccumulatedRelevance());
}
```

```

    }

    public Long getDocId() {
        return docId;
    }
}

```

Algoritmo B.10: Result.java

B.1.5 SYSTEM.SERVER.DEFINITIONS

```

package system.server.definitions;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import org.openrdf.model.Resource;

public interface Definitions extends Remote {

    public List<KBResult> possibleDenotationsFor(String
        term)
        throws RemoteException;

    public List<String> possibleLabelsFor(Resource entity
        )
        throws RemoteException;

}

```

Algoritmo B.11: Definitions.java

```

package system.server.definitions;

import org.openrdf.model.URI;

```



```
public class KBResult {
    private final URI uri;
    private final String name;

    public KBResult(final URI uri, final String name) {
        this.uri = uri;
        this.name = name;
    }

    public URI getUri() {
        return uri;
    }

    public String getName() {
        return name;
    }
}
```

Algoritmo B.12: KBResult.java

B.1.6 SYSTEM.SERVER.REPOSITORY

```
package system.server.repository;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
import java.util.Set;

import org.openrdf.model.Resource;
import org.openrdf.model.URI;

public interface Repository extends Remote {
```

```

public List<URI> getAnnotations(long docId) throws
    RemoteException;

public List<Long> retrieve(Resource entity) throws
    RemoteException;

public Set<String> getLabels() throws RemoteException
    ;
}

```

Algoritmo B.13: Repository.java

B.2 PROJETO SYSTEMIMPL

B.2.1 SYSTEMIMPL

```

package systemimpl;

public class CONSTANTS {
    public static String LABEL = "label";
    public static String OCCURRENCE = "occurrence";
    public static String TOPICWEIGHT = "topicWeight";
    public static String ASSOCIATIONWEIGHT = "
        associationWeight";
    public static String HISTORY = "history";
}

```

Algoritmo B.14: CONSTANTS.java

```

package systemimpl;

import systemimpl.client.gui.Gui;

public class Launcher {
    public static void main(final String[] args) {
        java.awt.EventQueue.invokeLater(new Runnable() {

```

```

        public void run() {

            final Gui gui = new Gui();
            gui.setVisible(true);
            gui.start();
        }
    });
}
}

```

Algoritmo B.15: Launcher.java

B.2.2 SYSTEMIMPL.CLIENT.CONTEXT

```

package systemimpl.client.context;

import prefuse.data.Edge;
import system.client.context.Association;
import system.client.context.Topic;
import systemimpl.CONSTANTS;

class AssociationPrefuse implements Association {

    Edge edge;

    public AssociationPrefuse(final Edge edge) {
        this.edge = edge;
    }

    public Topic getTargetTopic() {
        return new TopicPrefuse(edge.getTargetNode());
    }

    public Topic getOriginTopic() {
        return new TopicPrefuse(edge.getSourceNode());
    }
}

```

```

public double getWeight() {
    return (Double) edge.get(CONSTANTS.
        ASSOCIATIONWEIGHT);
}

public Edge getEdge() {
    return edge;
}

@Override
public boolean equals(final Object arg0) {
    if (arg0 == null) return false;
    if (!(arg0 instanceof AssociationPrefuse)) return
        false;
    return edge.equals(((AssociationPrefuse) arg0).
        edge);
}

@Override
public int hashCode() {
    return edge.hashCode();
}
}

```

Algoritmo B.16: AssociationPrefuse.java

```

package systemimpl.client.context;

import static systemimpl.CONSTANTS.HISTORY;
import static systemimpl.CONSTANTS.TOPICWEIGHT;

import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedList;

```

```

import java.util.List;

import org.openrdf.model.URI;

import prefuse.data.Edge;
import prefuse.data.Graph;
import prefuse.data.Node;
import prefuse.data.Table;
import prefuse.data.expression.Predicate;
import prefuse.data.expression.parser.ExpressionParser;
import prefuse.util.collections.IntIterator;
import system.client.context.Association;
import system.client.context.Topic;
import system.client.context.TopicGraphContext;
import systemimpl.CONSTANTS;

public class PrefuseTopicGraphContext implements
    TopicGraphContext {

    private static final double DECAY = 0.8;
    private static final double PRECISION = 1000;
    public Double DEFAULT\_TOPIC\_WEIGHT = 0.0;
    public Double DEFAULT\_ASSOCIATION\_WEIGHT = 0.0;
    public Double DEFAULT\_PASS\_TIME\_TOPIC\_WEIGHT =
        0.0;

    /** Minimum weight to remove topic threshold. */
    private final Double MWTRT\_THRESHOLD = -1.0d;

    /** Minimum weight to remove association threshold.
     */
    private final Double MWTRA\_THRESHOLD = 0.0d;

    /** Weighted topic graph. */
    protected Graph graph;

```

```

/** Table of topics. */
Table ttpcs;

/** Table of associations. */
Table tassocs;

public PrefuseTopicGraphContext() {
    graph = new Graph(true);
    ttpcs = graph.getNodeTable();
    ttpcs.addColumn(CONSTANTS.LABEL, String.class);
    ttpcs.addColumn(CONSTANTS.OCCURRENCE, String.
        class);
    ttpcs.addColumn(CONSTANTS.TOPICWEIGHT, Double.
        class);
    ttpcs.addColumn(CONSTANTS.HISTORY, Double.class,
        0d);
    tassocs = graph.getEdgeTable();
    tassocs.addColumn(CONSTANTS.ASSOCIATIONWEIGHT,
        Double.class);
    tassocs.addColumn(CONSTANTS.HISTORY, Double.class
        , 0d);
}

public Topic hasTopic(final String name, final URI
    occurrence) {

    if (name != null) {
        final IntIterator rows = search(CONSTANTS.
            LABEL + " = '" + name
                + "' AND " + CONSTANTS.OCCURRENCE + "
                    = '"
                + occurrence.toString() + "'", ttpcs)
            ;
        if (rows.hasNext()) {

```

```

        return new TopicPrefuse(graph.getNode(
            rows.nextInt()));
    } else {
        return null;
    }
} else {
    final IntIterator rows = search(CONSTANTS.
        OCCURRENCE + " = '"
            + occurrence.toString() + "'", ttpcs)
        ;
    if (rows.hasNext()) {
        return new TopicPrefuse(graph.getNode(
            rows.nextInt()));
    } else {
        return null;
    }
}

}

public Topic createOrRecoverTopic(final String name,
    final URI occurrence,
        final double weight) {
    Node answer;
    final IntIterator rows = search(CONSTANTS.LABEL +
        " = '" + name
            + "' AND " + CONSTANTS.OCCURRENCE + " = '"
            + occurrence.toString() + "'", ttpcs);

    if (rows.hasNext()) {
        answer = graph.getNode(rows.nextInt());
    } else {
        answer = graph.addNode();
        answer.set(CONSTANTS.LABEL, name);
    }
}

```

```

        answer.set(CONSTANTS.OCCURRENCE, occurrence.
            toString());
        answer.set(CONSTANTS.TOPICWEIGHT, weight);
        answer.set(CONSTANTS.HISTORY, new Double(0d))
            ;
    }
    return new TopicPrefuse(answer);
}

public Association createOrRecoverAssociation(final
    Topic topic1,
        final Topic topic2) {
    final Node node1 = ((TopicPrefuse) topic1).
        getNode();
    final Node node2 = ((TopicPrefuse) topic2).
        getNode();
    Edge answer = graph.getEdge(((TopicPrefuse)
        topic1).getNode(),
        ((TopicPrefuse) topic2).getNode());
    if (answer != null) {
        return new AssociationPrefuse(answer);
    } else {
        answer = graph.addEdge(node1, node2);
        answer.set(CONSTANTS.ASSOCIATIONWEIGHT,
            DEFAULT\_TOPIC\_WEIGHT);
        answer.set(CONSTANTS.HISTORY, new Double(0d))
            ;
        return new AssociationPrefuse(answer);
    }
}

public URI getOccurrence(final Topic topic1) {
    return topic1.getOccurrence();
}

```



```

public double getWeight(final Topic topic1) {
    return topic1.getWeight();
}

public List<Topic> listHomonyms(final String name) {
    final List<Topic> answer = new LinkedList<Topic>();

    final IntIterator rows = search(CONSTANTS.LABEL +
        " = '" + name + "'",
        ttpcs);
    while (rows.hasNext()) {
        answer.add(new TopicPrefuse(graph.getNode(
            rows.nextInt())));
    }
    return answer;
}

