UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
CURSO DE MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

FERNANDO SOARES DE FRANÇA

ENRIQUECIMENTO SEMÂNTICO DE PADRÕES MINERADOS EM GRAFOS UTILIZANDO ONTOLOGIAS

MOSSORÓ - RN
2011

FERNANDO SOARES DE FRANÇA

ENRIQUECIMENTO SEMÂNTICO DE PADRÕES MINERADOS EM GRAFOS UTILIZANDO ONTOLOGIAS

Dissertação apresentada ao Mestrado de Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semiárido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: D. Sc. Marcelino Pereira dos Santos Silva - UERN

MOSSORÓ - RN
2011

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

França, Fernando Soares de .
Enriquecimento semântico de padrões minerados em grafos utilizando ontologias. / Fernando Soares de França. – Mossoró, RN, 2011.

68 f.
Orientador(a): D. Sc. Marcelino Pereira dos Santos Silva.

Dissertação (Mestrado em Ciência da Computação). Universidade do Estado do Rio Grande do Norte. Curso de Mestrado em Ciência da Computação.

1. Grafos - Dissertação. 2. Ontologias - Dissertação. 3. Mineração de dados - Dissertação. I. Silva, Marcelino Pereira dos Santos. II.Universidade do Estado do Rio Grande do Norte. III.Título.

UERN/BC CDD 005.1

Bibliotecária: Elaine Paiva de Assunção CRB 15 / 492

FERNANDO SOARES DE FRANÇA

ENRIQUECIMENTO SEMÂNTICO DE PADRÕES MINERADOS EM GRAFOS UTILIZANDO ONTOLOGIAS

Dissertação apresentada ao Mestrado em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM __/__/____

BANCA EXAMINADORA

Prof. D.Sc. Marcelino Pereira dos Santos Silva (Orientador)
Universidade do Estado do Rio Grande do Norte - UERN

Prof. D.Sc. Francisco Chagas de Lima Júnior (Co-orientador)
Universidade do Estado do Rio Grande do Norte - UERN

Prof. D.Sc. Dario José Aloise
Universidade do Estado do Rio Grande do Norte - UERN

Prof. D.Sc. Jerffeson Teixeira de Souza
Universidade Estadual do Ceará - UECE

Dedicatória

Dedico este trabalho a Francisco Oliveira de França (meu pai) e a Maria Luciene Soares de França (minha mãe), as pessoas mais importantes da minha vida.

Agradecimentos

Agradeço primeiramente a DEUS, que fez com que eu não desistisse dos meus objetivos.

Agradeço a toda minha família, pelo apoio dado em todos os momentos que precisei, em especial ao meu pai e minha mãe.

Agradeço a todos meus amigos. Ao Cleone (Clecle), Jusciê, Artur (Cachimba), Mailson (rato), Rangel (Power), Phelipe (bahiano), Ana, Ana kátia, Mário Jorge, Mário Messi, Kikiko, Galego, Anchieta, Isaac, João paulo, Otaciana, Welliana, Aislânia, Lenardo, Gracon e a todos os outros que também contribuíram durante este período e que por ventura esqueci de citar. Um agradecimento muito especial para minha namorada lisabelle, que na reta final da defesa da dissertação esteve sempre ao meu lado apoiando e me dando forças.

Agradeço a todos os professores do Mestrado, em especial ao Marcelino (orientador). Agradeço também à Angélica, a nossa coordenadora durante este processo. Agradeço a todos os funcionários da instituição (UERN) em especial para Chagas de educação física e para Rosita da secretaria dos mestrados.

Agradeço a todos aqueles que passaram pelo caminho e me ajudaram de alguma forma.

"A dúvida é o princípio da sabedoria."

Aristóteles

Resumo

A maioria das pesquisas realizadas em mineração de grafos foca no desenvolvimento de eficientes algoritmos para a descoberta de padrões frequentes. No entanto, pouca atenção é dada à interpretação destes. A grande quantidade de padrões minerados e a falta de informações de contexto dificultam a interpretação e exploração dos mesmos. Este trabalho propõe o enriquecimento semântico de padrões minerados em grafos, onde ontologias OWL enriquecidas com regras SWRL são usadas para implementar a semântica de domínio. Dessa forma, além dos padrões minerados, informações semânticas sobre estes são recuperadas da ontologia de domínio e exibidas ao usuário. Assim, é propiciada uma interpretação mais eficiente dos dados, bem como a exploração de informações estratégicas dos mesmos. Uma base de dados de proteínas é utilizada na demonstração do processo de enriquecimento semântico dos padrões de grafos.

Palavras-chave: Mineração de dados. Grafos. Ontologias.

Abstract

Most research on graph mining focus on the development of efficient algorithms to discover frequent patterns. However, less attention is paid to their interpretation. The huge volume of discovered patterns and the lack of context information generally embarrasses the interpretation and exploration of such patterns. This work proposes the semantic enrichment of patterns mined from graph datasets, through OWL ontologies enriched with SWRL rules, used to implement the domain semantics. Thus, in addition to the mined patterns, semantic information about these are retrieved from the domain ontology and displayed to the user. Thus, it is given an interpretation more efficient of data, and the strategic use of information from them. A protein database is used in the demonstration of process of semantic enrichment of mined patterns in graphs.

Keywords: Data mining. Graph. Ontology.

Lista de Figuras

Lista de Siglas e Abreviaturas

VLSI - *Very-large-scale integration*
JUNG - *Java Universal Network/Graph Framework*
API - *Application Programming Interfaces*
SQL - *Structured Query Language*
DCBD - Descoberta de conhecimento em banco de dados
SMS - Short Message Service
AGM - *Apriori-based Graph Mining*
FSG - *Frequent SubGraphs*
MoFa - *Molecule Fragment Miner*
FFSM - *Fast Frequent Subgraph Mining*
SPIN - *SPanning tree based maximal graph mINing*
Gaston - *GrAph Sequence Tree extractiON*
gSpan - *graph-based Substructure pattern*
DFS - *depth-first search*
Flogic - Frame Logic
GRAIL - GALEN Representation And Integration Language
OCML - Operational Conceptual Modeling Language
OML - Ontology Markup Language
RDF - Resource Description Framework
RDFS - RDF Schema
NKRL - Narrative Knowledge Representation Language
SHOE - Simple HTML Ontology Extensions
OIL - Ontology Interchange Language
DAML - DARPA Agent Markup Language
OWL - Web Ontology Language
UNA - Unique Name Assumption
CODE4 - Conceptually Oriented Description Environment
GKB-editor - Generic Knowledge Base Editor
JOE - Java Ontology Editor
WebODE - Web Ontology Design Environment
HP - Hewlett-Packard
OKBC - *Open Knowledge Base Connectivity Protocol*
SWRL - Semantic Web Rule Language
SQWRL - Semantic Query-Enhanced Web Rule Language
COI - *Conflict of Interest*

FOAF - *friend-of-a-friend*
W3C - *World Wide Web Consortium*

Sumário

1 Introdução

Com o passar do tempo a quantidade de dados armazenados em servidores vem crescendo de forma acelerada. Fatores como barateamento do hardware e massificação da internet tem colaborado bastante para isso [7]. Crescimento que não se restringe apenas ao setor corporativo, mas também ao âmbito governamental e científico. Com o crescimento dos repositórios de dados, cresce também a dificuldade de analisar e extrair informações estratégicas. Devido a essa dificuldade na análise dos dados, cada vez mais vem se investindo em pesquisas de novas técnicas e ferramentas para análise automática e inteligente dos dados, especificamente a mineração de dados. Em termos gerais, mineração de dados pode ser definida como o processo de análise de grandes bancos de dados de forma semi-automática para encontrar padrões úteis [8].

Muitas aplicações científicas e comerciais requerem a descoberta de padrões mais complexos que os padrões convencionais encontrados, por exemplo, em repositórios relacionais, o que requer um esforço extra para serem descobertos. Tais padrões sofisticados podem ser encontrados em estruturas mais complexas como por exemplo em grafos, cuja extração de padrões denomina-se mineração de grafos. Dentre os padrões que podemos encontrar em grafos destacam-se os subgrafos frequentes [4].

A maioria das pesquisas em mineração de grafos focam no desenvolvimento de algoritmos eficientes para descoberta dos subgrafos frequentes, porém pouca atenção é dada à interpretação dos padrões encontrados [9]. As técnicas convencionais se baseiam em informações estatísticas (por exemplo, suporte mínimo) para o processo de mineração. No entanto, o excessivo volume de padrões gerados nos resultados de uma mineração de grafos juntamente com a falta de informações de contexto (semântica dos dados), vêm trazendo dificuldades na interpretação e exploração desses padrões.

Diante desta constatação, esse trabalho visa desenvolver uma arquitetura capaz de extrair informações semânticas agregadas aos padrões de grafos minerados, que serão obtidas através de ontologias OWL com regras SWRL embutidas. Dessa forma, a abordagem proposta trará avanços relevantes à mineração de grafos, especialmente o enriquecimento semântico dos padrões minerados e, consequentemente, uma interpretação mais eficiente dos mesmos. No estudo de caso foi utilizada uma base de dados de proteínas.

Este trabalho está organizado da seguinte forma: o capítulo 2 aborda conceitos básicos de grafos, o cápítulo 3 discorre sobre mineração de grafos. O capítulo 4 apresenta conceitos importantes sobre ontologias. O processo de enriquecimento semântico proposto é detalhado no capítulo 5, e no capítulo 6 um estudo de caso em uma bases de dados de proteínas é apresentado. Por fim, segue-se as conclusões e trabalhos futuros.

2 Grafos

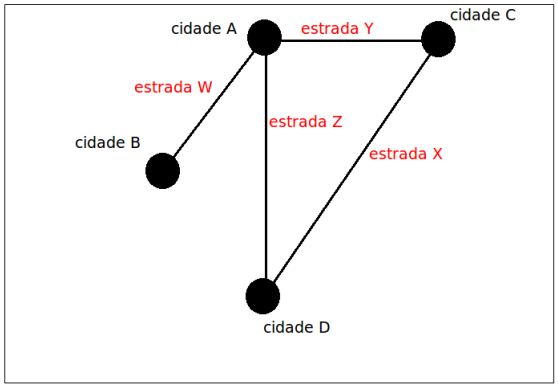
Um grafo é uma estrutura de abstração que representa um conjunto de elementos denominados *nós* ou vértices e suas relações de interdependência ou arestas [10]. Denominando por V o conjunto de vértices da estrutura, e por E o conjunto de arestas ou ligações entre os vértices, um grafo pode ser representado por: $G = (V, E)$ [10]. Em uma ampla gama de domínios os dados podem ser naturalmente modelados de tal forma, como por exemplo, redes de computadores que consistem de roteadores e computadores como sendo os nós e os links entre eles as arestas do grafo. Redes sociais também podem ser modeladas como grafos, onde a característica de interconexão entre indivíduos de uma rede social facilita esse tipo de modelagem. Também podemos citar outros domínios que podem ser modelados como grafos: estrutura de proteínas, Localização de bugs de software, dentre outras [11].

Nas subseções a seguir serão apresentados alguns conceitos importantes a respeito de grafos, cujo entendimento é de grande importância para a sequência desse trabalho.

2.1 Representação de um grafo

Basicamente, um grafo pode ser representado da seguinte maneira: círculos para representar os vértices e linhas representando as arestas que ligam os vértices. Vale também ressaltar que na maioria das vezes os vértices de um grafo possuem um rótulo, que é uma espécie de identificador. Embora menos comum, as arestas de um grafo podem também ser rotuladas. Para se entender um pouco melhor essa questão da rotulação, podemos citar um exemplo de um grafo, onde temos cidades como os vértices. Dessa forma os nomes dessas cidades passam a ser os rótulos dos vértices, e temos as estradas que ligam as cidades como sendo as arestas, e os nomes das estradas sendo os rótulos das arestas. A figura 1 mostra um esquema para a representação de um grafo.

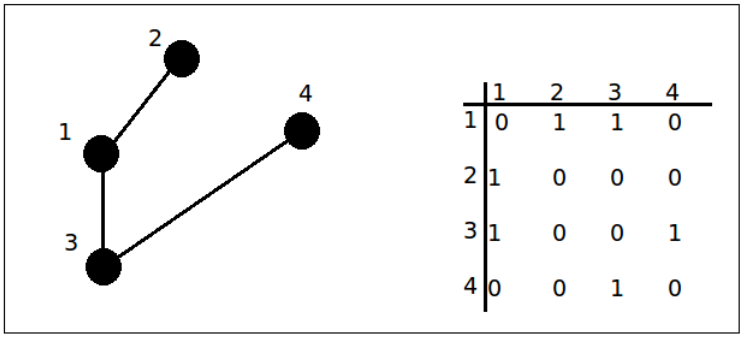
Figura 1: Representação de um grafo



Apesar da clareza de um esquema gráfico, não é algo trivial para um computador processar uma figura, por isso é necessário o uso de modelos matemáticos para essa representação. Um formalismo muito utilizado na representação matemática de um grafo é a matriz de adjacência [12].

Uma matriz de adjacência é uma matriz onde os vértices de um grafo são associados às linhas e às colunas. A matriz de adjacência pode ser indicada por $A = [a_{ij}]$. Dessa forma, para cada par de rótulos onde houver ligação será inserido um valor na matriz. Por convenção se utiliza 1 para indicar ligação entre vértices e 0 no caso de não haver ligação, e no caso de grafos valorados pode-se usar o próprio valor na matriz [12]. A figura 2 mostra um grafo com sua representação numa matriz de adjacência.

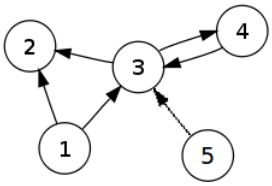
Figura 2: Matriz de adjacência



2.2 Grafo Orientado

Em um grafo orientado $G = (V, E)$, cada aresta $(i, j) \in E$ com orientação do vértice i até o vértice j. Um grafo não orientado é um grafo orientado, onde as arestas possuem ambos caminhos, ou seja, $(i, j) \in E \Rightarrow (j, i) \in E$ [13]. No caso de grafos orientados, suas arestas podem ser chamadas de arcos. A figura 3 mostra a representação gráfica de um grafo orientado.

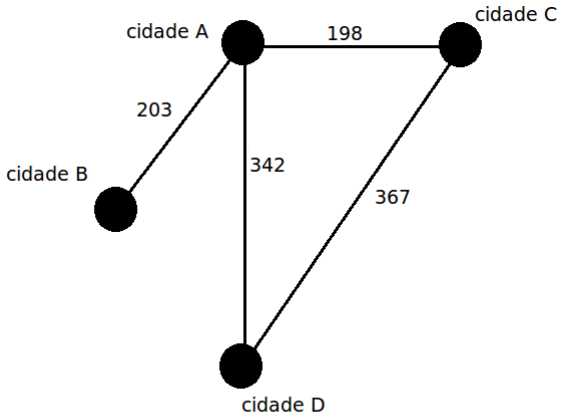
Figura 3: Grafo orientado



2.3 Grafo Rotulado e Grafo Valorado

Um grafo $G = (V, E)$ é dito ser rotulado em vértices ou arestas, quando a cada vértice ou aresta estiver associado um rótulo. Um grafo valorado $G = (V, E, W)$ possui um conjunto de vértices V, um conjunto de arestas E e o conjunto W, que representa os valores (ou pesos) correspondentes às arestas [12]. Um bom exemplo de grafo valorado seria um grafo que liga diversas cidades e os valores de sua arestas seriam correspondentes à distância entre elas. Um grafo não valorado é um caso especial de grafo valorado, onde todas as suas arestas possuem valor igual a 1. A figura 4 mostra um exemplo de grafo rotulado (cada vértice possui um rótulo referente a uma cidade) e valorado onde os valores das arestas representam as distâncias entre as cidades em KM.

Figura 4: Grafo Valorado



2.4 Subgrafo

Se temos um subgrafo H de $G = (V, E)$, então o conjunto de vértices de H está contido no de G e o conjunto de arestas de H também está contida em G, ou seja, $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Existem alguns tipos especiais de subgrafos: o abrangente e o induzido. Um subgrafo abrangente é um grafo que contém todos os vértices de outro, mas não obrigatoriamente todas as arestas. A Figura 5 mostra um exemplo de subgrafo abrangente. Um subgrafo induzido é um grafo que possui, do grafo original, apenas alguns vértices (no mínimo um vértice do grafo original tem que ser suprimido) e todas as ligações entre eles, que existam no grafo original [12]. A figura 6 mostra um subgrafo induzido, indicado pela cor vermelha dos seus vértices.

.

Figura 5: Subgrafo abrangente

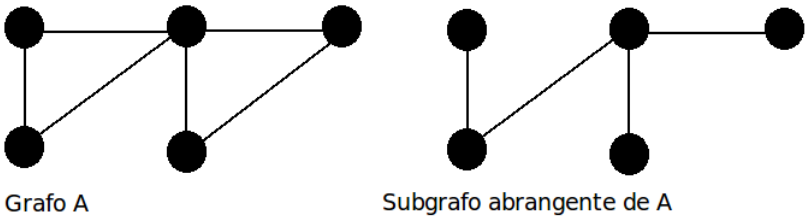
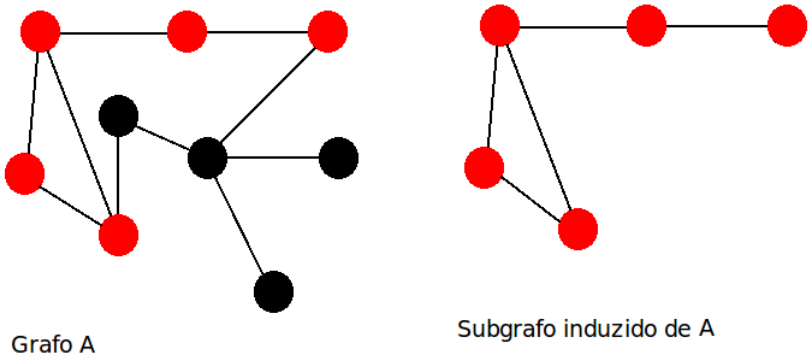


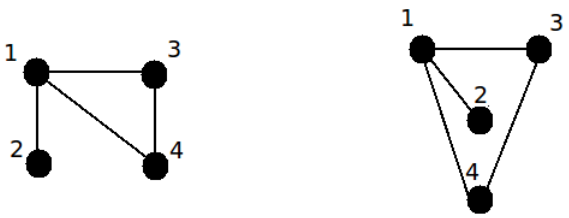
Figura 6: Subgrafo induzido



2.5 Isomorfismo de grafos

Dados dois grafos A e B, eles são ditos *isomorfos* quando é possível estabelecer uma correspondência biunívoca entre seus vértices e arestas, bem como entre suas relações vértices X arestas. Grafos isomorfos são analiticamente idênticos, contudo podem ser representados graficamente de forma diferente [10]. A figura 7 mostra um exemplo de isomorfismo de grafos.

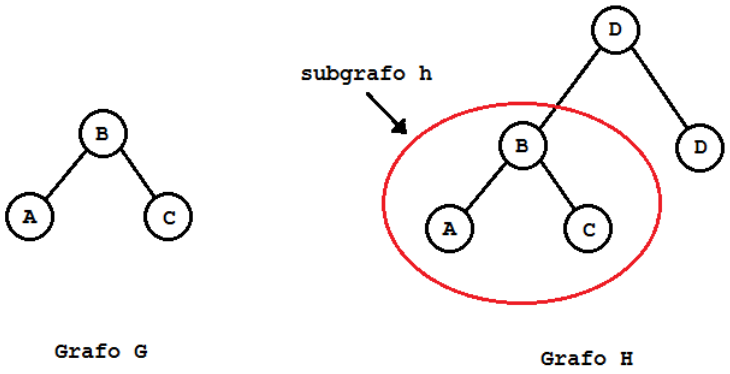
Figura 7: Dois grafos isomorfos



2.6 Isomorfismo de subgrafos

Um dado grafo G é isomorfo a um subgrafo h de um outro grafo H se existe um mapeamento dos nós de G sobre os nós de h, tal que todas relações de adjacência entre os vértices sejam mantidas. A figura 8 ilustra o isomorfismo de subgrafo, onde um grafo G é isomorfo ao subgrafo h do grafo H. O problema do isomorfismo de subgrafos é frequente numa vasta gama de aplicações em computação, onde a busca por um menor grafo a partir de um grafo maior se faz necessária, como exemplo dessas aplicações podemos citar: bioinformática, robótica e projetos de *Very-large-scale integration* (VLSI) [14]. O problema não possui grande importância apenas no que diz respeito a aplicações práticas, mas também é de grande interesse teórico, devido à sua conhecida complexidade NP-completo. Nenhum algoritmo eficiente para este problema é conhecido até o momento [15], e foi conjecturado por muitos especialistas que nenhum algoritmo de tempo polinomial existe por causa de sua NP-completude [14].

Figura 8: Isomorfismo de subgrafos



2.7 Ferramentas

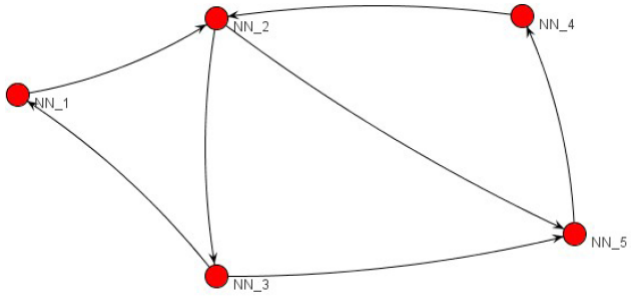
Existe um grande número de ferramentas e bibliotecas de software para manipulação e visualização de grafos, como por exemplo: Prefuse [16], JGraph [17], GUESS [18], Graphviz [19], *Java Universal Network/Graph Framework* (JUNG) [1], dentre outras [20]. Na subseção a seguir será abordado algumas características básicas do JUNG, que foi a biblioteca de manipulação de grafos escolhida para a implementação deste trabalho dentre as supracitadas.

2.7.1 JUNG

O JUNG é uma biblioteca de software de código aberto que fornece a capacidade de manipulação, análise e visualização de dados que podem ser representados como grafo ou rede. O JUNG é escrito em linguagem de programação Java, permitindo que as aplicações desenvolvidas baseadas em JUNG façam uso da extensa capacidade da API[1] Java, bem como de outras bibliotecas existentes de terceiros escritas em Java [22, 23].

JUNG inclui implementações de um grande número de algoritmos de teoria dos grafos, mineração de dados e análise de redes sociais, incluindo clusterização, decomposição, otimização, geração aleatória de grafos, análise estatística, cálculo de distâncias de rede e medidas de importância em redes (centralidade, PageRank, HITS, etc.)[24]. A figura 9 mostra um exemplo de grafo criado através do JUNG.

Figura 9: Um grafo dirigido implementado em JUNG. Fonte[1].



[1]Uma *Application Programming Interfaces* (API) é uma interface bem definida, que permite a um componente de software acessar outro componente de forma transparente através de rotinas de programação[21]

3 Mineração de Grafos

3.1 Mineração de Dados

A cada dia que se passa o volume de dados armazenados em servidores cresce de forma muito rápida. Fatores como barateamento do hardware e massificação da internet tem colaborado para isso [7]. Esse crescimento não se restringe apenas ao setor corporativo, mas também ao âmbito governamental e científico. Com o crescimento dos repositórios de dados, cresce também a dificuldade de analisar e extrair informações estratégicas, onde muitas vezes a capacidade humana e técnicas convencionais de gerenciamento de banco de dados (por exemplo, consultas *Structured Query Language* (SQL)) não são capazes de extrair grande parte do conhecimento intrínseco desses repositórios. Por isso cada vez mais vem se investindo em pesquisas de novas técnicas e ferramentas para análise automática e inteligente dos dados, dentre elas a mineração de dados. Em termos gerais, mineração de dados pode ser definida como o processo de análise inteligente de grandes bancos de dados de forma semi-automática para encontrar padrões úteis [25, 26].

A mineração de dados vem sendo alvo de grande atenção da indústria da informação nos últimos anos, devido a eminente necessidade de transformar os dados dos grandes repositórios em informações úteis e conhecimento. Essas informações e conhecimento adquiridos podem ser usados em várias espécies de aplicações, que vão desde análise de mercado, detecção de fraude, fidelização de clientes, controle de produção e exploração científica [4].

Descoberta de conhecimento em banco de dados (DCBD) é o processo não trivial de identificar nos dados padrões válidos, novos, úteis e compreensíveis. Mineração de dados é uma das etapas do processo de DCBD. Segundo [8] o processo de DCBD consiste na sequencia iterativa dos seguintes passos:

1. **Definição** – da fonte de dados a ser utilizada, bem como do tipo de conhecimento a descobrir, o que pressupõe uma compreensão do domínio da aplicação e da contribuição que esse novo conhecimento pode acarretar;

2. **Seleção** – de um conjunto de dados, ou focar num subconjunto, onde a descoberta de conhecimento deve ser realizada, dessa forma sendo criado um conjunto de dados alvo;

3. **Pré-processamento** – consiste de operações básicas tais como remoção de ruídos (limpeza dos dados), coleta da informação necessária para modelar ou estimar

ruído, escolha de estratégias para manipulação dos campos de dados com valores ausentes, formatação dos dados de forma a adequá-los à ferramenta de mineração;

4. **Transformação (Formatação) dos dados** – onde os dados são transformados ou consolidados em uma forma apropriada para a mineração.

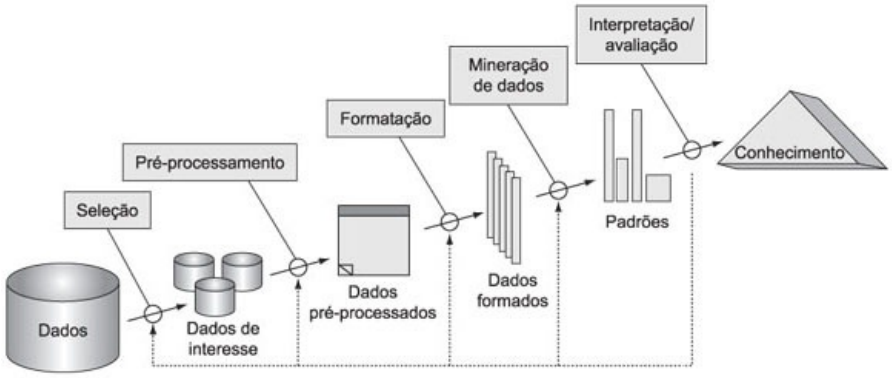
5. **Mineração de dados** – é a etapa essencial, onde métodos inteligentes (classificação, associação, clusterização, dentre outros) são aplicados de forma a se extrair padrões nos dados;

6. **Avaliação dos padrões** – para identificar que padrões são verdadeiramente interessantes para representação do conhecimento;

7. **Apresentação do conhecimento** – onde técnicas de visualização e representação são usadas para apresentar o conhecimento extraído ao usuário.

A figura 10 mostra todas as etapas do processo de DCBD.

Figura 10: Overview das etapas do processo de DCBD. Traduzido de [2].



Mineração de dados é um relevante e promissor tema de pesquisa sendo utilizado em muitas aplicações do mundo real. A seguir serão mostrados alguns dos principais campos de atuação da mineração de dados:

- **Análise de dados financeiros** – Quando solicita-se um empréstimo, é preciso preencher um formulário com perguntas sobre relevantes informações pessoais e financeiras. Essas informações são usadas pelas companhias financeiras como base na decisão de aceitar ou não o empréstimo solicitado. Essa decisão tipicamente é tomada em dois estágios, primeiramente métodos estatísticos são utilizados para determinar os casos pontuais de aceitação e rejeição dos empréstimos. Os casos não contemplados nessa primeira etapa são mais difíceis e são levados a

julgamento humano. Utilizando-se mineração de dados é possível construir mecanismos que viabilizem a predição de grande parte desses casos considerados mais difíceis [27]. No domínio financeiro, mineração de dados também pode ser usada para detecção de lavagem de dinheiro e outros crimes [4].

- **Indústria Varejista** - É a maior área de aplicação para mineração de dados, uma vez que reúne grande quantidade de dados sobre vendas, histórico de compras dos clientes, transporte de produtos, consumo e serviços. E a cada dia que se passa esse montante de dados cresce devido às grandes facilidades suportadas por compras na WEB ou *e-commerce*. Mineração de dados na indústria varejista pode ajudar a identificar comportamento de compra dos clientes, criação de *layouts* otimizados para prateleiras, descoberta de padrões de compras e tendências para marketing dirigido, desenvolvimento de rotas melhoradas para o transporte de mercadorias dentre outras [4].

- **Análise de imagens** - Desde o início do desenvolvimento da tecnologia de satélites, cientistas ambientalistas vem tentando detectar manchas de petróleo em regiões costeiras a partir de imagens de satélite. Esse tipo de detecção é um processo manual e caro, que exige pessoal altamente treinado para avaliar cada região da imagem. Mineração de dados é utilizada para filtrar imagens passíveis de conter manchas, diminuindo dessa forma o trabalho de análise manual das imagens [7].

- **Industria de Telecomunicações** - o setor de Telecomunicações vem evoluindo rapidamente, oferecendo serviços de telefonia local e longa distância, além de outros serviços como fax, pager, telefone celular, email, Short Message Service (SMS) e outros tipos de tráfego de dados. Esse rápido crescimento cria uma grande demanda para mineração de dados, a fim de ajudar no entendimento do negócio, identificação de padrões de telecomunicações, conter atividades fraudulentas, fazer uma melhor utilização dos recursos e melhorar a qualidade do serviço [4].

- **Análise de dados biológicos** - Bancos de dados biológicos possuem uma ampla variedade de tipos de dados, muitas vezes com rica estrutura relacional. Consequentemente técnicas de mineração de dados são freqüentemente aplicadas a dados biológicos [28], algumas dessas aplicações são: busca de similaridade e análise comparativa de múltiplas sequências de proteinas, descoberta de padrões estruturais e análise de redes genéticas, identificação de co-ocorrência de sequências de genes e relacionamento destas a diferentes fases do desenvolvimento de uma doença [4].

- **Detecção de intrusos** - A segurança dos sistemas de computadores e de seus dados estão sob risco contínuo. O crescimento da Internet e o aumento da disponibilidade de ferramentas para ataques em redes tornou a detecção de intrusos um componente fundamental na administração de redes [4]. Mineração de dados vem sendo utilizada para geração de regras de classificação que possam ajudar a detectar esses comportamentos suspeitos que se materializam em forma de invasões aos sistemas computacionais e bases de dados [29].

3.2 Mineração de Grafos

Muitas aplicações científicas e comerciais requerem a descoberta de padrões mais complexos que os padrões convencionais, por exemplo encontrados em repositórios relacionais, o que requer um esforço extra para serem descobertos. Tais padrões sofisticados podem ser encontrados e representados como árvores, grafos, redes e outras estruturas complexas [4].