IntIterator search(final String query, final Table
    table) {
    final Predicate myPredicate = (Predicate)
        ExpressionParser.parse(query);
    return table.rows(myPredicate);
}

public void cleanAssociations() {
    final IntIterator rows = search(CONSTANTS.
        ASSOCIATIONWEIGHT + " <= "
        + MWTRA\_THRESHOLD, tassocs);
    while (rows.hasNext()) {
        final Edge e = graph.getEdge(rows.nextInt());
        graph.removeEdge(e);
    }
}

```

```

public void cleanTopics() {
    final IntIterator rows = search(CONSTANTS.
        TOPICWEIGHT + " <= "
            + MWTRT\_THRESHOLD, ttpcs);
    while (rows.hasNext()) {
        final Node n = graph.getNode(rows.nextInt());
        graph.removeNode(n);
    }
}

public List<Association> listOutgoingAssociations(
    final Topic topic1) {
    final List<Association> answer = new LinkedList<
        Association>();
    final Iterator<Edge> i = graph.outEdges(((
        TopicPrefuse) topic1)
        .getNode());
    while (i.hasNext()) {
        answer.add(new AssociationPrefuse(i.next()));
    }
    return answer;
}

public void addInteractionWeight(final Topic topic,
    final double value) {
    final Node node = ((TopicPrefuse) topic).getNode
        ();
    node.set(CONSTANTS.HISTORY, (Double) node.get(
        CONSTANTS.HISTORY)
        + value);
}

public void addInteractionWeight(final Association
    association,

```

```

        final double value) {
    final Edge edge = ((AssociationPrefuse)
        association).getEdge();
    edge.set(CONSTANTS.HISTORY, (Double) edge.get(
        CONSTANTS.HISTORY)
        + value);
}

public double getInteractionWeight(final Topic topic1
) {
    final Node n = ((TopicPrefuse) topic1).getNode();
    return (Double) n.get(CONSTANTS.HISTORY);
}

public void resetInteraction(final Topic topic1) {
    final Node n = ((TopicPrefuse) topic1).getNode();
    n.set(CONSTANTS.HISTORY, 0d);
}

public double getInteractionWeight(final Association
    association1) {
    final Edge e = ((AssociationPrefuse) association1
        ).getEdge();
    return (Double) e.get(CONSTANTS.HISTORY);
}

public void resetInteraction(final Association
    association1) {
    final Edge e = ((AssociationPrefuse) association1
        ).getEdge();
    e.set(CONSTANTS.HISTORY, 0d);
}

public void normalize2() {
    final HashSet<String> labels = new HashSet<String

```

```

>());

final String query = CONSTANTS.LABEL + " != null"
;
final IntIterator i = search(query, ttpcs);
while (i.hasNext()) {
    labels
        .add((String) graph.getNode(i.nextInt
            ()).get(
                CONSTANTS.LABEL));
}

final Iterator<String> ilabels = labels.iterator
();

while (ilabels.hasNext()) {
    final String label = ilabels.next();
    final List<Topic> homonymous = listHomonyms(
        label);

    double currentTotal = 0d;
    for (final Topic topic : homonymous) {

        final Node node = ((TopicPrefuse) topic).
            getNode();
        currentTotal = currentTotal
            + (Double) node.get(CONSTANTS.
                HISTORY);
    }

    double current = 0d;
    double old = 0d;
    double temp = 0d;
    for (final Topic topic : homonymous) {

```

```

        final Node node = ((TopicPrefuse) topic).
            getNode();
        current = (Double) node.get(CONSTANTS.
            HISTORY) / currentTotal;
        old = (Double) node.get(CONSTANTS.
            TOPICWEIGHT);
        temp = (old * DECAy + current);
        node.set(CONSTANTS.TOPICWEIGHT, temp);
        final List<Association> outgoing =
            listOutgoingAssociations(topic);
        for (final Association association :
            outgoing) {
            final Edge a = ((AssociationPrefuse)
                association).getEdge();
            old = (Double) a.get(CONSTANTS.
                ASSOCIATIONWEIGHT);
            current = (Double) a.get(CONSTANTS.
                HISTORY);
            temp = (old * DECAy + current);
            a.set(CONSTANTS.HISTORY, temp);
        }
    }
}

public void normalize() {
    Double totalCurrent;
    final HashSet<String> hs = new HashSet<String>();
    int row;
    final String query = CONSTANTS.LABEL + " != null"
        ;
    final Predicate myPredicate = (Predicate)
        ExpressionParser.parse(query);
    final IntIterator tableIterator = ttpcs.rows(
        myPredicate);

```

```

while (tableIterator.hasNext()) {
    row = tableIterator.nextInt();
    hs.add(ttpcs.getString(row, CONSTANTS.LABEL))
        ;
}

final Iterator<String> si = hs.iterator();
String s;
Node n;
Double old;
Double now;
Double temp;
while (si.hasNext()) {
    s = si.next();

    totalCurrent = 0d;
    final List<Topic> homonyms = listHomonyms(s);
    for (final Topic topic : homonyms) {
        n = ((TopicPrefuse) topic).getNode();
        old = (Double) n.get(CONSTANTS.
            TOPICWEIGHT);
        now = (Double) n.get(CONSTANTS.HISTORY);
        temp = (old * DECAy + now);
        n.set(CONSTANTS.HISTORY, temp);
        totalCurrent = totalCurrent + temp;
    }

    for (final Topic topic : homonyms) {
        n = ((TopicPrefuse) topic).getNode();
        if (totalCurrent == 0) {
            n.set(TOPICWEIGHT, 1d);
        } else {
            n
                .set(
                    TOPICWEIGHT,

```

```

                (((Double) n.get(
                    HISTORY) /
                    totalCurrent) *
                    PRECISION)
                / PRECISION);
    }
    this.resetInteraction(topic);

    totalCurrent = 0d;

    final List<Association> outgoing =
        listOutgoingAssociations(topic);

    Edge a;
    for (final Association association :
        outgoing) {
        a = ((AssociationPrefuse) association
            ).getEdge();
        old = (Double) a.get(CONSTANTS.
            ASSOCIATIONWEIGHT);
        now = (Double) a.get(CONSTANTS.
            HISTORY);
        temp = (old * DECAy + now);
        a.set(CONSTANTS.HISTORY, temp);
        totalCurrent = totalCurrent + temp;
    }

    for (final Association association :
        outgoing) {
        a = ((AssociationPrefuse) association
            ).getEdge();
        if (totalCurrent == 0) {
            a.set(CONSTANTS.ASSOCIATIONWEIGHT
                , 1d);
        } else {

```

```

        a
            .set(
                CONSTANTS.
                    ASSOCIATIONWEIGHT
                ,
                ((int) (((Double)
                    a
                        .get(
                            CONSTANTS
                                .
                                    HISTORY
                                ) /
                                    totalCurrent
                                ) *
                                    PRECISION
                                ) /
                                    PRECISION
                            ));
            }
        this.resetInteraction(association);
    }
}
}

public void passTimeUnusedAssociations(
    final HashSet<Association> usedAssociations)
    {
        final Iterator<Edge> edges = graph.edges();
        Association association;

        while (edges.hasNext()) {
            association = new AssociationPrefuse(edges.
                next());
            if (!usedAssociations.contains(association))

```



```

        {
            this.addInteractionWeight(association,
                DEFAULT\_PASS\_TIME\_TOPIC\_
                _WEIGHT);
        }
    }
}

public void passTimeUnUsedTopics(final HashSet<Topic>
    usedTopics) {
    final Iterator<Node> nodes = graph.nodes();
    Topic topic;
    while (nodes.hasNext()) {
        topic = new TopicPrefuse(nodes.next());
        if (!usedTopics.contains(topic)) {
            this
                .addInteractionWeight(topic,
                    DEFAULT\_PASS\_TIME\_
                    _TOPIC\_WEIGHT);
        }
    }
}

public Graph getGraph() {
    return graph;
}
}

```

Algoritmo B.17: PrefuseTopicGraphContext.java

```

package systemimpl.client.context;

import org.openrdf.model.URI;
import org.openrdf.model.impl.URIImpl;

```

```
import prefuse.data.Node;
import system.client.context.Topic;
import systemimpl.CONSTANTS;

public class TopicPrefuse implements Topic {

    private final Node node;