Um grafo é um conjunto de nós conectados por um conjunto de arestas. Dentro do contexto de banco de dados, esses nós representam entidades individuais, enquanto as arestas ligações entre as mesmas [13]. Vários domínios do mundo real podem ser modelados na forma de grafos, como por exemplo: redes de computadores, ecologia, biologia, sociologia, psicologia de usuário, recuperação de informações dentre outros. A extração de padrões relevantes dessas bases modeladas como grafos é denominada mineração de grafos [4].

Muitas das tradicionais aplicações de mineração de dados também se aplicam ao caso de grafos. Essas aplicações são mais difíceis de se implementar no contexto de grafos, devido a restrições adicionais decorrente da natureza estrutural de grafos. Apesar destes desafios, algumas técnicas da mineração de dados tradicional vêm sendo estudadas e aplicadas a mineração de grafos, dentre as principais temos: clusterização, classificação e mineração de padrões frequentes (subgrafos frequentes) [3]. Outro ponto importante a ser discutido acerca de mineração de grafos são os padrões globais de grafos.

3.2.1 Padrões Globais de Grafo

Padrões globais de grafo têm como objetivo, encontrar propriedades que distinguam grafos do mundo real de grafos gerados sinteticamente, detectar anomalias em grafos, e gerar grafos sintéticos, porém realísticos [30]. Os mais comuns padrões globais de grafo utilizados nos dias de hoje são:

- Leis de potência – a distribuição gaussiana é muito comum na natureza, porém

existem muitos casos onde a probabilidade de um evento acontecer muito mais à direita ou esquerda da média é maior do que em uma distribuição gaussiana. Na Internet, por exemplo, muitos roteadores possuem um grau muito baixo (roteadores residenciais), enquanto poucos roteadores possuem um grau muito alto (roteadores do núcleo de um Backbone). Distribuição por leis de potência contempla esse modelo [31].

- Pequenos diâmetros – em meados da década de 60, Travers e Milgram conduziram um famoso experimento, onde era escolhido uma pessoa aleatoriamente e os indivíduos participantes teriam que enviar uma correspondência em cadeia até que a mesma chegasse as mãos do alvo escolhido. Eles constataram que o comprimento médio das correntes era de aproximadamente 6, um número muito pequeno levando-se em consideração o grande tamanho da população de participantes e indivíduos alvos. Tem-se que o diâmetro efetivo de um grafo é o número mínimo de hops (saltos) suficientes para que haja comunicação entre todos os pares de nós [11].

- Efeitos de comunidade – dentro do contexto de grafos, uma comunidade é geralmente considerada como um conjunto de nós, onde cada nó está mais relacionado a nós dentro da comunidade do que a nós de fora dela. Este efeito foi encontrado (ou acredita-se existir) em muitos grafos do mundo real, especialmente redes sociais [11].

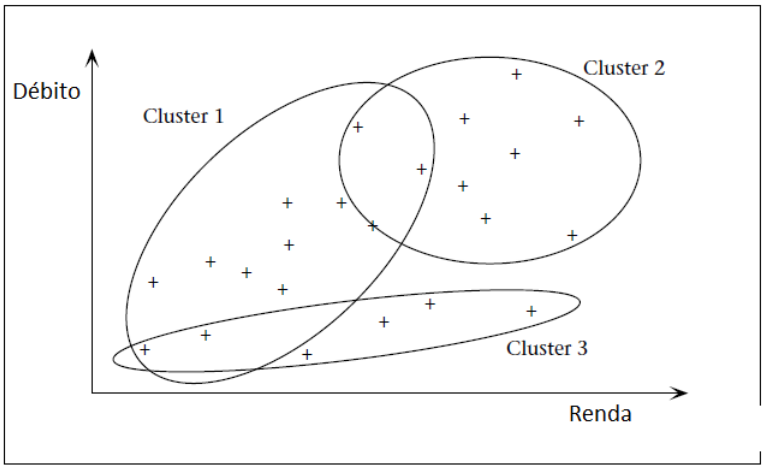
Outro campo que cresce nas pesquisas em mineração de grafos são os geradores de grafos, que são usados para criação de grafos sintéticos que podem, por exemplo, ser usados em estudos de simulação. Para que esses geradores sejam válidos eles precisam gerar grafos realísticos, ou seja, condizentes com a estrutura de grafos do mundo real. Um grafo para ser considerado realístico precisa corresponder a todos, ou pelo menos alguns dos padrões mencionados nos itens anteriores [11].

3.2.2 Clusterização

Clusterização (agrupamento) é a classificação não-supervisionada de padrões em grupos (*clusters*), onde padrões do mesmo *cluster* devem ser mais similares entre si em comparação a padrões de outro *cluster*, ou seja dado um determinado grupo de objetos, eles são divididos em grupos de objetos similares [32]. A figura 11 mostra um possível agrupamento de dados de empréstimo em 3 *clusters*. Vale ressaltar que os *clusters* se sobreponhem, ou seja, dessa forma um certo dado pode pertencer a mais de um cluster.

Algoritmos de clusterização têm uma significativa relevância em uma variedade de aplicações, como por exemplo processamento de imagens, reconhecimento de objetos

Figura 11: Exemplo de clusterização. Adaptada de: [2].



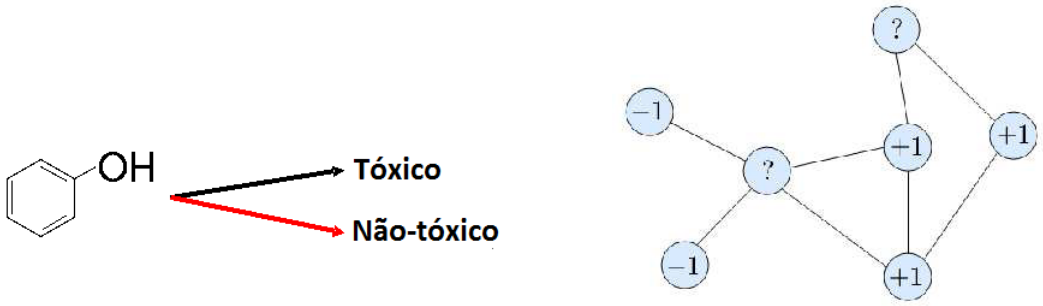
e recuperação de informação. Dentro do contexto de mineração de grafos, a clusterização pode ser de 2 tipos: algoritmos de clusterização de nós, onde nessa abordagem se tem um único grande grafo e tenta-se agrupar seus nós com a utilização de um valor de distância (similaridade) sobre suas arestas (em [33] são discutidos alguns algoritmos de clusterização de nós em grafos); e algoritmos de clusterização de grafos, neste caso temos um conjunto de grafos que necessitam ser agrupados baseados em seu comportamento estrutural [3].

A clusterização em bases de grafos pode ser utilizada em diversas aplicações, dentre as principais podemos citar, detecção de comunidades em aplicações WEB e redes sociais, redes de telecomunicação (onde os usuário são tratados como nós e podem ser agrupados de acordo com a duração de ligações por exemplo), e análise de email [3].

3.2.3 Classificação de grafos

Classificação é um tema central em mineração de dados e aprendizado de máquina, e no contexto de mineração de bases de grafos, classificação vem atraindo atenção tanto na academia como na indústria. O termo classificação pode significar duas tarefas distintas: a primeira chamada de classificação de grafos, que consiste na construção de um modelo para prever o rótulo da classe de um grafo como um todo e a segunda tarefa é chamada de propagação de rótulos, que analogamente à primeira consiste na construção de um modelo para prever o rótulo da classe de um nó dentro de um grafo [3]. A figura 12 mostra um exemplo de classificação de grafo e de propagação de rótulo, no grafo da esquerda a classificação de grafo é utilizada pra determinar se o composto químico é ou não tóxico, já no grafo da direita a propagação de rótulo pode ser usada pra definir os rótulos dos nós desconhecidos.

Figura 12: Classificação de grafos e propagação de rótulos. (Adaptada de [3].)



3.2.4 Mineração de Subgrafos frequentes

Grafos vêm ganhando cada vez mais importância na modelagem de estruturas complexas, tais como circuitos, imagens, compostos químicos, estruturas de proteínas, redes biológicas, documentos XML, redes sociais, WEB e fluxogramas. Muitos algoritmos de busca em grafos vêm sendo desenvolvidos nos campos da química, visão computacional, indexação de vídeo e recuperação de textos. Com o crescimento da demanda na análise de grandes conjuntos de dados estruturais, mineração de grafos vem ganhando importância nas pesquisas de mineração de dados [34].

Algoritmos eficientes para encontrar *itemsets* frequentes em bases de dados de transacionais tem sido um dos pontos fundamentais na história de sucesso das pesquisas em mineração de dados [35]. A descoberta de *itemsets* frequentes pode ser utilizada para a geração de regras de associação, extração de padrões predominantes que existem nas bases de dados, ou para a classificação. No entanto, as técnicas de mineração de dados vêm sendo cada vez mais aplicadas a domínios não-tradicionais (como por exemplo bases de dados científicas e geográficas), onde tendem a ocorrer casos em que não podemos aplicar os algoritmos de mineração de *itemsets* frequentes, pois estes problemas são de difícil tratamento e modelagem com as abordagens tradicionais de transações em carrinhos de compra[2]. Um caminho alternativo de se modelar esses itens de domínios não triviais, bem como a relação entre eles, é a utilização de grafos rotulados não orientados. Basicamente, cada vértice do grafo corresponderá a uma entidade e cada aresta corresponderá a uma relação entre duas entidades. Neste modelo, tanto vértices como arestas podem ter rótulos associados e não obrigatoriamente precisam ser únicos [37].

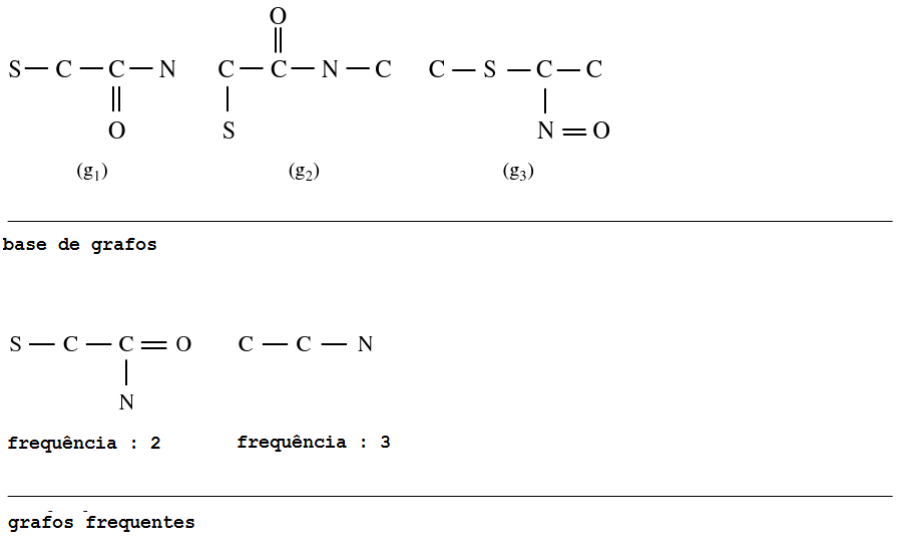
Entre os diversos tipos de padrões que podem ser encontrados em grafos, subes-

[2]Bancos de dados de carrinhos de compra organizam coleções de transações. Cada transação consiste de um conjunto de itens de um domínio especificado. Várias organizações armazenam grandes bancos de dados acerca de dados de carrinhos de compra, como por exemplo, empresas de varejo. Sites de ecommerce também vêm produzindo bastantes dados de carrinhos de compra. Estas transações podem ser representadas por produtos que foram comprados, páginas que foram visitadas, etc [36].

truturas frequentes (ou subgrafos frequentes) são um dos mais comuns. Subgrafos frequentes podem ser muito úteis para a caracterização de conjunto de grafos, discriminando diferentes grupos de grafos, classificação e clusterização de grafos, construção de índices e também pode facilitar a busca de similaridade em bases de dados de grafos [4].

Podemos definir que um subgrafo é frequente dentro de um outro grafo da seguinte forma: dado o conjunto de vértices de um grafo G por $V(G)$ e o conjunto de arestas por $E(G)$, um grafo G é subgrafo de outro G' se existe **isomorfismo** de G em G'. Dado um conjunto de grafos rotulados, $D = g_1, g_2, \ldots, g_n$, o **suporte(G)** (ou $frequencia(G)$) é definido como a porcentagem ou número de grafos em D onde G é subgrafo [38]. Diante disso, um **subgrafo frequente** é definido como um grafo no qual o seu suporte não é menor que o mínimo suporte estabelecido [4]. Para externar melhor esse conceito de subgrafo frequente, a figura 13 mostra um conjunto de dados de uma estrutura química e a extração de 2 subgrafos frequentes, dado o mínimo suporte igual a $66,6\%$.

Figura 13: Extração de subgrafos frequentes (traduzido de [4]).



Para se encontrar subgrafos frequentes numa dada base, basicamente são necessários dois passos: primeiramente são gerados candidatos a subgrafos frequentes e, na segunda etapa, a frequência dos candidatos gerados é checada. Atualmente, muitos estudos focam na otimização da primeira etapa, uma vez que a segunda etapa envolve teste de isomorfismo de subgrafos, tarefa essa que possui custo computacional elevadíssimo (NP-completo[3]), e dessa forma diminui-se o número de candidatos a subgrafos frequentes

[3]Um problema de decisão X é NP-completo se X está em NP e qualquer outro problema em NP pode ser polinomialmente reduzido a X. Informalmente, X é NP-completo se for tão difícil quanto qualquer outro problema em NP.

que precisam ser checados [39, 40]. De uma forma geral, existem duas abordagens básicas para o problema de mineração de subgrafos frequentes: a abordagem *apriori* e a abordagem *pattern-growth*.

A abordagem baseada *apriori* inicia a busca por subgrafos frequentes com grafos de tamanho pequeno e procede de uma forma *bottom-up*, onde a cada iteração o tamanho dos novos subgrafos descobertos é incrementado em 1. Esses novos subgrafos são gerados pela junção de 2 subgrafos frequentes que já foram descobertos, e logo após isso a frequência deles é checada e comparada ao suporte mínimo. A figura 14 mostra o algoritmo de mineração de grafos *apriori*. Os mais comuns algoritmos para mineração de subgrafos frequentes *apriori* são: o *Apriori-based Graph Mining* (AGM) , o *Frequent SubGraphs* (FSG) e o *Edge-disjoint Path-join*.

Figura 14: Algoritmo AprioriGraph. Traduzido de: [4].

Algoritmo: AprioriGraph.

Entrada:

- D, uma base de grafos.
- min_sup, o mínimo suporte estabelecido.

Saída:

- S_k, Conjunto de subestruturas frequentes.

Método:

$S_1 \leftarrow$ Elementos simples (tamanho 1) frequentes na base de dados.
chamar AprioriGraph(D, min_sup, S_1);

procedimento AprioriGraph(D, min_sup, S_k)

(1) $S_{k+1} \leftarrow \varnothing$;
(2) **para cada** frequente $g_i \in S_k$ **faça**
(3) **para cada** frequente $g_j \in S_k$ **faça**
(4) **para cada** grafo g de tamanho (k+1) formado pela junção de g_i e g_j **faça**
(5) **if** g é frequente
(6) insira g em S_{k+1};
(7) **se** $s_{k+1} \neq \varnothing$ **então**
(8) *AprioriGraph*(D, min_sup, S_{k+1});
(9) **retorne;**

Os algoritmos de mineração baseados em *pattern-growth* estendem um subgrafo frequente de uma dada base adicionando uma nova aresta em todas as posições possíveis, e para cada novo subgrafo frequente encontrado nessa extensão o processo é executado novamente de forma recursiva. A figura 15 mostra o algoritmo de mineração de grafos baseado em *pattern-growth*. Dentre os principais algoritmos de mineração de grafos baseados em *pattern-growth* temos: *Molecule Fragment Miner* (MoFa) , *Fast Frequent Subgraph Mining* (FFSM) , *SPanning tree based maximal graph mINing* (SPIN) , *GrAph Sequence Tree extractiON* (Gaston) e *graph-based Substructure pattern*

(gSpan) .