    TopicPrefuse(final Node node) {
        this.node = node;
    }

    public URI getOccurence() {
        return new URIImpl((String) node.get(CONSTANTS.
            OCCURRENCE));
    }

    public double getWeight() {
        return (Double) node.get(CONSTANTS.TOPICWEIGHT);
    }

    public Node getNode() {
        return node;
    }

    @Override
    public boolean equals(final Object arg0) {
        if (arg0 == null) return false;
        if (!(arg0 instanceof TopicPrefuse)) return false
            ;
        return node.equals(((TopicPrefuse) arg0).node);
    }

    @Override
    public int hashCode() {
```

```

        return node.hashCode();
    }

}

```

Algoritmo B.18: TopicPrefuse.java

B.2.3 SYSTEMIMPL.CLIENT.GUI

```

package systemimpl.client.gui;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.util.LinkedList;
import java.util.List;

import javax.swing.JCheckBox;

import system.server.definitions.KBResult;

public class DisambDialog extends javax.swing.JDialog {

    private static final long serialVersionUID = 1L;
    private List<KBResult> selectedOptions;
    private final List<KBResult> options;

    /** Creates new form DisambiguationDialog */
    public DisambDialog(final java.awt.Frame parent,
        final boolean modal,
        final List<KBResult> options, final String
            keyword) {
        super(parent, modal);
        initComponents(options, keyword);
        this.options = options;
    }
}

```

```
/**
 * This method is called from within the constructor
 * to initialize the form.
 * WARNING: Do NOT modify this code. The content of
 * this method is always
 * regenerated by the Form Editor.
 */
private void initComponents(final List<KBResult>
    options,
        final String keyword) {

    new javax.swing.JScrollBar();
    jLabel1 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    jPanel1 = new javax.swing.JPanel();
    boxes = new JCheckBox[options.size()];

    jPanel1.setLayout(new GridLayout(options.size(),
        1));
    setDefaultCloseOperation(javax.swing.
        WindowConstants.DISPOSE_ON_CLOSE);
    getContentPane().setLayout(new BorderLayout());

    jLabel1.setText("Chose the intended meaning(s)
        for the keyword \""
        + keyword + "\"");
    getContentPane().add(jLabel1, BorderLayout.PAGE\
        _START);

    jButton1.setText("OK");
    jButton1.addActionListener(new java.awt.event.
        ActionListener() {
        public void actionPerformed(final java.awt.
            event.ActionEvent evt) {
```

```

        jButton1ActionPerformed(evt);
    }
});
getContentPane().add(jButton1);
getContentPane().add(jButton1, BorderLayout.PAGE\
    _END);

int i = 0;
for (final KBRResult result : options) {
    boxes[i] = new JCheckBox();
    boxes[i].setText(result.getName());
    jPanel1.add(boxes[i]);
    i++;
}
jScrollPane1.setViewportViewView(jPanel1);
getContentPane().add(jScrollPane1);
pack();
} // </editor-fold>

private void jButton1ActionPerformed(final java.awt.
    event.ActionEvent evt) {
    int i = 0;
    selectedOptions = new LinkedList<KBRResult>();
    for (final JCheckBox c : boxes) {
        if (c.isSelected()) {
            selectedOptions.add(options.get(i));
        }
        i++;
    }
    dispose();
}

public List<KBRResult> getSelectedOptions() {
    return selectedOptions;
}

```

```

    }

    /**
     * @param args
     *         the command line arguments
     */

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private JCheckBox [] boxes;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration

}

```

Algoritmo B.19: DisambDialog.java

```

package systemimpl.client.gui;

import static javax.swing.JFileChooser.APPROVE_OPTION;

import java.rmi.RemoteException;
import java.util.LinkedList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.table.AbstractTableModel;

import org.openrdf.model.URI;

```

```

import prefuse.Visualization;
import system.client.foragingEngine.Core;
import system.client.foragingEngine.GuiInterface;
import system.common.Result;
import system.server.definitions.Definitions;
import system.server.definitions.KBResult;
import system.server.repository.Repository;
import systemimpl.client.context.PrefuseTopicGraphContext
    ;
import systemimpl.server.Defs;
import systemimpl.server.Rep;

public class Gui extends JFrame implements GuiInterface {

    private static final long serialVersionUID = 1L;
    Thread t;
    Core e;
    Object [][] dataResults;
    LinkedList<Result> selected;
    List<Result> returned;
    Repository r;
    Definitions d;

    /** Creates new form Gui */
    public Gui() {
        setTitle("Prototipo");
        setVisible(true);
        try {
            e = new Core(new PrefuseTopicGraphContext(),
                new Rep(), new Defs(),
                this);
        } catch (final RemoteException e) {
            e.printStackTrace();
        }
        selected = new LinkedList<Result>();
    }
}

```

```

        this.repaint();

    }

    private void initComponents() {

        System.out.println("comeco");
        display = new PaineL(((PrefuseTopicGraphContext)
            e.getContext())
            .getGraph());
        jMenuBar1 = new javax.swing.JMenuBar();
        menuFile = new javax.swing.JMenu();
        menuEdit = new javax.swing.JMenu();
        menuSearch = new javax.swing.JMenu();
        itemMTR = new javax.swing.JMenuItem();
        itemMAR = new javax.swing.JMenuItem();
        itemMSR = new javax.swing.JMenuItem();
        menuContext = new javax.swing.JMenu();
        itemMTW = new javax.swing.JMenuItem();
        itemMAW = new javax.swing.JMenuItem();
        itemHistory = new javax.swing.JMenuItem();
        menuFile.setText("File");
        jMenuBar1.add(menuFile);
        menuEdit.setText("Edit");
        menuSearch.setText("Search");
        itemMTR.setText("minimum topic relevance");
        itemMTR.addMouseListener(new java.awt.event.
            MouseAdapter() {

            @Override
            public void mouseReleased(final java.awt.
                event.MouseEvent evt) {
                itemMTRMouseReleased(evt);
            }
        });
    }
}

```



```
menuSearch.add(itemMTR);
itemMAR.setText("minimum association relevance");
itemMAR.addMouseListener(new java.awt.event.
    MouseAdapter() {

    @Override
    public void mouseReleased(final java.awt.
        event.MouseEvent evt) {
        itemMARMouseReleased(evt);
    }
});
menuSearch.add(itemMAR);
itemMSR.setText("minimum search relevance");
itemMSR.addMouseListener(new java.awt.event.
    MouseAdapter() {

    @Override
    public void mouseReleased(final java.awt.
        event.MouseEvent evt) {
        itemMSRMouseReleased(evt);
    }
});
menuSearch.add(itemMSR);
menuEdit.add(menuSearch);
menuContext.setText("Context");
itemMTW.setText("minimum topic weight");
itemMTW.addMouseListener(new java.awt.event.
    MouseAdapter() {

    @Override
    public void mouseReleased(final java.awt.
        event.MouseEvent evt) {
        itemMTWMouseReleased(evt);
    }
});
```

```
menuContext.add(itemMTW);
itemMAW.setText("minimum association weight");
itemMAW.addMouseListener(new java.awt.event.
    MouseAdapter() {

    @Override
    public void mouseReleased(final java.awt.
        event.MouseEvent evt) {
        itemMAWMouseReleased(evt);
    }
});
menuContext.add(itemMAW);
itemHistory.setText("history length");
itemHistory.addMouseListener(new java.awt.event.
    MouseAdapter() {

    @Override
    public void mouseReleased(final java.awt.
        event.MouseEvent evt) {
        itemHistoryMouseReleased(evt);
    }
});
menuContext.add(itemHistory);
menuEdit.add(menuContext);
jMenuBar1.add(menuEdit);
setJMenuBar(jMenuBar1);
caixaDeBusca = new javax.swing.JTextField();
caixaDeBusca.setText("sao paulo");
botaoPesquisar = new javax.swing.JButton();
jScrollPane3 = new javax.swing.JScrollPane();
jScrollPane2 = new javax.swing.JScrollPane();
listaVocabulario = new javax.swing.JList();
jScrollPane4 = new javax.swing.JScrollPane();
jScrollPane1 = new javax.swing.JScrollPane();
resultados = new javax.swing.JTable();
```

```
labelMapa = new javax.swing.JLabel();
nomeVocabulario = new javax.swing.JLabel();
labelResultados = new javax.swing.JLabel();
itemMenuNovo = new javax.swing.JMenuItem();
itemMenuCarregar = new javax.swing.JMenuItem();
itemMenuGravar = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.
    WindowConstants.EXIT_ON_CLOSE);
final javax.swing.GroupLayout painelLayout = new
    javax.swing.GroupLayout(
        display);
display.setLayout(painelLayout);
painelLayout.setHorizontalGroup(painelLayout.
    createParallelGroup(
        javax.swing.GroupLayout.Alignment.LEADING).
    addGap(0, 399,
        Short.MAX_VALUE));
painelLayout.setVerticalGroup(painelLayout.
    createParallelGroup(
        javax.swing.GroupLayout.Alignment.LEADING).
    addGap(0, 349,
        Short.MAX_VALUE));

botaoPesquisar.setText("SEARCH");
botaoPesquisar.addMouseListener(new java.awt.
    event.MouseAdapter() {

    @Override
    public void mouseReleased(final java.awt.
        event.MouseEvent evt) {
        botaoPesquisarMouseReleased(evt);
    }
});
```

```
listaVocabulario.setModel(new javax.swing.  
    AbstractListModel() {  
  
        private static final long serialVersionUID =  
            1L;  
        String[] strings = {};  
  
        public int getSize() {  
            return strings.length;  
        }  
  
        public Object getElementAt(final int i) {  
            return strings[i];  
        }  
    });  
jScrollPane2.setViewportView(listaVocabulario);  
  
jScrollPane3.setViewportView(jScrollPane2);  
  
resultados.setModel(new Model(0));  
  
jScrollPane1.setViewportView(resultados);  
  
jScrollPane4.setViewportView(jScrollPane1);  
  
labelMapa.setHorizontalAlignment(javax.swing.  
    SwingConstants.CENTER);  
labelMapa.setText("User's Map");  
  
nomeVocabulario  
    .setHorizontalAlignment(javax.swing.  
        SwingConstants.CENTER);  
nomeVocabulario.setText("Vocabulary");  
  
labelResultados
```

```
        .setHorizontalAlignment(javax.swing.
            SwingConstants.CENTER);
labelResultados.setText("Results");

menuFile.setText("File");

itemMenuNovo.setText("New map");
itemMenuNovo.addActionListener(new java.awt.event
    .ActionListener() {

        public void actionPerformed(final java.awt.
            event.ActionEvent evt) {
            itemMenuNovoActionPerformed(evt);
        }
    });
menuFile.add(itemMenuNovo);

itemMenuCarregar.setText("Load map");
itemMenuCarregar.addActionListener(new java.awt.
    event.ActionListener() {

        public void actionPerformed(final java.awt.
            event.ActionEvent evt) {
            itemMenuCarregarActionPerformed(evt);
        }
    });
menuFile.add(itemMenuCarregar);

itemMenuGravar.setText("Save map");
itemMenuGravar.addActionListener(new java.awt.
    event.ActionListener() {

        public void actionPerformed(final java.awt.
            event.ActionEvent evt) {
            itemMenuGravarActionPerformed(evt);
```

```
    }
  });
  menuFile.add(itemMenuGravar);

  final javax.swing.GroupLayout layout = new javax.
    swing.GroupLayout(
      getContentPane());
  getContentPane().setLayout(layout);
  layout.setHorizontalGroup(layout.
    createParallelGroup(
      javax.swing.GroupLayout.Alignment.LEADING).
    addGroup(
      layout.createSequentialGroup().
        addContainerGap().addGroup(
          layout.createParallelGroup(
            javax.swing.GroupLayout.Alignment.
              LEADING).addComponent(
                jScrollPane4, javax.swing.GroupLayout.
                  DEFAULT\_SIZE, 594,
                Short.MAX\_VALUE).addGroup(
                  javax.swing.GroupLayout.Alignment.
                    TRAILING,
                  layout.createSequentialGroup().
                    addGroup(
                      layout.createParallelGroup(
                        javax.swing.GroupLayout.
                          Alignment.TRAILING)
                        .addComponent(labelMapa,
                          javax.swing.GroupLayout.
                            Alignment.LEADING,
                          javax.swing.GroupLayout.
                            DEFAULT\_SIZE, 399,
                          Short.MAX\_VALUE).
                          addComponent(display,
                            javax.swing.GroupLayout.
```

```

        Alignment.LEADING,
        javax.swing.GroupLayout.
        DEFAULT_SIZE,
        javax.swing.GroupLayout.
        DEFAULT_SIZE,
        Short.MAX_VALUE).
        addComponent(
        botaoPesquisar,
        javax.swing.GroupLayout.
        Alignment.LEADING,
        javax.swing.GroupLayout.
        DEFAULT_SIZE, 399,
        Short.MAX_VALUE).
        addComponent(
        caixaDeBusca,
        javax.swing.GroupLayout.
        Alignment.LEADING,
        javax.swing.GroupLayout.
        DEFAULT_SIZE, 399,
        Short.MAX_VALUE)).addGap
        (18, 18, 18).addGroup(
layout.createParallelGroup(
        javax.swing.GroupLayout.
        Alignment.LEADING, false)
        .addComponent(nomeVocabulario
        ,
        javax.swing.GroupLayout.
        DEFAULT_SIZE,
        javax.swing.GroupLayout.
        DEFAULT_SIZE,
        Short.MAX_VALUE).
        addComponent(
        jScrollPane3,
        javax.swing.GroupLayout.
        DEFAULT_SIZE, 177,

```

```

Short.MAX\_VALUE)))
        addComponent(
labelResultados, javax.swing.
        GroupLayout.DEFAULT\_SIZE, 594,
Short.MAX\_VALUE)).addContainerGap())
        );
layout
        .setVerticalGroup(layout
        .createParallelGroup(javax.swing.
        GroupLayout.Alignment.LEADING)
        .addGroup(
        layout
        .createSequentialGroup())
        .addGap(12, 12, 12)
        .addGroup(
        layout.createParallelGroup(
        javax.swing.GroupLayout.
        Alignment.BASELINE)
        .addComponent(labelMapa).
        addComponent(
        nomeVocabulario))
        .addPreferredGap(
        javax.swing.LayoutStyle.
        ComponentPlacement.RELATED
        )
        .addGroup(
        layout
        .createParallelGroup(
        javax.swing.
        GroupLayout.
        Alignment.TRAILING
        )
        .addGroup(
        layout
        .

```



```
        createSequentialGroup  
        (  
        .addComponent(  
            display,  
            javax.swing.  
                GroupLayout  
                .DEFAULT\  
                _SIZE,  
            javax.swing.  
                GroupLayout  
                .DEFAULT\  
                _SIZE,  
            Short.MAX\  
                _VALUE)  
        .addPreferredGap(  
            javax.swing.  
                LayoutStyle  
                .  
                ComponentPlacement  
                .UNRELATED  
        )  
        .addComponent(  
            caixaDeBusca,  
            javax.swing.  
                GroupLayout  
                .PREFERRED  
                \_SIZE,  
            javax.swing.  
                GroupLayout  
                .DEFAULT\  
                _SIZE,  
            javax.swing.  
                GroupLayout  
                .PREFERRED  
                \_SIZE)  
        )  
    }
```

```

        .addPreferredGap(
            javax.swing.
                LayoutStyle
                    .
                        ComponentPlacement
                            .RELATED)
        .addComponent(
            botaoPesquisar
        ))
    .addComponent(
        jScrollPane3,
        javax.swing.
            GroupLayout.
                DEFAULT\_SIZE,
                409,
                Short.MAX\_VALUE)).
        addGap(18, 18, 18)
    .addComponent(labelResultados).
        addPreferredGap(
            javax.swing.LayoutStyle.
                ComponentPlacement.RELATED
        )
    .addComponent(jScrollPane4,
        javax.swing.GroupLayout.
            PREFERRED\_SIZE, 136,
        javax.swing.GroupLayout.
            PREFERRED\_SIZE)
    .addContainerGap()));

pack();

System.out.println("fim");
} // </editor-fold> // GEN-END: initComponents

private void itemMTRMouseReleased(final java.awt.
    event.MouseEvent evt) {

```