Figura 15: Algoritmo PatternGrowthGraph. Traduzida de: [4].

Algoritmo: PatternGrowthGraph.

Entrada:

- g, um grafo frequente;
- D, uma base de grafos;
- min_sup, mínimo suporte estabelecido.

Saída:

- O conjunto de grafos frequentes, S.

Método:

$S \leftarrow \varnothing$;
chamar PatternGrowthGraph(g, D, min_sup, S);

procedimento PatternGrowthGraph(g, D, min_sup, S)

(1) **se** $g \in S$ **então retorne;**
(2) **senão** insira g em S;
(3) varra D, e encontre todas arestas e que possam ser extendidas para $g \diamond_x e$;
(4) **para cada** frequente $g \diamond_x e$ **faça**
(5) $PatternGrowthGraph(g \diamond_x e, D, min\_sup, S)$;
(6) **retorne;**

A seguir serão abordadas informações adicionais sobre os algoritmos de mineração de grafos supracitados.

AGM – o algoritmo AGM utiliza um método de geração de candidatos baseado em vértice, que aumenta o tamanho da subestrutura em 1 vértice a cada iteração do algoritmo [41]. Dois grafos frequentes de tamanho-k são agregados somente se eles têm o mesmo subgrafo de tamanho(k-1), nesse caso considera-se o tamanho do grafo como o número de vértices do mesmo. O novo candidato gerado inclui o subgrafo de tamanho(k-1) em comum e 2 vértices adicionais dos padrões agregados. A figura 16 mostra esse processo de geração de candidatos à subgrafo frequente no algoritmo AGM. Após esse processo de geração de candidatos, o suporte deles é checado e o processo se repete até que não sejam mais encontrados subgrafos frequentes [4].

FSG – o algoritmo FSG utiliza um método de geração de candidatos baseado em aresta, que aumenta o tamanho da subestrutura em 1 aresta a cada iteração do algoritmo [42]. Dois padrões de tamanho-k são mesclados se e somente se eles compartilham o mesmo subgrafo contendo k-1 arestas, que é denominado de núcleo. No FSG o tamanho do grafo é referente ao número de arestas contido nele. O novo candidato a subgrafo frequente gerado inclui o núcleo e mais 2 arestas dos padrões mesclados. A

Figura 16: AGM: Geração de candidatos à subgrafo frequente. Fonte: [4].

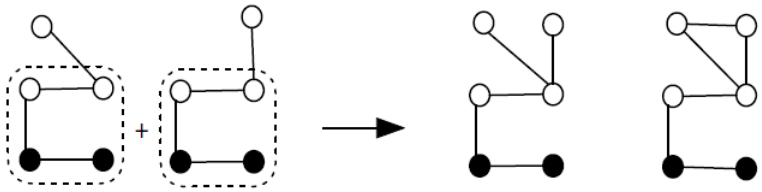
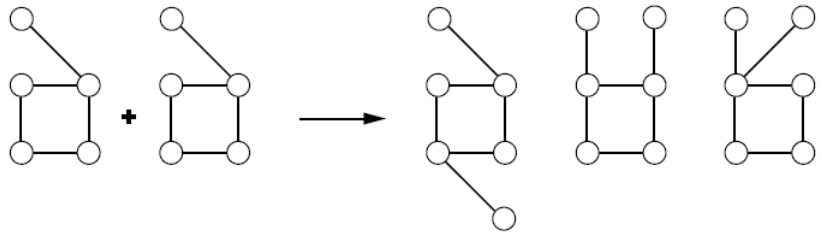


figura 17 mostra esse processo de geração de candidatos à subgrafo frequente no algoritmo FSG. Após gerados os candidatos, o suporte deles é checado e o processo se repete até que não sejam mais encontrados subgrafos frequentes [4].

Figura 17: FSG: Geração de candidatos à subgrafo frequente. Fonte: [4].



gSpan – a abordagem *pattern-growth* é bem simples, porém muitas vezes pode ser ineficiente uma vez que o mesmo subgrafo pode ser descoberto várias vezes, o que compromete o desempenho computacional do algoritmo [4]. O algoritmo gSpan surgiu com o propósito de reduzir a geração de grafos duplicados, onde ele não estende grafos duplicados e ainda garante a descoberta completa dos subgrafos frequentes. O gSpan utiliza busca em profundidade (*depth-first search* (DFS)) na extração de subgrafos frequentes. O gSpan introduz duas técnicas, *DFS lexicographic order* e *minimum DFS code*, que formam um novo sistema de rotulação canônica, que dá suporte ao processo de busca em profundidade [43].

Edge-disjoint Path-join – no algoritmo *Edge-disjoint Path-join* os grafos são classificados pelo número de caminhos disjuntos que eles têm, e 2 caminhos são *edge-disjoint* se eles não compartilham nenhuma aresta em comum [44]. Um candidato a subgrafo frequente com k+1 caminhos disjuntos é gerado pela junção de 2 subestruturas com k caminhos disjuntos [34].

MoFa – a maioria das tarefas de mineração de dados em bioinformática na análise de grandes conjuntos de moléculas com o objetivo de encontrar alguma regularidade (padrão) entre as moléculas de uma classe específica, o MoFa é algoritmo de mineração que foi desenvolvido com esse intuito. Apesar do MoFa ter sido concebido para encontrar padrões em bases de dados moleculares, ele também pode ser utilizado para bases de grafos arbitrárias [45].

FFSM – o FFSM aplica um esquema de busca vertical para reduzir o número de candidatos a subgrafo frequente redundantes, em testes realizados em bases de grafos sintéticas e reais foi demonstrado que o FFSM se mostrou competitivo em termos de performance comparado ao gSpan [46].

SPIN – em uma base de dados de grafos de grande porte o número de subgrafos frequentes pode ser tão grande de modo que não permita sua listagem completa utilizando recursos computacionais razoáveis. Visando contornar essa dificuldade mencionada anteriormente, o SPIN minera somente subgrafos de frequência máxima, ou seja, subgrafos que não são parte de nenhum outro subgrafo frequente. Essa estratégia adotada pelo SPIN diminui o tamanho do arquivo de saída (arquivo contendo os padrões minerados) em cerca de 2 ou 3 vezes [47].

Gaston – normalmente os algoritmos de mineração de grafos buscam por todas subestruturas que satisfaçam restrições, como por exemplo o suporte mínimo [48]. Para tornar o processo de mineração de subgrafos frequentes mais eficiente, o Gaston realiza uma busca em etapas com incremento de complexidade. O Gaston primeiramente busca por caminhos frequentes, depois por árvores livres frequentes e finaliza a busca encontrando grafos cíclicos [49].

4 Ontologias

A palavra ontologia tem origem da filosofia, onde significa uma explanação sistemática do ser (da existência) [50]. Nas últimas décadas a palavra ontologia vêm ganhando importância dentro da ciência da computação, especialmente na engenharia de conhecimento. Dentro do contexto da computação, ontologia possui diversas definições. Uma das mais citada foi proposta em [51, 52], e diz que uma ontologia é uma especificação explícita de uma conceitualização, que é uma visão abstrata e simplificada do mundo que desejamos representar para algum propósito. Segundo NOY e MCGUINESS (2000), a utilização de ontologias pode ser útil para compartilhar conhecimento comum da estrutura da informação entre pessoas ou agentes de software, permitir a reutilização do conhecimento do domínio, fazer suposições de domínio explícito, separar conhecimento de domínio de conhecimento operacional, bem como analisar o conhecimento de domínio, conforme segue:

- Compartilhar conhecimento comum da estrutura da informação entre pessoas ou agentes de software – suponhamos que diversos web sites possuam informações do domínio médico por exemplo, e esses web sites compartilham e publicam dados de uma mesma ontologia, dessa forma agentes de software podem extrair e agregar informações a partir desses diferentes sites. Essas informações compartilhadas podem servir para responder consultas de usuários ou como entrada de dados para outras aplicações por exemplo.

- Permitir a reutilização do conhecimento do domínio – um dos carros-chefe do crescimento nas pesquisas em ontologias. Por exemplo, diferentes domínios precisam representar a noção do tempo, representação essa que inclui conceitos de intervalos de tempo, pontos no tempo, medidas relativas de tempo, e assim por diante. Se um grupo de pesquisadores desenvolver tal ontologia em detalhe, outros poderão simplesmente reutilizá-la para seus domínios.

- Fazer suposições explícitas de domínio – Com suposições explícitas de domínio a manutenção da aplicação se torna mais fácil uma vez que não é necessário conhecimento profundo em linguagem de programação para manipulação e alteração dos dados na ontologia. Outro aspecto positivo em se fazer suposições explícitas de domínio é o fato de ser úteis para novos usuários que precisem aprender o significado dos termos de um domínio.

- Separar conhecimento de domínio de conhecimento operacional – Por exemplo, temos uma ontologia que descreve uma tarefa de configuração de algum equipamento eletrônico em geral, um algoritmo externo utiliza dados dessa ontologia

pra implementar essas configurações. Com a utilização da ontologia esse mesmo algoritmo pode ser utilizado tanto para configurar um PC ou um celular por exemplo.

- analisar o conhecimento de domínio – Uma ontologia por si só não possui muita relevância, ou seja, ela é construída para outros programas poderem utilizá-la. Por exemplo suponhamos que temos uma ontologia de alimentos e temos uma aplicação para sugestão de cardápios. Esta aplicação pode analisar os dados de cardápios presentes na ontologia para gerar uma resposta para a consulta de um usuário da aplicação.

4.1 Componentes de uma Ontologia

Ontologias podem ser utilizadas em diversas áreas, podendo conceitualizar diferentes domínios de interesse. Isso faz com que as ontologias apresentem estruturas que as distinguam umas das outras. Porém existem componentes básicos que compõem a maioria das ontologias para representação de conhecimento [51, 53]:

Classes – São coleções de elementos formados por um determinado conjunto de atributos. Formam a unidade básica de uma ontologia e também podem ser tratadas como conceitos que definem um determinado objeto;

Relações – São responsáveis pelos relacionamentos semânticos entre os conceitos de um dado domínio, ou seja, representam um tipo de interação entre as classes do domínio. Elas irão formar a taxonomia do domínio;

Axiomas – são as regras declaradas sobre as relações, as quais devem ser cumpridas pelos elementos da ontologia, ou seja, são sempre verdadeiras. Elas possibilitam inferir conhecimento que não esteja explícito nas taxonomias da ontologia;

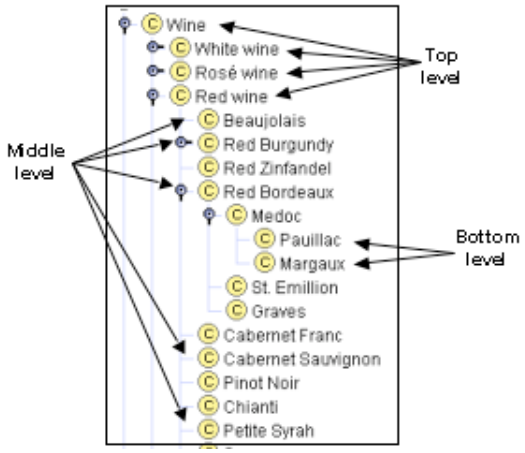
Instâncias – representam um determinado objeto de uma classe, ou seja, são os próprios dados da ontologia.

A figura 18 mostra uma ontologia de vinhos, onde uma classe de vinhos representa todos os vinhos. Exemplo de uma instância é uma garrafa de vinho *Bordeaux*, que é uma instância da classe *Red Bordeaux*. Uma classe pode possuir uma especificação, onde a subclasse possuirá conceitos mais específicos em relação à sua classe pai. No exemplo, as subclasses são *White*, *Rose* e *Red*. As subclasses herdam todos os atributos de suas super classes [53].

4.2 Tipos de Ontologia

Segundo [6], as ontologias podem ser divididas em 4 tipos de acordo com seu nível de generalidade:

Figura 18: Exemplo de uma ontologia de vinhos. Fonte: [5].



- Ontologias de alto nível ou genéricas – descrevem termos mais gerais, como por exemplo espaço, tempo, matéria, objeto, evento e ação. São independentes de um problema ou domínio particular;

- Ontologias de Domínio – descrevem o vocabulário relacionado a um domínio genérico (como medicina ou automóveis).

- Ontologias de Tarefa – descrevem uma tarefa ou atividade genérica (como diagnósticos ou vendas), especializando os termos introduzidos na ontologia de alto nível;

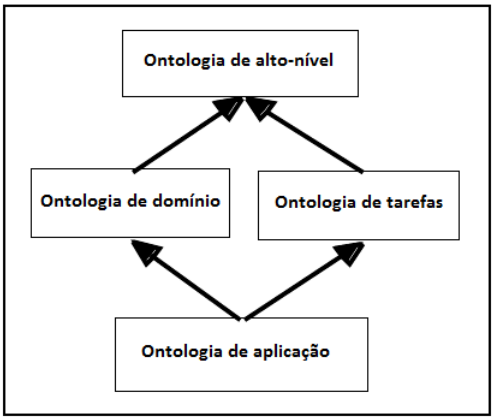
- Ontologias de aplicação – descrevem conceitos dependentes do domínio e de tarefas particulares. Estes conceitos, frequentemente, correspondem a papéis desempenhados por entidades de domínio, quando da realização de uma determinada atividade. Por exemplo, uma ontologia de um hospital, que pode ter um médico da ontologia de domínio "medicina"ligado a uma tarefa como "diagnóstico"de uma ontologia de tarefas.

A figura 19 mostra o relacionamento entre os tipos de ontologia, lembrando que as setas representam as relações de especialização.

4.3 Linguagens para manipulação de Ontologias

Existem várias linguagens para criação e manipulação de ontologias. Em [54] são citadas algumas das mais conhecidas: CycL [55], Frame Logic (Flogic) [56], Loom [57] e PowerLoom [58], CARIN [59], GALEN Representation And Integration Language (GRAIL) [60], Operational Conceptual Modeling Language (OCML) [61], Ontology

Figura 19: Tipos de ontologias. Traduzido de [6].



Markup Language (OML) [62], Resource Description Framework (RDF) [63], RDF Schema (RDFS) [64], Narrative Knowledge Representation Language (NKRL) [65], Simple HTML Ontology Extensions (SHOE) [66], Ontology Interchange Language (OIL) [67], DARPA Agent Markup Language (DAML) + OIL [68], e Web Ontology Language (OWL) [69]. As linguagens RDF, RDFS e OWL tiveram um maior destaque dentre as outras nos últimos anos, por isso serão explanadas nas subseções a seguir.

4.3.1 RDF e RDFS

RDF é um modelo padrão para intercâmbio de dados na WEB. RDF estende a estrutura de links da WEB utilizando URIs para nomear o relacionamento entre as coisas, bem como as 2 extremidades dos links. Esse modelo permite que dados estruturados e semi-estruturados possam ser combinados, expostos e compartilhados entre aplicações diferentes. RDF vem sendo muito utilizado para: prover informações sobre recursos WEB e sistemas que a utilizam servindo de meta-dados, construção de aplicações que requerem modelo de informações aberto ao invés de restrito, tornar as informações processáveis por máquinas, interoperabilidade entre aplicações (ou seja, combinando dados de diversas aplicações para se chegar em novas informações) e processamento automatizado de informações WEB por agentes de software [63].

O RDF-S é uma extensão semântica do RDF, que fornece mecanismos para descrever grupos de recursos e relacionamentos entre esses recursos [64].

4.3.2 OWL

A expressividade de RDF e RDF-S é um tanto quanto limitada. Diante disso, o W3C identificou características para ontologias na WEB que requerem muito mais expressividade que RDF e RDF-S. Grupos de pesquisa da América e Europa já haviam

identificado a necessidade de uma linguagem de modelagem de ontologias mais poderosa, o que levou a uma iniciativa conjunta na criação de uma linguagem mais rica, chamada DAM+OIL (este nome é a junção da linguagem americana DAML-ONT com a linguagem europeia OIL). O DAML+OIL foi o ponto de partida para o grupo de trabalho em ontologias WEB da W3C na criação do OWL, que tem como objetivo ser padronizado e amplamente aceito na WEB semântica [70].

OWL foi projetado pra ser usada por aplicações que necessitam processar o conteúdo das informações e não apenas exibi-las para humanos. OWL facilita a interpretação do conteúdo da WEB por máquinas em relação as outras linguagens pelo fato de fornecer um vocabulário adicional com uma semântica formal [69]. O OWL é dividido em três sub-linguagens, classificadas por seu grau de expressividade:

- OWL *Full* – utiliza todas as primitivas da linguagem OWL, e tem como grande vantagem ser totalmente compatível com RDF, tanto sintática quanto semanticamente, onde um documento RDF válido é também um documento OWL *Full* válido. O fato de possuir o máximo da expressividade do OWL traz algumas desvantagens para o OWL *Full*, como por exemplo, alto custo computacional de processamento, o que torna improvável que algum software de raciocínio seja capaz de suportar todos os recursos do OWL *Full* [69].

- OWL DL – de modo a recuperar a eficiência computacional, OWL DL é a sublinguagem de OWL que restringe a forma como os construtores de OWL e RDF são utilizados, não dando suporte à completa expressividade da linguagem, o que permite um eficiente suporte a softwares raciocinadores. A desvantagem dessa abordagem é que OWL DL não é totalmente compatível com RDF [70].

- OWL *Lite* – é uma versão bem mais restrita da OWL, onde muitas funcionalidades não estão disponíveis. A vantagem da OWL *Lite* é que tanto seu entendimento como implementação são mais fáceis, e a desvantagem é a expressividade muito restrita para ontologias [70].

Uma ontologia OWL consiste de indivíduos, propriedades e classes. Os indivíduos representam os objetos do domínio de interesse e são instâncias das classes OWL. Vale ressaltar que OWL não utiliza *Unique Name Assumption (UNA)*. Dessa forma, dois ou mais diferentes nomes podem se referir ao mesmo indivíduo. "Fenômeno"e "Ronaldo", por exemplo, podem se referir ao mesmo indivíduo, o jogador de futebol mundialmente conhecido Ronaldo Nasário [71]. Em OWL as instâncias de classes são declaradas como em RDF:

```
<rdf:Description rdf:ID="156">
```

```
    <rdf:type rdf:resource="#professor"/>
</rdf:Description>
```

ou equivalentemente:

```
<professor rdf:ID="156"/>
```

Em owl existem dois tipos de propriedades:

- Propriedades de objeto – são relações binárias sobre indivíduos. Como por exemplo uma propriedade "lecionadaPor", que liga uma instância da classe disciplina a uma instância da classe Professor [70]. A seguir o código OWL para a propriedade "lecionadaPor":

```
<owl:ObjectProperty rdf:ID="lecionadaPor">
    <owl:domain rdf:resource="#disciplina"/>
    <owl:range rdf:resource="#professor"/>
</owl:ObjectProperty>
```

- Propriedades de tipo de dados – relacionam objetos à *datatype values*, como por exemplo, telefone e idade. OWL não possui nenhum tipo de dados predefinido, por isso provê facilidades pra essa definição de tipos de dados, onde normalmente esses tipos são importados de um XMLSchema para serem usados na ontologia OWL [70]. A seguir um exemplo da propriedade idade, definida como inteiro não negativo:

```
<owl:DatatypeProperty rdf:ID="idade">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XLMSchema
    #nonNegativeInteger"/>
</owl:DatatypeProperty>
```

Classes OWL são interpretadas como conjuntos que contêm indivíduos, como por exemplo, a classe gerente deve conter todos os indivíduos que são gerentes naquele determinado domínio de interesse. As classes podem ser organizadas em hierarquias de superclasse-subclasse, conhecido também como taxonomia. Subclasses são classificados por sua superclasse, tomando-se por exemplo a classe "gerente"e "funcionario", onde "gerente"é subclasse de "funcionario"(logo "funcionario"é superclasse de gerente). Isso implica dizer que "todo gerente é funcionário", ou seja todo indivíduo da classe "gerente"pertence a classe "funcionario". Dessa forma um "gerente"é classificado como um "funcionario"[71]. A seguir o código OWL para a classe "gerente"sendo subclasse de "funcionario":

```
<owl:Class rdf:ID="gerente">
    <rdfs:subClassOf rdf:resource="#funcionario"/>
</owl:Class>
```

4.4 Ferramentas

Por se tratar de uma tarefa dispendiosa, a utilização de ferramentas de apoio na contrução de ontologias traz ganhos significativos. Dentre as ferramentas para a manipulação e construção de ontologias podemos citar: Conceptually Oriented Description Environment (CODE4) [72], Ontolingua [73], Generic Knowledge Base Editor (GKB-editor) [74], Java Ontology Editor (JOE) [75], Web Ontology Design Environment (WebODE) [76], WebOnto [77], Asium [78], Jena [79] , *Protégé* [80], dentre outras. Nas próximas subseções alguns detalhes sobre Jena e *Protégé* serão abordados, uma vez que foram as tecnologias utilizadas neste trabalho no que diz respeito a manipulação de ontologias.

4.4.1 Jena

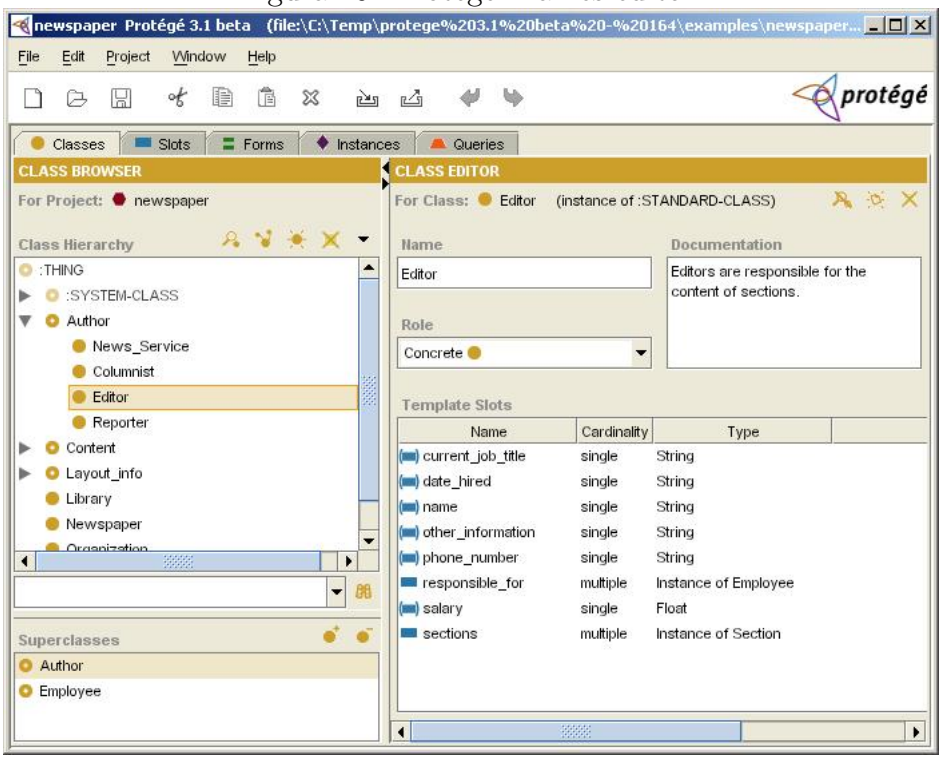
Jena é uma API Java para aplicações na WEB semântica, que permite a manipulação de ontologias RDF, RDFS e OWL, é uma iniciativa da Hewlett-Packard (HP) [81, 82]. O JENA possui código aberto e disponibiliza várias classes Java para manipulação de ontologias OWL. Dentre as principais podemos destacar: *OntModel*, *OntClass*, *OntProperty* e *Individual* [83, 84]. De uma forma simples, podemos dizer que o Jena transforma uma dada ontologia em um modelo abstrato de dados orientado a objetos e permite que suas primitivas (classes, relacionamentos, axiomas e instâncias) possam ser tratadas como objetos [85].

4.4.2 Protégé

Protégé é uma plataforma gratuita e de código aberto que fornece um conjunto de ferramentas para construção de modelos de domínio e aplicações baseadas em conhecimento com o uso de ontologias. O Protégé é baseado em Java, é extensível (por meio de uma arquitetura de plug-in) e fornece um ambiente *plug-and-play* que o torna bastante flexível para prototipagem rápida como para o desenvolvimento de aplicações [80]. O Protégé suporta 2 formas de modelagem de ontologias: O *Protégé-Frames editor* e o *Protégé-OWL editor*.

- ***Protégé-Frames editor*** - Permite criar e popular ontologias baseadas em frame, de acordo com o protocolo *Open Knowledge Base Connectivity Protocol* (OKBC) . Nesta abordagem baseada em frames, uma ontologia consiste em um conjunto de classes organizadas em uma hierarquia para representação de importantes conceitos de um domínio, um conjunto de *slots* associados às classes para descrever suas propriedades e relacionamentos, e por um conjunto de instâncias dessas classes [86]. A figura 20 apresenta uma interface do *Protégé-Frames editor*.

Figura 20: Protégé-Frames editor.



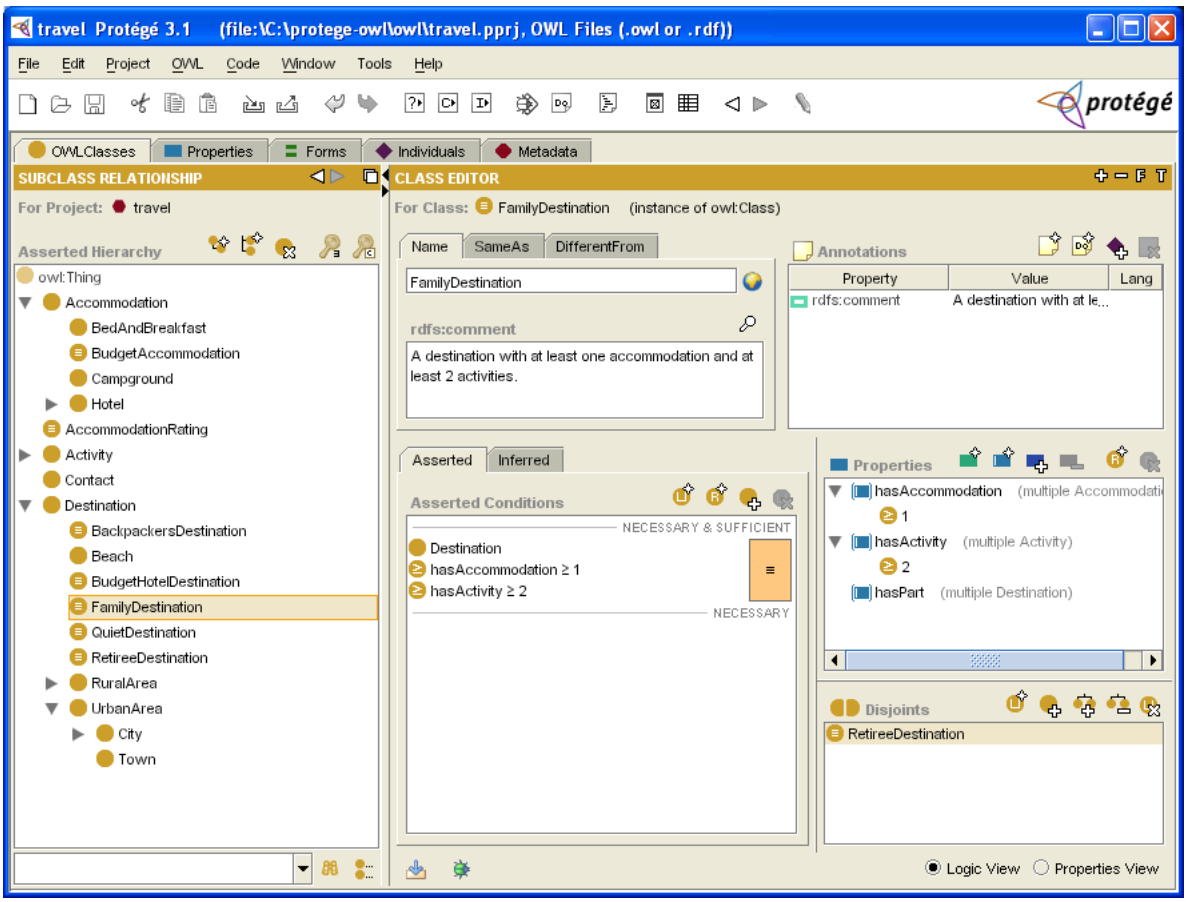
- ***Protégé-OWL editor*** - Possibilita a criação de ontologias para web semântica. Através do Protégé-OWL editor é possível a manipulação de ontologias em OWL e RDF [87]. A figura 21 apresenta a interface do *Protégé-OWL editor*.

4.5 Ontologias e Regras

Nos últimos anos, Sistemas baseados em regras vem despertando bastante interesse no contexto da WEB semântica, sendo principalmente utilizados para expressar a lógica de aplicações em termos de regras. Sistemas baseados em regras tomam como entrada um estímulo externo (o acontecimento de algum evento) e reagem (pelo acionamento das regras) produzindo alguns efeitos. A relação causal entre fatos observados (entrada) e os efeitos observados (resultados) é definida pelas regras que o usuário especifica como uma base de conhecimento [88].

OWL foi desenvolvida como uma linguagem para criação de ontologias, provendo descrição de alto-nível ao conteúdo WEB, porém nem sempre é possível expressar tudo que se quer em linguagem OWL. Recentes trabalhos tem se concentrado em adicionar regras ao OWL para inserir uma camada adicional de expressividade, e a Semantic Web Rule Language (SWRL) é um dos resultados dessas pesquisas [89].

Figura 21: Protégé-OWL editor.



4.5.1 SWRL

SWRL possui, em comum com outras linguagens de processamento de regras, sua escrita no formato antecedente e consequente. O antecedente é referido como o **corpo** da regra e o consequente como a **cabeça**. A cabeça e o corpo consistem de conjunções de um ou mais átomos [90]. Quando as condições especificadas no corpo forem satisfeitas, então as condições especificadas no consequente também são ditas como verdadeiras [91].