```
try {
    e.setTThreshold(Double.parseDouble(
        JOptionPane
            .showInputDialog("Type a numeric value in
                [0,1]"))));
} catch (final Exception e) {
    JOptionPane.showMessageDialog(null, "Invalid
        input");
}

}

private void itemMARMouseReleased(final java.awt.
    event.MouseEvent evt) {
    try {
        e.setAThreshold(Double.parseDouble(
            JOptionPane
                .showInputDialog("Type a numeric value in
                    [0,1]"))));
    } catch (final Exception e) {
        JOptionPane.showMessageDialog(null, "Invalid
            input");
    }
}

private void itemMSRMouseReleased(final java.awt.
    event.MouseEvent evt) {
    try {
        e.setSThreshold(Double.parseDouble(
            JOptionPane
                .showInputDialog("Type a numeric value in
                    [0,1]"))));
    } catch (final Exception e) {
        JOptionPane.showMessageDialog(null, "Invalid
```

```
        input");
    }
}

private void itemMTWMouseReleased(final java.awt.
    event.MouseEvent evt) {
    try {
        // e.getContext().setMWTRT\_THRESHOLD(Double.
        parseDouble(JOptionPane.showInputDialog("
        Type a numeric value in [0,1]")));
    } catch (final Exception e) {
        JOptionPane.showMessageDialog(null, "Invalid
        input");
    }
}

private void itemMAWMouseReleased(final java.awt.
    event.MouseEvent evt) {
    try {
        // e.getContext().setMWTRA\_THRESHOLD(Double.
        parseDouble(JOptionPane.showInputDialog("
        Type a numeric value in [0,1]")));
    } catch (final Exception e) {
        JOptionPane.showMessageDialog(null, "Invalid
        input");
    }
}

private void itemHistoryMouseReleased(final java.awt.
    event.MouseEvent evt) {
    JOptionPane.showMessageDialog(null, "Ops!");
    // JOptionPane.showInputDialog("Type a numeric
    value in [0,1]");
}
}
```

```

private void botaoPesquisarMouseReleased(final java.
    awt.event.MouseEvent evt) {// GEN-FIRST:event\
    _botaoPesquisarMouseReleased
        display.setSwitch(false);
        final Model tm = (Model) resultados.getModel();
        final List<Result> res = new LinkedList<Result>()
            ;
        for (int i = 0; i < resultados.getRowCount(); i
            ++) {
            if ((resultados.getValueAt(i, 0) != null)
                && ((Boolean) resultados.getValueAt
                    (i, 0))) {
                selected.add((Result) resultados.
                    getValueAt(i, 1));
            }
            res.add((Result) resultados.getValueAt(i, 1))
                ;
        }

        e.updateContext(selected, res);
        display.setSwitch(true);

        selected = new LinkedList<Result>();
        final String keyword = caixaDeBusca.getText();

        try {
            returned = e.search(keyword.split(", "));

            dataResults = new Object[returned.size()][4];
            Result rs;

            int j = returned.size() - 1;
            for (int i = 0; i < returned.size(); i++) {
                rs = returned.get(i);
                dataResults[j][0] = Boolean.valueOf(false

```

```

        );
        dataResults[j][1] = rs;
        final List<String> temp = new LinkedList<
            String>();
        System.out.println(rs);
        for (final URI z : e.getRepository().
            getAnnotations(
                rs.getDocId())) {
            temp.add(z.getLocalName());
        }
        dataResults[j][2] = temp;

        dataResults[j][3] = rs.
            getRelatedExpandedSearches();
        j--;
    }
    tm.set(returned.size(), dataResults);
} catch (final Exception g) {
    g.printStackTrace();
    JOptionPane.showMessageDialog(null, "Oops!");
}

} // GEN-LAST:event\_botaoPesquisarMouseReleased

private void itemMenuNovoActionPerformed(
    final java.awt.event.ActionEvent evt) { // GEN
    -FIRST:event\_itemMenuNovoActionPerformed
    Repository rep = null;
    Definitions def = null;
    final JFileChooser menu = new JFileChooser();

    if (menu.showOpenDialog(this) == APPROVE\_OPTION)
    {
        try {
            rep = new Rep();

```

```

        def = new Defs();
    } catch (final Exception e1) {
        e1.printStackTrace();
        JOptionPane.showMessageDialog(null, "
            Could not open the file");
        return;
    }
} else {
    return;
}

e.setRepository(rep);
e.setDefinitions(def);

try {
    listaVocabulario.setListData(rep.getLabels().
        toArray());
} catch (final RemoteException ex) {
    ex.printStackTrace();
}

this.repaint();
} // GEN-LAST:event\_itemMenuNovoActionPerformed

private void itemMenuCarregarActionPerformed(
    final java.awt.event.ActionEvent evt) { // GEN
    -FIRST:event\_
        _itemMenuCarregarActionPerformed

        JOptionPane.showMessageDialog(null, "Not
            implemented");

} // GEN-LAST:event\_itemMenuCarregarActionPerformed

private void itemMenuGravarActionPerformed(
    final java.awt.event.ActionEvent evt) { // GEN

```

```

        -FIRST:event\
        _itemMenuGravarActionPerformed
    JOptionPane.showMessageDialog(null, "Not
        implemented");

}// GEN-LAST:event\_itemMenuGravarActionPerformed

// Variables declaration - do not modify//GEN-BEGIN:
    variables
private javax.swing.JButton botaoPesquisar;
private javax.swing.JTextField caixaDeBusca;
// private javax.swing.JMenu itemMenuBuscador;
private javax.swing.JMenuItem itemMenuCarregar;
private javax.swing.JMenuItem itemMenuGravar;
// private javax.swing.JMenu itemMenuMapa;
private javax.swing.JMenuItem itemMenuNovo;
public javax.swing.JScrollPane jScrollPane1;
public javax.swing.JScrollPane jScrollPane2;
public javax.swing.JScrollPane jScrollPane3;
public javax.swing.JScrollPane jScrollPane4;
private javax.swing.JLabel labelMapa;
private javax.swing.JLabel labelResultados;
private javax.swing.JList listaVocabulario;
private javax.swing.JLabel nomeVocabulario;
// public Display display;
public Painei display;

private javax.swing.JTable resultados;
Visualization vis;
// ////////////
private javax.swing.JMenuItem itemHistory;
private javax.swing.JMenuItem itemMAR;
private javax.swing.JMenuItem itemMAW;
private javax.swing.JMenuItem itemMSR;
private javax.swing.JMenuItem itemMTR;

```



```
private javax.swing.JMenuItem itemMTW;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenu menuContext;
private javax.swing.JMenu menuEdit;
private javax.swing.JMenu menuFile;
private javax.swing.JMenu menuSearch;

// End of variables declaration//GEN-END:variables
public void start() {
    Logger.getLogger("prefuse").setLevel(Level.OFF);
    initComponents();

    display.start();
}

static public class Model extends AbstractTableModel
{
    private static final long serialVersionUID = 1L;
    Object[][] data;
    int r, c;
    final String[] columnNames = { "Checked", "Name",
        "Annotated To",
        "Brought By" };

    Model(final int r) {
        this.r = r;
        c = 4;
        data = new Object[r][c];
    }

    public void reset() {
        data = new Object[r][c];
        fireTableDataChanged();
    }
}
```

```
}

public void set(final int r, final Object[][]
    data) {
    this.data = data;
    this.r = r;
    // this.c = data.length;
    fireTableDataChanged();
}

public int getRowCount() {
    return r;
}

public int getColumnCount() {
    return c;
}

@Override
public void setValueAt(final Object o, final int
    rowIndex,
    final int columnIndex) {
    data[rowIndex][columnIndex] = o;
}

@Override
public Class<?> getColumnClass(final int aColumn)
    {
    if (aColumn == 0) {
        return Boolean.valueOf(false).getClass();
    } else {
        return new Object().getClass();
    }
    // return getValueAt(0, aColumn).getClass();
}
```

```
@Override
public String getColumnName(final int aColumn) {
    return columnNames[aColumn];
}