Regras SWRL podem raciocinar a cerca de indivíduos de uma ontologia OWL, levando-se em consideração as definições de classes e propriedades da ontologia. Em uma regra expressando que um homem é avô de uma pessoa, por exemplo, são necessários a captura dos conceitos de "Pessoa", "Masculino", "Pai"e "Avô". De forma intuitiva, os conceitos Pessoa e Masculino podem ser recuperados usando uma classe OWL chamada *Pessoa* com uma subclasse chamada *Homem* e o conceito Pai e Avô podem ser relacionados as propriedades *temFilho* e *AvoDe* respectivamente, que ligam

elementos da classe *Pessoa* [90]. Essa regra expressa em SWRL ficaria da seguinte forma:

```
temFilho(?x,?y) ^ temFilho(?y,?z) ^ Homem(?x) -> AvoDe(?x,?z)
```

Isto significa que sempre quando houver uma pessoa x que seja pai de uma pessoa y e essa pessoa y tenha como filho o indivíduo z, isto implica que x é avô de z.

4.5.2 SQWRL

SWRL é uma linguagem de regras e não uma linguagem de consulta. No entanto, muitas aplicações baseadas em ontologia requerem a capacidade de extrair informações dessas ontologias. Para tal tarefa de extração de informações de ontologias OWL foi desenvolvido o Semantic Query-Enhanced Web Rule Language (SQWRL) , que é uma linguagem para consultas em ontologias OWL baseada em SWRL [89].

SQWRL trata o antecedente de uma regra SWRL como uma especificação padrão para uma consulta e o consequente da regra é tratado como uma especificação padrão do que se deseja recuperar. O principal operador SQWRL é o *sqwrl:select*, que manipula um ou mais argumentos que são normalmente variáveis usadas na especificação da consulta e retorna uma tabela como resultado [92]. Por exemplo, uma consulta que visa recuperar todas as instâncias de pessoas do sexo masculino de determinada ontologia, poderia ser escrita da seguinte forma:

```
Pessoa(?x,) ^ Homem(?x) -> sqwrl:select(?x)
```

Dessa forma, o comando *sqwrl:select(?x)* retorna todas as instâncias que satisfaçam os critérios da consulta, onde a variável x será instanciada por cada pessoa do sexo masculino da ontologia.

5 Enriquecimento Semântico de Padrões Minerados em Grafos

A maioria dos trabalhos desenvolvidos no âmbito da mineração de grafos foca no desenvolvimento de algoritmos de mineração eficientes para a descoberta de padrões frequentes, até mesmo porque, pelo fato de ser uma área relativamente nova, ainda há espaço para se investir na construção de novos algoritmos bem como na melhoria dos algoritmos já existentes. Uma base de dados de grafos é geralmente modelada em formatos específicos, na maioria das vezes baseados em formalismos matemáticos de representação como listas de adjacência, matriz de adjacência, matriz de incidência ou similares a uma destas representações. A visualização dos padrões minerados também se dá nesse formato matemático utilizado para a modelagem da base de dados de grafos, o que traz consigo alguns encargos no que tange à interpretação e entendimento dos padrões, dificuldade essa que vai crescendo à medida que se aumenta o tamanho do repositório a ser minerado. Para exemplificar melhor essa dificuldade na interpretação dos padrões podemos citar o exemplo mostrado em [93], onde uma base de grafos de Antivirais para AIDS com cerca de 400 componentes que foi executado com suporte igual a 5% e gerou algo em torno de 1 milhão de padrões de grafos.

Este trabalho tem como objetivo o desenvolvimento de uma arquitetura capaz de extrair informações semânticas agregadas aos padrões de grafos minerados. Tais informações são obtidas através do uso de ontologias OWL enriquecidas por regras SWRL. Assim sendo, a abordagem proposta trará avanços relevantes à mineração de grafos, como por exemplo, uma interpretação mais amigável dos padrões minerados.

5.1 Trabalhos Relacionados

5.2 Semantic Graph Mining

A proposta de [94] apresenta a criação de uma metodologia, denominada *Semantic Graph Mining*, para extração assistida de regras, com o objetivo de extrair padrões em grafos a partir de instruções semânticas consolidadas. Inicialmente o grafo semântico é tratado como um grafo dirigido, então é aplicado sobre o grafo algoritmos de mineração de grafos, como o de extração de subgrafos frequentes por exemplo. Em seguida é realizada a inferência de conhecimento na descoberta de padrões e regras. Por fim, as regras candidatas são apresentadas aos especialistas de domínio para uma análise, para em seguida serem utilizadas.

5.3 Taxogram

O Taxogram [95] é um algoritmo de mineração de taxonomia de grafos sobrepostos que pode descobrir eficientemente estruturas frequentes em um banco de dados de taxonomia de grafos sobrepostos. Ele apresenta duas vantagens, primeiramente executa um teste de isomorfismo no subgrafo frequente por classes de padrões que são estruturalmente isomórficas, mas que possuem identificações (nomes) diferentes, e como segunda vantagem do Taxogram, ele concilia métodos de mineração de grafos com a taxonomia de mineração de grafos e tira vantagem de métodos bem estudados na literatura. Basicamente o Taxogram possui 3 estágios: primeiramente há a re-rotulação dos nós no banco de dados de entrada; em seguida a mineração dos padrões de classes e a construção do índice de ocorrência associados; e por fim é feito a computação dos padrões e a eliminação dos padrões irrelevantes pelo pós-processamento dos índices de ocorrência.

5.4 Web semântica aplicada a detecção de conflitos de interesse

Em [96, 97] é apresentada uma aplicação da Web semântica para detecção de *Conflict of Interest* (COI) nas relações entre os potenciais revisores e autores de trabalhos científicos. Essa aplicação descobre várias "associações semânticas"entre os revisores e autores em uma ontologia populada para determinar o grau de COI. A fim de demonstrar a abordagem de detecção COI, foram utilizados dados de duas redes sociais: a *friend-of-a-friend* (FOAF) e a rede de co-autores da DBLP. Um componente de visualização gráfica provê ao usuário uma visualização dos participantes (e seus relacionamentos) de um COI detectado. Para a avaliação da eficácia da técnica proposta foi utilizado um subconjunto de artigos e revisores do *2004 International World Wide Web Conference*.

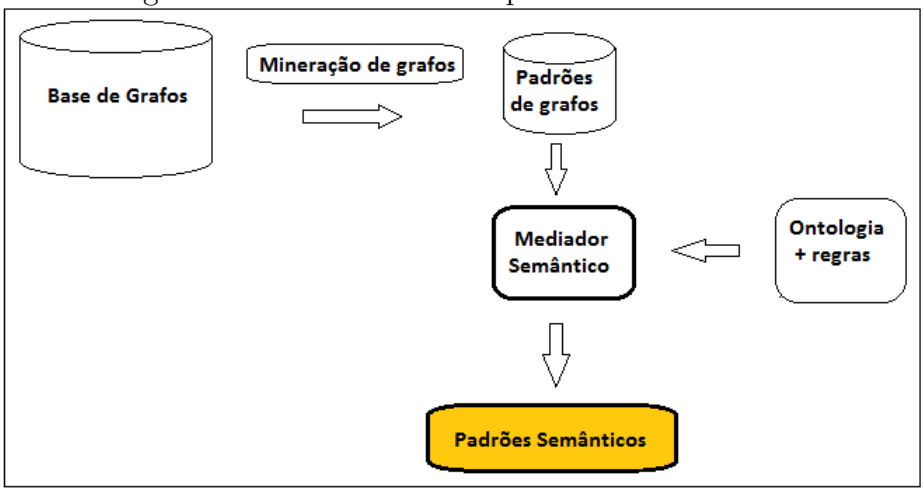
5.5 Anotação semântica de padrões frequentes

Nos trabalhos [9, 98] é abordado o problema da geração anotações semânticas para padrões frequentes. O estudo objetivou descobrir o significado oculto dos padrões frequentes minerados em grafos. O trabalho propõe uma abordagem geral para realizar a geração de uma anotação para um padrão frequente e assim construir o modelo de contexto, realizando a seleção de indicadores de contexto informativo e a extração de transações representativas de padrões semanticamente similares, além de poder incorporar o conhecimento prévio do usuário.

5.6 Arquitetura Proposta

Trabalhos relacionados que envolvem mineração de grafos e recursos semânticos não contemplam a utilização de ontologias para aperfeiçoar a interpretação dos padrões de grafos minerados através da associação de recursos ontológicos. A arquitetura proposta nesse trabalho visa agregar valor semântico aos padrões numa mineração de grafos, utilizando para tal ontologias como âncora semântica, uma vez que ontologia é uma tecnologia que permite o compartilhamento de conhecimento, bem como a reutilização desse conhecimento. A arquitetura é mostrada na figura 22, cujo núcleo é o mediador semântico, responsável por todo o processamento realizado no método de enriquecimento semântico proposto. O mediador semântico foi implementado em linguagem JAVA, e como mostrado na figura 23 o mediador semântico é constituído de 3 módulos principais: o módulo de leitura, responsável pela leitura dos dados de entrada; módulo de processamento, que faz a integração entre os padrões e a ontologia a fim de agregar informações de contexto aos padrões; e o módulo de visualização, responsável pela exibição dos padrões semanticamente enriquecidos. Guiada pela própria estrutura de componentes do mediador, o processo de enriquecimento semântico consiste em 3 fases: leitura e mapeamento dos dados de entrada (padrão minerado e a ontologia de domínio), processamento dos dados mapeados, e exibição dos Padrões minerados enriquecidos com informações de contexto.

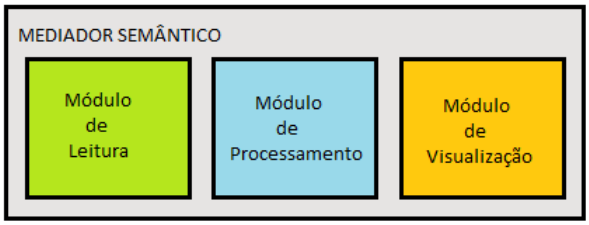
Figura 22: Processo de Enriquecimento Semântico



5.6.1 Leitura e Mapeamento dos Dados de Entrada

Na maioria das implementações de mineradores de subgrafos frequentes existentes, os padrões minerados são representados em arquivos de texto simples com uma estrutura baseada em modelos matemáticos de representação de grafos. O minerador de

Figura 23: Mediador Semântico



grafos utilizado nos nossos experimentos foi uma implementação do FSG disponível no PAFI [99]. Essa implementação do FSG foi utilizada em nossos experimentos por conseguir manipular vértices e arestas rotuladas com *string*, o que facilitou a integração dos padrões de grafos com a ontologia de domínio.

O arquivo de saída do FSG (*.fp) é um conjunto de subgrafos frequentes que também pode ser considerado um conjunto de grafos [99]. A figura 24 mostra um padrão representado na forma gráfica convencional e no formalismo de saída do FSG. Neste exemplo consideraremos que o padrão foi encontrado 3 vezes na base de grafos. Cada subgrafo frequente no arquivo de saída inicia-se com a letra "t"e nessa mesma linha em forma de comentário (comentário são precedidos por "#") são passados 2 informações adicionais: um identificador para o padrão, que no caso do exemplo da figura 24 é "2-0"(o 2 representa o número de arestas do grafo e o 0 indica que foi o primeiro subgrafo desse tamanho a ser encontrado) e a frequência do padrão minerado, que nesse caso é 3. Nas linhas iniciadas com o caractere "v"temos a definição dos vértices, com o *id* do vértice e um rótulo (um vértice representa uma determinada instância da ontologia, e seu rótulo deve ser igual ao identificador dessa instância na ontologia) para o mesmo respectivamente. Nas linhas iniciadas com o caractere "u"temos a definição das arestas, onde são passados os 2 vértices que se conectam através dos seus *ids*, e com o último parâmetro sendo para a rotulação da aresta (o rótulo de uma aresta representa alguma propriedade objeto da ontologia, ou seja algum tipo de relacionamento entre 2 instâncias).

O mediador semântico faz o trabalho de leitura do arquivo de padrões (*.fp) por meio de bibliotecas Java de manipulação de entrada e saída de dados. Como pode ser visto na figura 25, o módulo de leitura do mediador semântico recebe como entrada os padrões de grafos minerados e os converte (mapeia) para grafos JUNG, dessa forma possibilitando que os padrões de grafos possam ser manipulados e processados por aplicações em linguagem java.

As ontologias utilizadas em nossos experimentos são modeladas em linguagem OWL (recomendação *World Wide Web Consortium* (W3C)) e possuem semântica adicional com inserção de regras SWRL. Para leitura e mapeamento da ontologia, o módulo

Figura 24: Exibição de um Padrão de Grafo em 2 formalismos

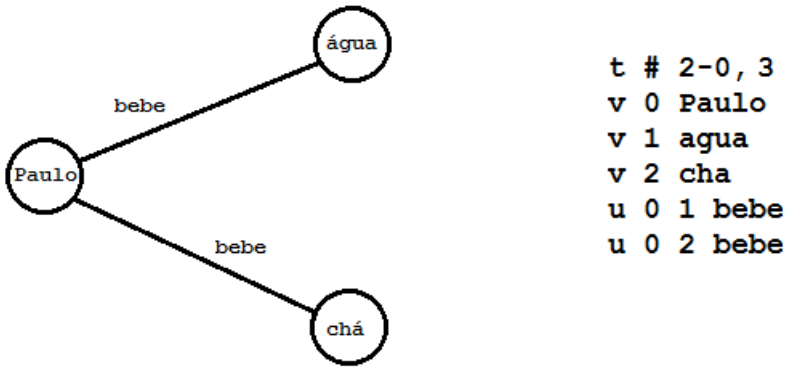
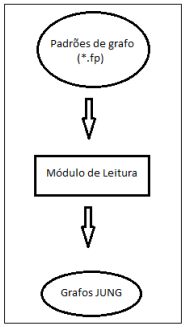


Figura 25: Leitura dos Padrões



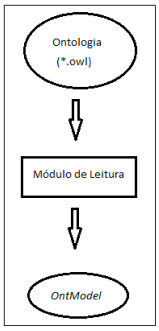
de leitura do mediador semântico utiliza bibliotecas nativas Java para manipulação de arquivos XML para fazer a leitura das regras SWRL, bem como o JENA para a leitura e mapeamento da ontologia OWL para objetos JAVA. A figura 26 mostra o esquema básico de leitura e mapeamento da ontologia, onde o módulo de leitura recebe a ontologia como entrada e a mapeia para um objeto da classe *OntModel* da API Jena, classe essa que possui métodos para manipulação de ontologias.

Em suma, essa etapa de leitura e mapeamento dos dados de entrada é responsável por converter os dados de entrada para objetos JAVA, dessa forma preparando o terreno para a próxima fase, o processamento dos dados.

5.6.2 Processamento dos Dados

Após os dados de entrada terem sido lidos e mapeados para um formato conveniente para a aplicação, inicia-se a fase de processamento dos dados, que é realizada pelo módulo de processamento do mediador semântico. É nesta etapa que é aplicado o algoritmo de enriquecimento semântico dos padrões minerados. O fluxograma da figura 27 ilustra bem esse processo de enriquecimento semântico. O processamento dos dados tem como atividades principais a geração de regras SWRL para os padrões e a busca

Figura 26: Leitura da Ontologia



de regra aplicável ao padrão.