@Override
public boolean isCellEditable(final int aRow,
    final int aColumn) {
    return (aColumn == 0 ? true : false);
}

public Object getValueAt(final int rowIndex,
    final int columnIndex) {
    if (columnIndex == 0) {
        return data[rowIndex][columnIndex];
    } else {
        return data[rowIndex][columnIndex];
    }
}
}

public List<KBResult> disambiguateKeyword(final List<
    KBResult> options,
    final String keyword) {
    final DisambDialog dd = new DisambDialog(
        getFrames()[0],
        rootPaneCheckingEnabled, options, keyword);
    if (options.size() > 1) {
        dd.setVisible(true);
        return dd.getSelectedOptions();
    } else {
        return options;
    }
}
```

```
    }  
}
```

Algoritmo B.20: Gui.java

```
package systemimpl.client.gui;  
  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.io.RandomAccessFile;  
import java.io.Serializable;  
import java.util.LinkedList;  
import java.util.List;  
import java.util.Vector;  
  
public class IO {  
  
    RandomAccessFile raf;  
  
    File arq;  
  
    IO() {  
  
    }  
  
    public void criar(final String path) throws  
        IOException {  
        arq = new File(path);  
        if (arq.exists()) {  
            throw (new IOException("Arquivo j existe."  
                ));  
        }  
    }  
}
```

```
    } else {
        if (!arq.createNewFile())
            throw (new IOException("Arquivo n o pode
                ser criado."));
        raf = new RandomAccessFile(arq, "rwd");
        System.out.println("Novo arquivo criado em "
            + arq.getAbsolutePath() + "\n");
    }
}

public boolean abrir(final String path) throws
FileNotFoundException {
    arq = new File(path);
    if (arq.exists()) {
        raf = new RandomAccessFile(arq, "rwd");
        return true;
    } else {
        return false;
    }
}

public String lerLinha() throws IOException {
    String resposta = "";
    String stop = "";
    do {
        stop = raf.readLine() + "\n";
        if (!stop.equals("null\n")) {
            resposta += stop;
        }
    } while (!stop.equals("null\n"));
    return resposta;
}
```

```
public int lerLinha(final String palavra, final
    String[] linhas,
        final int ind) throws IOException {
    int resposta = 0;
    while (!linhas[ind].startsWith(palavra)) {
        linhas[ind] = raf.readLine();
        resposta = +linhas[ind].length();
    }
    return resposta;
}

public void escrever(final String texto, final long
    pos) throws IOException {
    raf.seek(pos);
    raf.writeChars(texto);
}

public void escrever(final String texto) throws
    IOException {
    raf.writeBytes(texto);
}

public void escreverBinario(final Serializable s)
    throws IOException {

    final FileOutputStream arquivo = new
        FileOutputStream(arq
            .getAbsolutePath());

    final ObjectOutputStream saida = new
        ObjectOutputStream(arquivo);

    saida.writeObject(s);
}
```

```
public Serializable lerBinario() throws IOException,
    ClassNotFoundException {

    final FileInputStream arquivo = new
        FileInputStream(arq
            .getAbsolutePath());

    final ObjectInputStream entrada = new
        ObjectInputStream(arquivo);

    // Read an object
    return (Serializable) entrada.readObject();
}

public void moverPonteiro(final long pos) throws
    IOException {

    raf.seek(pos);

}

public boolean vazio() throws IOException {
    return raf.length() == 0;
}

public void fechar() throws IOException {
    raf.close();
}

public void truncar() throws IOException {
    raf.setLength(raf.getFilePointer());
}
}
```

```
public Vector<String> buscarPorArquivosTipo(final
String diretorioBase,
    final String extensao) {
    final Vector<String> resposta = new Vector<String>
        >();
    final List<File> raizes = new LinkedList<File>();
    File arquivo;
    raizes.add(new File(diretorioBase));

    int i = 0;

    while (i < raizes.size()) {
        arquivo = raizes.get(i++);
        File[] subs;
        if (arquivo.isDirectory()) {
            subs = arquivo.listFiles();

            for (final File sub : subs) {
                raizes.add(sub);
            }
        } else {
            final String r = arquivo.getAbsolutePath
                ();
            if (r.contains("." + extensao.replace(".",
                ""))) {
                resposta.add(r);
            }
        }
    }

    for (final String s : resposta) {
        System.out.println(">> " + s);
    }
}
```



```

        return resposta;
    }

}

```

Algoritmo B.21: IO.java

```

package systemimpl.client.gui;

import java.awt.geom.Rectangle2D;
import java.util.Iterator;

import prefuse.action.layout.Layout;
import prefuse.visual.DecoratorItem;
import prefuse.visual.VisualItem;

class LabelledEdgeLayout extends Layout {
    public LabelledEdgeLayout(final String group) {
        super(group);
    }

    @Override
    public void run(final double frac) {
        final Iterator<?> iter = m\_vis.items(m\_group);
        while (iter.hasNext()) {
            final DecoratorItem decorator = (
                DecoratorItem) iter.next();
            final VisualItem decoratedItem = decorator.
                getDecoratedItem();
            final Rectangle2D bounds = decoratedItem.
                getBounds();

            final double x = bounds.getCenterX();
            final double y = bounds.getCenterY();

            setX(decorator, null, x);

```

```

        setY(decorator, null, y);
    }
}

```

Algoritmo B.22: LabelledEdgeLayout.java

```

package systemimpl.client.gui;

import prefuse.Constants;
import prefuse.Display;
import prefuse.Visualization;
import prefuse.action.ActionList;
import prefuse.action.assignment.ColorAction;
import prefuse.action.assignment.DataColorAction;
import prefuse.action.layout.graph.ForceDirectedLayout;
import prefuse.activity.Activity;
import prefuse.controls.DragControl;
import prefuse.controls.PanControl;
import prefuse.controls.ZoomControl;
import prefuse.data.Graph;
import prefuse.data.Schema;
import prefuse.render.DefaultRendererFactory;
import prefuse.render.EdgeRenderer;
import prefuse.render.LabelRenderer;
import prefuse.util.ColorLib;
import prefuse.util.FontLib;
import prefuse.util.PrefuseLib;
import prefuse.util.force.DragForce;
import prefuse.util.force.ForceSimulator;
import prefuse.util.force.NBodyForce;
import prefuse.util.force.RungeKuttaIntegrator;
import prefuse.util.force.SpringForce;
import prefuse.visual.EdgeItem;
import prefuse.visual.NodeItem;
import prefuse.visual.VisualItem;

```

```

import prefuse.visual.expression.InGroupPredicate;
import systemimpl.CONSTANTS;

public class Paine1 extends Display implements Runnable {
    private static final long serialVersionUID = 1L;
    Thread relaxer;
    Visualization vis;
    Graph g;
    Boolean b = false;

    public static final String GRAPH = "graph";
    public static final String NODES = "graph.nodes";
    public static final String EDGES = "graph.edges";
    public static final String EDGE\_DECORATORS = "
        edgeDeco";
    public static final String NODE\_DECORATORS = "
        nodeDeco";
    private static final Schema DECORATOR\_SCHEMA =
        PrefuseLib
            .getVisualItemSchema();
    private static final Schema DECORATOR\_SCHEMA2 =
        PrefuseLib
            .getVisualItemSchema();

    static {
        DECORATOR\_SCHEMA.setDefault(VisualItem.
            INTERACTIVE, false);
        DECORATOR\_SCHEMA.setDefault(VisualItem.TEXTCOLOR
            , ColorLib.gray(128));
        DECORATOR\_SCHEMA.setDefault(VisualItem.FONT,
            FontLib.getFont("Tahoma",
                14));
    }
    static {
        DECORATOR\_SCHEMA2.setDefault(VisualItem.

```

```

        INTERACTIVE, false);
DECORATOR\_SCHEMA2.setDefault(VisualItem.
    TEXTCOLOR, ColorLib.gray(128));
DECORATOR\_SCHEMA2.setDefault(VisualItem.FONT,
    FontLib.getFont(
        "Verdana", 18));
}

public Paine1(final Graph g) {
    super(new Visualization());
    vis = getVisualization();
    this.g = g;
    vis.addGraph(GRAPH, g);
    vis.setInteractive(EDGES, null, false);
    vis.setValue(NODES, null, VisualItem.SHAPE,
        Integer
            .valueOf(Constants.SHAPE\_ELLIPSE));

    final DefaultRendererFactory drf = new
        DefaultRendererFactory();
    drf.setDefaultEdgeRenderer(new EdgeRenderer(
        Constants.EDGE\_TYPE\_CURVE,
        Constants.EDGE\_ARROW\_FORWARD));
    drf.add(new InGroupPredicate(EDGE\_DECORATORS),
        new LabelRenderer(
            CONSTANTS.ASSOCIATIONWEIGHT));
    drf.add(new InGroupPredicate(NODE\_DECORATORS),
        new LabelRenderer(
            CONSTANTS.LABEL));

    vis.setRendererFactory(drf);

    DECORATOR\_SCHEMA.setDefault(VisualItem.TEXTCOLOR
        , ColorLib.rgb(102,
            102, 255));
}

```

```

vis.addDecorators(EDGE\_DECORATORS , EDGES ,
    DECORATOR\_SCHEMA);

DECORATOR\_SCHEMA2.setDefault(VisualItem.
    TEXTCOLOR, ColorLib.gray(128));
vis.addDecorators(NODE\_DECORATORS , NODES ,
    DECORATOR\_SCHEMA2);

final ColorAction nStroke = new ColorAction(NODES
    ,
    VisualItem.STROKECOLOR);
nStroke.setDefaultColor(ColorLib.gray(100));
nStroke.add("\_hover", ColorLib.gray(50));

final ColorAction nFill = new ColorAction(NODES ,
    VisualItem.FILLCOLOR);
nFill.setDefaultColor(ColorLib.gray(255));
nFill.add("\_hover", ColorLib.gray(200));

final ColorAction nEdges = new ColorAction(EDGES ,
    VisualItem.STROKECOLOR);
nEdges.setDefaultColor(ColorLib.gray(100));

final int[] palette = new int[] { ColorLib.rgba
    (255, 150, 150, 150) };
final ForceSimulator fsim = new ForceSimulator(
    new RungeKuttaIntegrator());

final float gravConstant = -1f;
final float minDistance = -1f;
final float theta = 0.9f;

final float drag = 0.01f;
final float springCoeff = 1E-6f;
final float defaultLength = 100f;

```

```

fsim.addForce(new NBodyForce(gravConstant,
    minDistance, theta));
fsim.addForce(new DragForce(drag));
fsim.addForce(new SpringForce(springCoeff,
    defaultLength));

final ActionList layout = new ActionList(Activity
    .INFINITY);

layout.add(new ForceDirectedLayout(GRAPH, fsim,
    true));
final ForceDirectedLayout fdl = new
    CustomizedForceDirectedLayout(
        GRAPH, fsim, true);
layout.add(new LabelledEdgeLayout(EDGE\
    _DECORATORS));
layout.add(new LabelledEdgeLayout(NODE\
    _DECORATORS));
layout.add(fdl);
layout.add(nStroke);
layout.add(new DataColorAction(EDGES, CONSTANTS.
    ASSOCIATIONWEIGHT,
        Constants.NOMINAL, VisualItem.FILLCOLOR,
        palette));
layout.add(nFill);
layout.add(nEdges);
vis.putAction("layout", layout);
this.setSize(720, 500);
setHighQuality(true);
addControlListener(new DragControl());
addControlListener(new PanControl());
addControlListener(new ZoomControl());
vis.run("layout");
}

```

```
public void setSwitch(final Boolean b) {
    this.b = b;
}

public void run() {
    final Thread me = Thread.currentThread();
    while (relaxer == me) {
        if (b) {
            this.repaint();
        }
        try {

            Thread.sleep(100);
        } catch (final InterruptedException e) {
            break;
        }

    }

}

public void start() {
    relaxer = new Thread(this);
    relaxer.start();
}

static class CustomizedForceDirectedLayout extends
    ForceDirectedLayout {

    public CustomizedForceDirectedLayout(final String
        group,
            final ForceSimulator fsim, final boolean
                enforceBounds) {
        super(group, fsim, enforceBounds, false);
    }
}
```

```

    }

    @Override
    protected float getSpringLength(final EdgeItem e)
    {
        final NodeItem source = e.getSourceItem();
        final NodeItem target = e.getTargetItem();

        if (source.getInt("type") == target.getInt("
            type")) {
            return 140;
        } else {
            return 200;
        }
    }

    @Override
    protected float getMassValue(final VisualItem n)
    {
        return 1.0f;
    }

    @Override
    protected float getSpringCoefficient(final
        EdgeItem e) {
        return -1;
    }
}
}

```

Algoritmo B.23: Painel.java

B.2.4 SYSTEMIMPL.SERVER

```

package systemimpl.server;

```



```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.LinkedList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.openrdf.model.Literal;
import org.openrdf.model.Resource;
import org.openrdf.model.URI;

import system.server.definitions.Definitions;
import system.server.definitions.KBResult;

import com.ontotext.kim.client.CompareStyleConstants;
import com.ontotext.kim.client.entity.EntityDescription;
import com.ontotext.kim.client.model.PROTONSConstants;
import com.ontotext.kim.client.query.KIMQueryException;
import com.ontotext.kim.client.query.SemanticQuery;
import com.ontotext.kim.client.query.SemanticQueryResult;
import com.ontotext.kim.client.query.
    SemanticQueryResultRow;

public class Defs extends UnicastRemoteObject implements
    Definitions {

    private static final long serialVersionUID = 1L;
    String TERMLABEL = "http://www.w3.org/2000/01/rdf-
        schema\#label";

    public Defs() throws RemoteException {
        KIMFactory.connectKimService();
    }
}
```

```

public List<KBResult> possibleDenotationsFor(final
String keyword) {
    final LinkedList<KBResult> l = new LinkedList<
        KBResult>();
    try {
        final SemanticQuery semQuery = new
            SemanticQuery();
        semQuery.setReturnLabels(true);
        semQuery.addRequestedVar("Entity");
        semQuery.setClass("Entity", PROTONSConstants.
            CLASS\_ENTITY);
        semQuery.addNameRestriction("Entity",
            CompareStyleConstants.COMPARE\_STYLE\_
                _CONTAINS, keyword);
        final SemanticQueryResult results =
            KIMFactory.getQueryApi()
                .getEntities(semQuery);
        for (final Object r : results) {
            final SemanticQueryResultRow result = (
                SemanticQueryResultRow) r;
            l.add(new KBResult((URI) result.get(0),
                result.get(1)
                    .toString()));
        }

    } catch (final KIMQueryException ex) {
        Logger.getLogger(Defs.class.getName()).log(
            Level.SEVERE, null, ex);
    }
    return l;
}

public List<String> possibleLabelsFor(final Resource
entity) {
    final List<String> l = new LinkedList<String>();

```

```

        EntityDescription b = null;
        try {

            b = KIMFactory.getEntityApi().
                getEntityDescription(entity);
        } catch (final KIMQueryException ex) {
            ex.printStackTrace();
            return l;
        }

        for (final Literal label : b.getLabels()) {
            l.add(label.toString());
        }
        return l;
    }
}

```

Algoritmo B.24: Defs.java

```

package systemimpl.server;

import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.ontotext.kim.client.GetService;
import com.ontotext.kim.client.KIMService;
import com.ontotext.kim.client.corpora.
    KIMCorporaException;
import com.ontotext.kim.client.documentrepository.
    DocumentQuery;
import com.ontotext.kim.client.documentrepository.
    DocumentRepositoryAPI;
import com.ontotext.kim.client.entity.EntityAPI;
import com.ontotext.kim.client.query.DocumentQueryResult;

```

```
import com.ontotext.kim.client.query.  
    DocumentQueryResultRow;  
import com.ontotext.kim.client.query.KIMQueryException;  
import com.ontotext.kim.client.query.QueryAPI;  
  
public class KIMFactory {  
  
    private static KIMService kim;  
    private static EntityAPI apiEntity;  
    private static DocumentRepositoryAPI apiDocument;  
    private static QueryAPI apiQuery;  
  
    public static void connectKimService() {  
        if (kim == null) {  
            try {  
                kim = GetService.from();  
            } catch (final RemoteException ex) {  
                Logger.getLogger(KIMFactory.class.getName()  
                   ()).log(Level.SEVERE,  
                        null, ex);  
            } catch (final NotBoundException ex) {  
                Logger.getLogger(KIMFactory.class.getName()  
                   ()).log(Level.SEVERE,  
                        null, ex);  
            }  
        }  
    }  
  
    public static DocumentRepositoryAPI getDocumentApi()  
    {  
        if (apiDocument == null) {  
            try {  
                apiDocument = kim.  
                    getDocumentRepositoryAPI();  
            } catch (final RemoteException ex) {
```