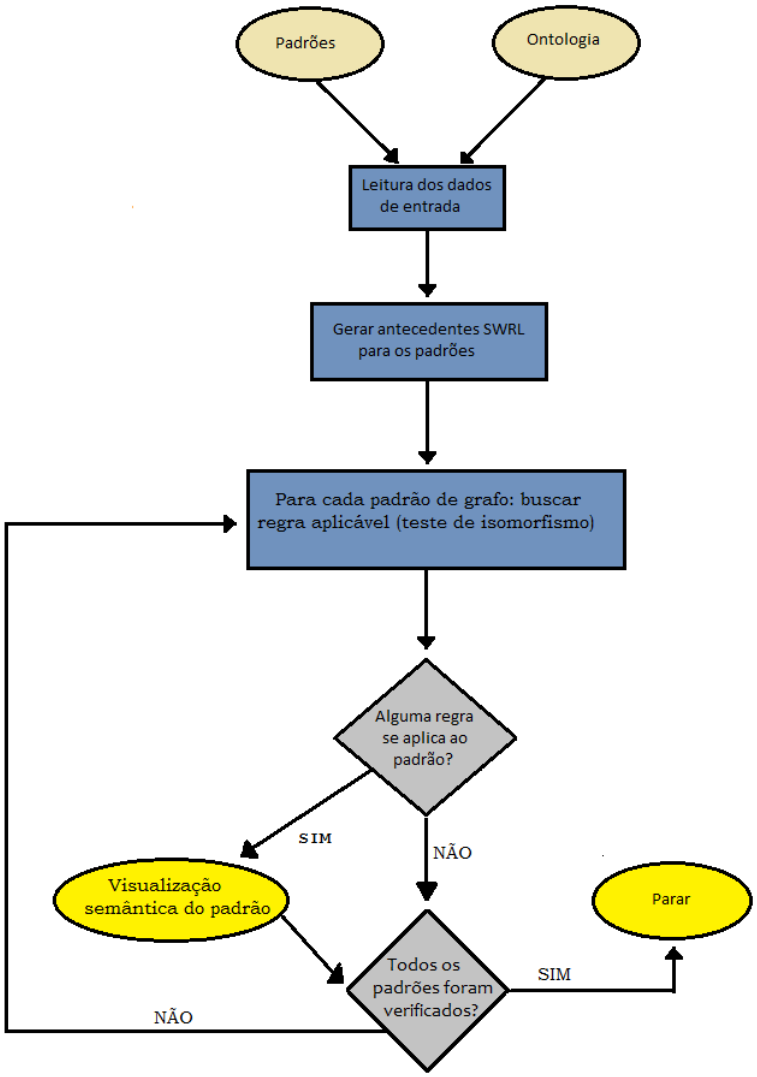
Geração de regras SWRL para os padrões De um lado nós temos os padrões de grafo minerados e do outro temos a ontologia de domínio. De alguma forma precisamos estabelecer uma conexão entre os padrões minerados e a ontologia, ou seja, queremos saber se existe na ontologia alguma regra SWRL que possa ser aplicada a determinado padrão minerado. Como mencionado anteriormente, o antecedente de uma regra SWRL é tratada como uma especificação padrão para uma consulta. Tomando por base essa informação viu-se primeiramente necessária a conversão dos padrões de grafos em antecedentes de regra SWRL. A figura 28 mostra um padrão de grafo simples e os antecedentes de regra SWRL geradas a partir dele.

O processo de geração de regras SWRL consiste em: ler o padrão de grafo a ser analisado; logo em seguida cada nó é transformado em átomo de classe (tanto do tipo instância como do tipo variável) e cada aresta é transformada em átomo de propriedade (tanto do tipo instância como do tipo variável), após isso, a partir das combinações possíveis entre os átomos gerados as possíveis regras para o grafo analisado são montadas. Por exemplo, na primeira regra gerada no exemplo da figura 28, é dito que Carlos é uma pessoa e ele joga Poker (vale ressaltar que a informação de que Carlos é uma pessoa é recuperada da própria ontologia de domínio). Da mesma forma, outra combinação de regra baseada naquele grafo diz que Carlos é uma pessoa e que ele joga alguma coisa (representado pela variável $?x$).

Busca de regra aplicável ao padrão Uma vez que temos os antecedentes de regra SWRL gerados para o padrão de grafo, o próximo passo do algoritmo é realizar a integração entre os padrões e a ontologia, onde é feito uma busca nas regras SWRL da ontologia, afim de encontrar uma regra que possa ser aplicada ao padrão minerado, ou seja, aplicado ao antecedente de regra gerado para o padrão.

Esse processo de casamento entre padrões minerados e regras SWRL é realizado

Figura 27: Fluxograma do processo de enriquecimento semântico



através de teste de isomorfismo de grafos, a figura 29 mostra a correspondência entre um padrão e uma regra (o grafo utilizado é o mesmo grafo da figura 28). O isomorfismo implica que esses grafos serão isomorfos se for possível ordenar seus vértices de forma que suas matrizes de adjacência sejam iguais. O problema de isomorfismo de grafos é de complexidade NP, e o algoritmo de teste de isomorfismo por força bruta convencional é caro computacionalmente na sua execução, uma vez que todas as combinações possíveis de sequências de vértices são testadas. Para reduzir esses problemas de execução, o algoritmo implementado usa algumas estratégias de refinamento para que não sejam executadas todas as possibilidades de sequência de vértices. Primeiramente o algoritmo só realizará o teste se os dois grafos forem de mesmo tamanho (número de vértices); e o outro refinamento é que um vértice só poderá ter como correspondente outro vértice da mesma classe, evitando que testes desnecessários sejam realizados. A figura 30 mostra

Figura 28: Geração de antecedentes SWRL para um padrão de grafo

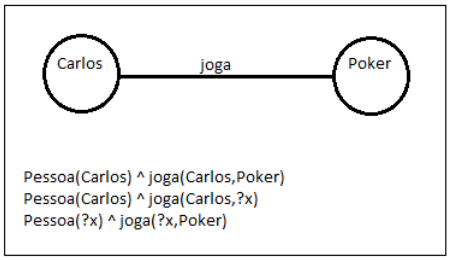
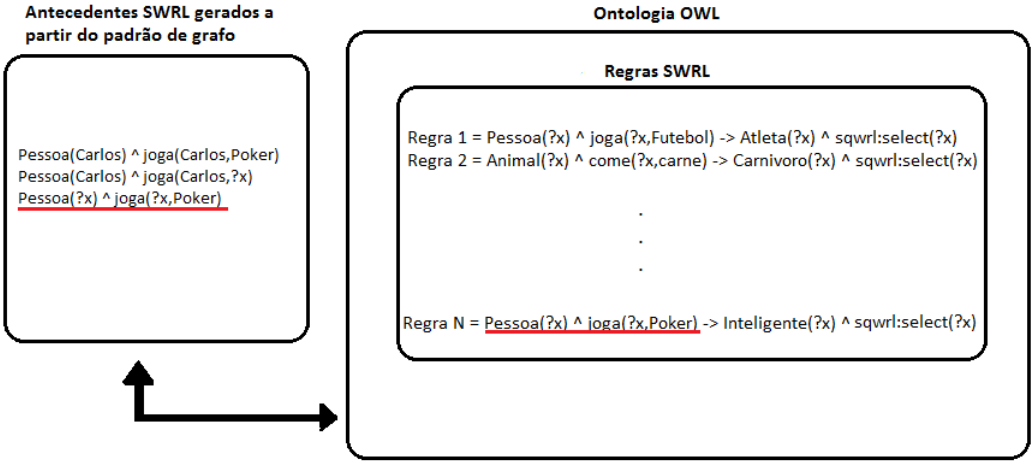


Figura 29: Correspondência entre antecedentes SWRL



o algoritmo de teste de isomorfismo implementado, perceba que o algoritmo retorna 1 quando o teste é realizado com sucesso (dois grafos são isomorfos) e 0 quando o contrário (quando os grafos não são isomorfos).

Figura 30: Teste de isomorfismo de grafos

Algoritmo: Força bruta com refinamentos para teste de isomorfismo de grafos

```
Função isomorfos(G,H)
    Se numVertices_G ≠ numVertices_H então
        retorne 0
    Fixe uma ordenação para o grafo G
    Escreva a matriz de adjacência A_c com esta ordenação
    Para cada ordenação de vértices de H que respeite a classe dos vértices faça
        Escreva A_H
        Se A_H=A_G então
            retorne 1
    retorne 0
```

A figura 29 mostra um exemplo de uma regra SWRL na ontologia com antecedente igual a um dos antecedentes gerados a partir do padrão. Após o algoritmo ter localizado uma regra compatível ao padrão é dado início à fase de exibição dos padrões semânticos.

5.6.3 Exibição dos padrões semânticos

Uma vez que foi encontrada uma regra que possa recuperar informações de contexto da ontologia e assim enriquecer os padrões minerados, faz-se necessário exibir essas informações de forma amigável ao usuário da aplicação. O módulo de exibição do mediador semântico então interpreta a regra encontrada e baseado em informações do padrão minerado retorna para o usuário informações de contexto extra a cerca do padrão, bem como uma visualização gráfica do mesmo. Tomando por base o exemplo da figura 29, temos o seguinte padrão: Carlos é uma pessoa e joga Poker. A regra encontrada na ontologia que satisfaça esse padrão é a seguinte: se uma pessoa x joga poker, então essa pessoa é considerada inteligente, ou seja, é também instância da classe inteligente. Vale ressaltar que essa informação não está explícita no padrão minerado, só foi possível obtê-la com a realização do processo de enriquecimento semântico.

6 Estudo de caso - Extração de motivos protéicos em bases de dados de proteínas

Antes de serem expostos os testes realizados com bases de proteínas, alguns conceitos sobre esse domínio serão abordados visando um melhor entendimento do trabalho.

6.1 Proteínas

As proteínas são polímeros constituídos de aminoácidos (20 aminoácidos diferentes podem constituir uma proteína, a tabela 1 mostra todos esses aminoácidos), e são as bio-moléculas mais abundantes nos seres vivos, estando presentes em todas as partes de uma célula, possuindo uma diversidade de funções biológicas. As proteínas podem ser representadas em 4 níveis estruturais: estrutura primária, secundária, terciária e quaternária. [100].

Nome	**Abreviatura**	**Letra**
Alanina	Ala	A
Cisteína	Cys	C
Aspartato (Ácido aspártico)	Asp	D
Glutamato (Ácido glutâmico)	Glu	E
Fenilalanina	Phe	F
Glicina	Gly	G
Histidina	His	H
Isoleucina	Ile	I
Lisina	Lys	K
Leucina	Leu	L
Metionina	Met	M
Asparagina	Asn	N
Prolina	Pro	P
Glutamina	Gln	Q
Arginina	Arg	R
Serina	Ser	S
Treonina	Thr	T
Valina	Val	V
Triptofano	Trp	W
Tirosina	Tyr	Y

Tabela 1: Aminoácidos da proteína

6.1.1 Estrutura Primária da Proteína

Nesse trabalho foram utilizadas proteínas representadas em estrutura primária, que basicamente são formadas por um arranjo linear (sequência) de aminoácidos, podendo assim ser facilmente modeladas como grafo [101]. Cada aminoácido pode ser representado por uma letra específica, como visto na tabela 1.

6.1.2 Motivos Protéicos

Motivos protéicos referem-se a uma sequência particular de aminoácidos com a característica de ter uma função bioquímica específica. O entendimento da funcionalidade dos motivos protéicos é de grande importância para se conhecer os mecanismos de atuação de uma proteína e consequentemente compreender a sua função [102]. A figura 31 mostra a sequência de aminoácidos que constituem a mioglobina humana, a sequência destacada representa um motivo protéico presente nessa proteína.

Figura 31: Motivo Protéico em destaque

6.2 Dados Utilizados

Para este trabalho foi utilizada uma base de dados de proteínas, mais especificamente a Mioglobina (de 24 espécies diferentes), em sua estrutura primária para a extração de padrões frequentes. Esses dados foram retirados da base de dados do UNIPROT [103]. Para a representação computacional dessas proteínas em forma de grafos (para que possam ser mineradas pelo FSG), foi criado um conversor, que recebe como entrada a sequência de aminoácidos da proteína e converte para o formato aceito pelo minerador de grafos FSG utilizado. Tomando por base a Mioglobina humana, a sua representação como grafo FSG pode ser visualisada na figura 32.

Uma vez que temos uma base de grafos de proteínas pronta para ser minerada, faz-se necessário a criação da ontologia para dar suporte ao enriquecimento semântico dos padrões minerados. Foi desenvolvida uma ontologia de Aminoácidos, bem como regras para a formação de motivos protéicos, que foram extraídos da base de dados ELM [104]. A figura 33 mostra a taxonomia da ontologia criada. Para exemplificar a utilização de regras na ontologia desenvolvida, tomaremos por base o motivo protéico

Figura 32: Representação da mioglobina humana como grafo FSG

```
T #Human
v 0 G
v 1 L
v 2 S
v 3 D
v 4 G
v 5 E
v 6 W
v 7 Q
v 8 L
v 9 V
v 10 L
      .
      .
      .
u 144 145 ligacao
u 145 146 ligacao
u 146 147 ligacao
u 147 148 ligacao
u 148 149 ligacao
u 149 150 ligacao
u 150 151 ligacao
u 151 152 ligacao
```

Figura 33: Hierarquia de classes da ontologia

encontrado na Mioglobina humana destacado na figura 31. O motivo protéico representado pela sequência de aminoácidos "KELGF"é denominado por "LIG_CYCLIN_1"e a regra SWRL da ontologia utilizada para tal motivo protéico é a seguinte:

```
Lisina(?k) ^ Glutamato(?e) ^ Leucina(?l) ^ Glicina(?g) ^ Fenilanina(?f) ^
ligacao(?k, ?e) ^ ligacao(?e, ?l) ^ ligacao(?l, ?g) ^ ligacao(?g, ?f) ->
sqwrl:select(LIG_CYCLIN_1)
```

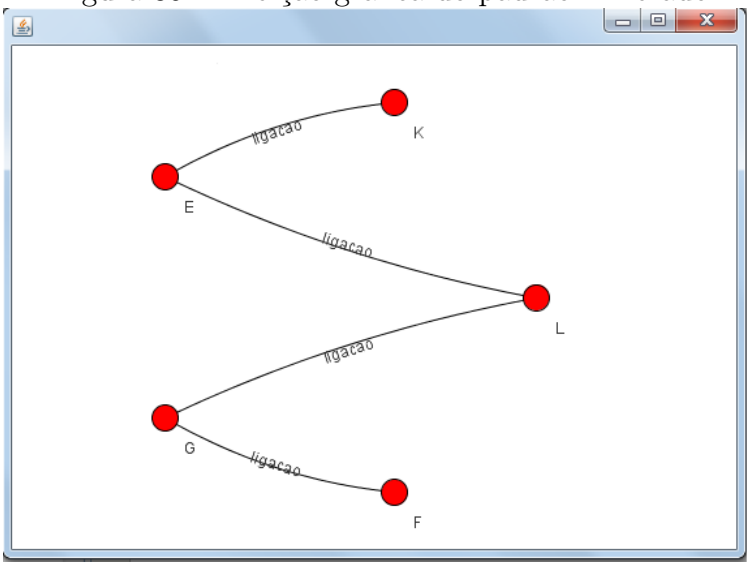
Figura 34: Motivo protéico LIG_CYCLIN_1 no formato de saída do FSG

```
t # 4-0, 17
v 0 K
v 1 E
v 2 L
v 3 G
v 4 F
u 0 1 ligacao
u 1 2 ligacao
u 2 3 ligacao
u 3 4 ligacao
```

6.3 Resultados

Nos testes realizados com as 24 proteínas da base de grafos, foi utilizado suporte mínimo de 30%. Foram encontrados 1897 padrões frequentes. Dentre todos os subgrafos frequentes encontrados, muitos não possuem um significado ou relevância maior. Porém, dentre os padrões encontrados alguns representam motivos protéicos já conhecidos na literatura, como por exemplo, o *LIG_14-3-3_3*, *LIG_CYCLIN_1*, *MOD_SUMO* e o *TRG_ENDOCYTIC_2*. Mesmo dentre esses padrões com significado e relevância comprovada, fica difícil de serem visualizados e interpretados apenas com o arquivo de saída do minerador. Por exemplo o motivo protéico "LIG_CYCLIN_1"que é representado pela sequência de aminoácidos "KELGF"pode ser visualizada no arquivo padrão de saída do minerador FSG como mostrado na figura 34. Levando-se em consideração que o arquivo de saída possui todos os 1897 subgrafos frequentes representados em sequência de forma textual, fica difícil até mesmo de ser localizado visualmente tal padrão, muitas vezes podendo passar desapercebido ao usuário.