```
        Logger.getLogger(KIMFactory.class.getName  
           ()).log(Level.SEVERE,  
                null, ex);  
    }  
}  
return apiDocument;  
}  
  
public static EntityAPI getEntityApi() {  
    if (apiEntity == null) {  
        try {  
            apiEntity = kim.getEntityAPI();  
        } catch (final RemoteException ex) {  
            Logger.getLogger(KIMFactory.class.getName  
               ()).log(Level.SEVERE,  
                    null, ex);  
        }  
    }  
    return apiEntity;  
}  
  
public static QueryAPI getQueryApi() {  
    if (apiQuery == null) {  
        try {  
            apiQuery = kim.getQueryAPI();  
        } catch (final RemoteException ex) {  
            Logger.getLogger(KIMFactory.class.getName  
               ()).log(Level.SEVERE,  
                    null, ex);  
        }  
    }  
    return apiQuery;  
}  
  
public static void main(final String args[]) {
```

```

try {
    connectKimService();
    final DocumentQueryResult documents =
        KIMFactory.getDocumentApi()
            .getDocuments(new DocumentQuery());
    for (final DocumentQueryResultRow doc :
        documents) {
        try {
            System.out.println(doc.getDocumentId
                () + " "
                    + doc.getDocument().
                        getFeatures().get("TITLE")
                            );
        } catch (final KIMCorporaException ex) {
            Logger.getLogger(KIMFactory.class.
                getName()).log(
                    Level.SEVERE, null, ex);
        }
    }
} catch (final KIMQueryException ex) {
    Logger.getLogger(KIMFactory.class.getName()).
        log(Level.SEVERE,
            null, ex);
}
}
}

```

Algoritmo B.25: KIMFactory.java

```

package systemimpl.server;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.HashSet;
import java.util.Iterator;

```

```
import java.util.LinkedList;
import java.util.List;
import java.util.Set;

import org.openrdf.model.Resource;
import org.openrdf.model.URI;
import org.openrdf.model.impl.URIImpl;

import system.server.repository.Repository;

import com.ontotext.kim.client.corpora.KIMAnnotation;
import com.ontotext.kim.client.corpora.KIMAnnotationSet;
import com.ontotext.kim.client.corpora.
    KIMCorporaException;
import com.ontotext.kim.client.corpora.KIMDocument;
import com.ontotext.kim.client.corpora.KIMFeatureMap;
import com.ontotext.kim.client.documentrepository.
    DocumentQuery;
import com.ontotext.kim.client.model.FeatureConstants;
import com.ontotext.kim.client.query.DocumentQueryResult;
import com.ontotext.kim.client.query.
    DocumentQueryResultRow;
import com.ontotext.kim.client.query.KIMQueryException;

public class Rep extends UnicastRemoteObject implements
    Repository {

    private static final long serialVersionUID = 1L;
    private String name;
    public static String ANNOTATION = "http://www.w3.org
        /2000/01/rdf-schema#isDefinedBy";
    public static String TERMLABEL = "http://www.w3.org
        /2000/01/rdf-schema#label";

    public Rep() throws RemoteException {
```

```
        KIMFactory.connectKimService();
    }

    public Set<String> getLabels() {
        return null;
    }

    public List<Long> retrieve(final Resource entity) {
        final List<Long> l = new LinkedList<Long>();
        final DocumentQuery q = new DocumentQuery();

        final Set<String> entitiesRestriction = new
            HashSet<String>();
        entitiesRestriction.add(entity.toString());

        q.setCooccurringEntities(entitiesRestriction);

        try {
            final DocumentQueryResult documents =
                KIMFactory.getDocumentApi()
                    .getDocuments(q);
            for (final DocumentQueryResultRow doc :
                documents) {
                l.add(doc.getDocumentId());
            }
        } catch (final KIMQueryException ex) {
            ex.printStackTrace();
        }
        return l;
    }

    public List<URI> getAnnotations(final long docID) {
        final List<URI> l = new LinkedList<URI>();
        KIMDocument doc;
        KIMAnnotationSet annotations;
```



```

try {
    doc = KIMFactory.getDocumentApi().
        loadDocument(docID);
    annotations = doc.getAnnotations();

    final Iterator<?> annIterator = annotations.
        iterator();

    while (annIterator.hasNext()) {
        final KIMAnnotation kimAnnotation = (
            KIMAnnotation) annIterator
                .next();
        final KIMFeatureMap kimFeatures =
            kimAnnotation.getFeatures();
        if (kimFeatures != null && filter(
            kimAnnotation.getType())) {
            final String resourceUri = (String)
                kimFeatures
                    .get(FeatureConstants.INSTANCE);
            l.add(new URIImpl(resourceUri));
        }
    }
} catch (final KIMQueryException ex) {
    ex.printStackTrace();
} catch (final KIMCorporaException ex) {
    ex.printStackTrace();
}

return l;
}

@Override
public String toString() {
    return name;
}

```

```
public String getName() {
    return name;
}

public void setName(final String name) {
    this.name = name;
}

private boolean filter(final String type) {
    return !(type.equals("Percent") || type.equals("
        Money") || type
            .equals("Time"));
}
}
```

Algoritmo B.26: Rep.java

REFERÊNCIAS

- BAEZA-YATES, R.; RIBEIRO-NETO, B. et al. *Modern information retrieval*. [S.l.]: Addison-Wesley Harlow, England, 1999.
- BAYERL, P. et al. Methodology for Reliable Schema Development and Evaluation of Manual Annotations. *Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (KCAP)*, 2003.
- BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic Web. *Scientific American*, v. 284, n. 5, p. 28–37, 2001.
- BRIN, S.; PAGE, L. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, Elsevier, v. 30, n. 1-7, p. 107–117, 1998.
- BROEKSTRA, J.; KAMPMAN, A.; HARMELEN, F. van. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *The Semantic Web-ISWC*, Springer, v. 2342, p. 54–68, 2002.
- D'AGOSTINI, C. et al. Inferring user's intentions through context. *Sessão de Posters do Simpósio Brasileiro de Bancos de Dados (SBBDD)*, p. 7–10, 2007.
- D'AGOSTINI, C. et al. Contextual semantic search - capturing and using the user's context to direct semantic search. *In 10th International Conference on Enterprise Information Systems (to appear)*, 2008.
- DAVIES, J.; STUDER, R.; WARREN, P. *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. [S.l.]: John Wiley & Sons, 2006.
- DEGLER, D.; LEWIS, R. Maintaining Ontology Implementations: The Value of Listening. *Extreme Markup Languages 2004: Proceedings*, IDEAlliance, 2004.
- FRAUENFELDER, M. Sir Tim Berners-Lee. *Technology Review (September 30, 2004)*. URL: <http://www.techreview.com/articles/04/10/frauenfelder1004.asp>, 2004.
- GRUBER, T. *Ontology*. [S.l.]: Springer, 2008. <http://tomgruber.org/writing/ontology-definition-2007.htm>. Acessada em 23/06/2008.
- GUHA, R.; MCCOOL, R.; MILLER, E. Semantic search. *Proceedings of the 12th international conference on World Wide Web*, ACM Press New York, NY, USA, p. 700–709, 2003.
- HUNT, A.; THOMAS, D. *The pragmatic programmer: from journeyman to master*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2000.
- KIRYAKOV, A. et al. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, Elsevier, v. 2, n. 1, p. 49–79, 2004.

- MANGOLD, C. A survey and classification of semantic search approaches. *Int. J. Metadata, Semantics and Ontology*, v. 2, n. 1, p. 23–34, 2007.
- MANI, A.; SUNDARAM, H. Modeling user context with applications to media retrieval. *Multimedia Systems*, Springer, v. 12, n. 4, p. 339–353, 2007.
- PEPPER, S. The TAO of topic maps. In: *Proceedings of XML Europe*. [S.l.: s.n.], 2000.
- REEVE, L.; HAN, H. A comparison of semantic annotation systems for text-based web documents. *Web Semantics and Ontology*, v. 1, 2006.
- SATYANARAYANAN, M. et al. Pervasive computing: vision and challenges. *IEEE Personal Communications*, Washington, DC, USA, v. 8, n. 4, p. 10–17, 2001.