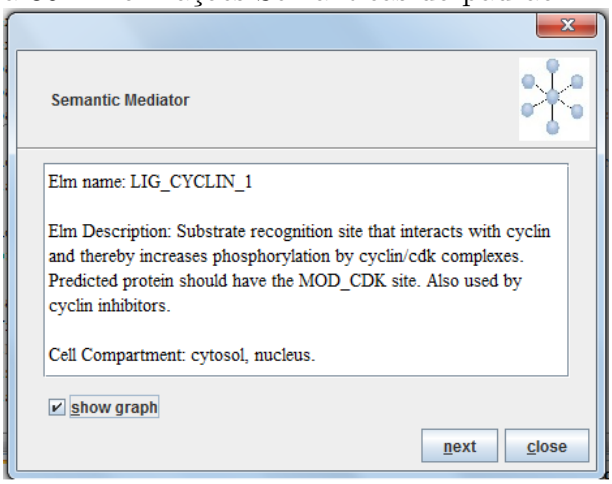
Figura 35: Exibição gráfica do padrão minerado



Tendo em vista essa dificuldade na visualização e interpretação dos padrões minerados (tomando por base o padrão da figura 34), a aplicação desenvolvida além de

exibir o padrão de forma gráfica (como apresentado na figura 35), também exibe uma janela (como apresentado na figura 36) com dados semânticos sobre tal padrão, dados esses vindos da ontologia de domínio, como proposto pelo processo de enriquecimento semântico.

Figura 36: Informações Semânticas do padrão minerado



7 Conclusões e Trabalhos Futuros

Este trabalho aborda uma proposta de enriquecimento semântico de padrões de grafos, onde os padrões minerados são integrados a recursos de ontologias para possibilitar uma interpretação mais eficiente das informações estratégicas extraídas dos domínios modelados como grafos. Uma aplicação prática com utilização de bases de dados de proteínas foi apresentada, mostrando assim um avanço na interpretação dos padrões minerados, onde informações extras acerca desses padrões (motivos protéicos) são exibidas de forma mais amigável ao usuário. Viu-se também na pesquisa relizada que a utilização de regras SWRL agregadas a ontologia OWL dá um maior poder de expressividade a ontologia.

O desenvolvimento do mediador semântico, elemento estratégico da arquitetura, tem demandado esforços no sentido de disponibilizar um componente de *software* aberto e eficiente, que possa associar adequadamente padrões de grafos e recursos da ontologia que maximizem o poder de interpretação do usuário. Alguns componentes da aplicação já foram totalmente desenvolvidos e outros ainda se encontram em desenvolvimento.

Dentre as maiores dificuldades encontradas no desenvolvimento dessa pesquisa, podemos citar a falta de bases de grafos modeladas nos formalismos utilizados pelos mineradores. Na maioria das vezes o que podemos utilizar são bases que podem ser convertidas para tais formalismos, como foi o caso da base de proteínas utilizada, onde um conversor foi desenvolvido para convertê-la ao formato entendido pelo minerador FSG. Outra dificuldade reside no campo das ontologias, apesar de ter o objetivo de padronizar as informações dispostas na rede, o que se vê é que existem diversas ontologias desenvolvidas para o mesmo domínio, o que vem a dificultar os trabalhos na área. Onde se torna muitas vezes mais viável desenvolver sua própria ontologia que utilizar uma já existente, uma vez que (na maioria dos casos) não há um repositório padrão.

Como trabalho futuro espera-se:

- Desenvolver ou melhorar um algoritmo de grafo já existente, que utilize ontologias dentro do próprio processo de mineração para a extração de informações semânticas;
- Disponibilizar uma ferramenta completamente funcional, bem como seu código-fone para a comunidade científica;
- expandir a implementação desenvolvida para outros algoritmos de mineração de grafos, bem como para outras linguagens de regras;

- otimizar o processo de enriquecimento semântico através da inserção de técnicas de teste de isomorfismo de grafos mais eficientes.

Referências

[1] G. Bernstein, "Jung 2.0 tutorial," 2009. Disponível em <http://www.grotto-networking.com/JUNG/JUNG2-Tutorial.pdf>, Acesso: 20 de março de 2011.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *American Association for Artificial Intelligence*, 1996.

[3] C. C. Aggarwal and H. Wang, *Managing and Mining Graph Data*. Springer Publishing Company, Incorporated, 2010.

[4] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2 ed., January 2006.

[5] N. F. Noy and D. L. Mcguinness, "Ontology Development 101: A Guide to Creating Your First Ontology," 2001.

[6] N. Guarino, "Formal ontology in information systems," in *FOIS'98: Proceedings of first International Conference on Formal Ontology in Information Systems*, pp. 3–15, 1998.

[7] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[8] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[9] Q. Mei, D. Xin, H. Cheng, J. Han, and C. Zhai, "Semantic annotation of frequent patterns," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 3, p. 11, 2007.

[10] M. C. Goldbarg and H. P. L. Luna, *Otimização Combinatória e Programação Linear*. Campus, 2000.

[11] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Comput. Surv.*, vol. 38, June 2006.

[12] P. O. B. Netto and S. Jurkiewicks, *Grafos: introdução e prática*. Blucher, 2009.

[13] D. Chakrabarti, *Tools for Large Graph Mining*. PhD thesis, 2005.

[14] Z. Ling, *An Effective Approach for Solving Subgraph Isomorphism Problem*, vol. 7, pp. 342–345. ACM, 1996.

[15] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," in *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, (Philadelphia, PA, USA), pp. 632–640, Society for Industrial and Applied Mathematics, 1995.

[16] J. Heer, S. K. Card, and J. A. Landay, "prefuse: a toolkit for interactive information visualization," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, (New York, NY, USA), pp. 421–430, ACM, 2005.

[17] D. Benson and G. Alder, "Jgraphx (jgraph 6) user manual," 2011.

[18] E. Adar, "Guess: a language and interface for graph exploration," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, (New York, NY, USA), pp. 791–800, ACM, 2006.

[19] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull, "Graphviz and dynagraph - static and dynamic graph drawing tools," in *GRAPH DRAWING SOFTWARE*, pp. 127–148, Springer-Verlag, 2003.

[20] S. Shrivastava, K. Kulshrestha, P. Singh, and S. N. Pal, "Pruthak: mining and analyzing graph substructures," in *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, (New York, NY, USA), pp. 110–118, ACM, 2010.

[21] C. R. B. de Souza, D. Redmiles, L.-T. Cheng, D. Millen, and J. Patterson, "Sometimes you need to see through walls: a field study of application programming interfaces," in *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, CSCW '04, (New York, NY, USA), pp. 63–71, ACM, 2004.

[22] S. Kelley, M. Goldberg, M. Magdon-Ismail, K. Mertsalov, W. Wallace, and M. Zaki, "graphont: an ontology based library for conversion from semantic graphs to jung," in *Proceedings of the 2009 IEEE international conference on Intelligence and security informatics*, ISI'09, (Piscataway, NJ, USA), pp. 170–172, IEEE Press, 2009.

[23] J. O'Madadhain, D. Fisher, P. Smyth, S. White, and Y.-B. Boey, "Analysis and visualization of network data using jung," *Journal of Statistical Software*, 2005.

[24] L. Singh, M. Beard, L. Getoor, and M. B. Blake, "Visual mining of multi-modal social networks at different abstraction levels," in *Proceedings of the 11th International Conference Information Visualization*, (Washington, DC, USA), pp. 672–679, IEEE Computer Society, 2007.

[25] M. P. dos Santos Silva, *Mineração de Dados: Conceitos, Aplicações e Experimentos com Weka*. 2004.

[26] A. Silberchtz, H. F. Korth, and S. Sudarshan, *Sistema de Banco de Dados*. 2006.

[27] S. Chakrabarti, E. Cox, E. Frank, R. H. Guting, J. Han, X. Jiang, M. Kamber, S. S. Lightstone, T. P. Nadeau, R. E. Neapolitan, D. Pyle, M. Refaat, M. Schneider, T. J. Teorey, and I. H. Witten, *Data Mining: Know It All*. Morgan Kaufmann, November 2008.

[28] D. Page and M. Craven, "Biological applications of multi-relational data mining," *SIGKDD Explor. Newsl.*, vol. 5, pp. 69–79, July 2003.

[29] Y. Hu and B. Panda, "A data mining approach for database intrusion detection," pp. 711–716, 2004.

[30] C. Borgelt, "Graph mining: An overview," in *Proc. 19th GMA/GI Workshop Computational Intelligence*, 2009.

[31] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, (New York, NY, USA), pp. 251–262, ACM, 1999.

[32] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, pp. 264–323, September 1999.

[33] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, "Graph clustering and minimum cut trees," *Internet Mathematics*, vol. 1, pp. 385–408, 2004.

[34] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," *Data Min. Knowl. Discov.*, vol. 15, no. 1, pp. 55–86, 2007.

[35] R. Agrawal and R. Srikant, "Readings in database systems (3rd ed.)," ch. Fast algorithms for mining association rules, pp. 580–592, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[36] A. Nanopoulos and Y. Manolopoulos, "Efficient similarity search for market basket data," *The VLDB Journal*, vol. 11, pp. 138–152, October 2002.

[37] M. Kuramochi and G. Karypis, "An efficient algorithm for discovering frequent subgraphs," *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, pp. 1038–1051, September 2004.

[38] T. Washio and H. Motoda, "State of the art of graph-based data mining," *SIGKDD Explor. Newsl.*, vol. 5, no. 1, pp. 59–68, 2003.

[39] I. Takigawa and H. Mamitsuka, "Efficiently mining o-tolerance closed frequent subgraphs," *Machine Learning*, vol. 82, pp. 95–121, 2011. 10.1007/s10994-010-5215-6.

[40] Z. Zeng, J. Wang, L. Zhou, and G. Karypis, "Out-of-core coherent closed quasi-clique mining from large dense graph databases," *ACM Trans. Database Syst.*, vol. 32, June 2007.

[41] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, (London, UK), pp. 13–23, Springer-Verlag, 2000.

[42] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," *IEEE International Conference on Data Mining*, vol. 0, p. 313, 2001.

[43] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," *Proceeding of the 2002 international conference on data mining (ICDM'02)*, pp. 721–724, 2002.

[44] N. Vanetik, E. Gudes, and S. E. Shimony, "Computing frequent graph patterns from semistructured data," *IEEE International Conference on Data Mining*, vol. 0, p. 458, 2002.

[45] C. Borgelt and M. R. Berthold, "Mining molecular fragments: Finding relevant substructures of molecules," in *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, (Washington, DC, USA), pp. 51–, IEEE Computer Society, 2002.

[46] J. Huan, W. Wang, and J. Prins, "Efficient mining of frequent subgraphs in the presence of isomorphism," in *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, (Washington, DC, USA), pp. 549–, IEEE Computer Society, 2003.

[47] J. Huan, W. Wang, J. Prins, and J. Yang, "Spin: mining maximal frequent subgraphs from graph databases," in *KDD'04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 581–586, ACM, 2004.

[48] S. Nijssen and J. N. Kok, "A quickstart in frequent structure mining can make a difference," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 647–652, ACM, 2004.

[49] S. Nijssen and J. N. Kok, "The gaston tool for frequent subgraph mining," *Electron. Notes Theor. Comput. Sci.*, vol. 127, no. 1, pp. 77–87, 2005.

[50] O. Corcho, M. Fernandez-López, and A. Gomez-Pérez, "Methodologies, tools and languages for building ontologies. where is their meeting point?," *Data & Knowledge Engineering*, vol. 46, no. 1, pp. 41 – 64, 2003.

[51] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199 – 220, 1993.

[52] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907 – 928, 1995.

[53] D. C. Kern, "Inferência sobre ontologias." 2007.

[54] M. B. Almeida and M. P. Bax, "Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção," *Ciência da Informação*, vol. 32, no. 3, pp. 7–20, 2003.

[55] D. B. Lenat and R. V. Guha, "The evolution of cycl, the cyc representation language," *SIGART Bull.*, vol. 2, pp. 84–87, June 1991.

[56] M. Kifer, G. Lausen, and J. Wu, "Logical foundations of object-oriented and frame-based languages," tech. rep., 1990.

[57] U. of Southern California, "Quickguide to loom functions and relations," 2009. Disponível em <http://www.isi.edu/isd/LOOM/documentation/LOOM-DOCS.html>, Acesso: 14 de julho de 2011.

[58] U. of Southern California, "Powerloom manual," 2006. Disponível em <http://www.isi.edu/isd/LOOM/PowerLoom/documentation/documentation.html>, Acesso: 14 de julho de 2011.

[59] F. Goasdoué, M.-C. Rousset, and et al., "The use of carin language and algorithms for information integration: The picsel system," in *INTELLIGENT INFORMATION INTEGRATION WORKSHOP ASSOCIATED WITH ECAI'98 CONFERENCE*, 1999.

[60] A. Rector, S. Bechhofer, C. Goble, I. Horrocks, W. Nowlan, and W. Solomon, "The grail concept modelling language for medical terminology," *Artificial Intelligence in Medicine*, vol. 9, pp. 139–171, 1997.

[61] E. Motta, "An overview of the ocml modelling language," in *In Proceedings KEML'98: 8th Workshop on Knowledge Engineering Methods & Languages*, pp. 21–22, 1998.

[62] R. Kent, "Conceptual knowledge markup language: An introduction," *Netnomics*, vol. 2, pp. 139–169, March 2000.

[63] W3C, "Resource description framework (rdf): Concepts and abstract syntax," 2004. Disponível em <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>, Acesso: 09 de dezembro de 2010.

[64] W3C, "Rdf vocabulary description language 1.0: Rdf schema," 2004. Disponível em <http://www.w3.org/TR/rdf-schema/>, Acesso: 10 de dezembro de 2010.

[65] G. P. Zarri, "Nkrl, a knowledge representation language for narrative natural language processing," in *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, (Stroudsburg, PA, USA), pp. 1032–1035, Association for Computational Linguistics, 1996.

[66] J. Heflin, J. Hendler, and S. Luke, "Applying ontology to the web: A case study," in *In Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, IWANN'99*, pp. 715–724, Springer, 1999.

[67] D. Fensel, I. Horrocks, F. v. Harmelen, S. Decker, M. Erdmann, and M. C. A. Klein, "Oil in a nutshell," in *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, EKAW '00, (London, UK), pp. 1–16, Springer-Verlag, 2000.

[68] D. L. McGuinness, R. Fikes, J. Hendler, and L. A. Stein, "Daml+oil: An ontology language for the semantic web," *IEEE Intelligent Systems*, vol. 17, pp. 72–80, September 2002.

[69] W3C, "Owl web ontology language overview," 2004. Disponível em <http://www.w3.org/TR/owl-features/>, Acesso: 12 de dezembro de 2010.

[70] G. Antoniou and F. v. Harmelen, *A Semantic Web Primer, 2nd Edition (Cooperative Information Systems)*. The MIT Press, 2 ed., 2008.

[71] M. Horridge, "A practical guide to building owl ontologies using protégé 4 and co-ode tools edition 1.2," 2009.

[72] D. Skuce and T. C. Lethbridge, "Code4: A unified system for managing conceptual knowledge," *International Journal of Human-Computer Studies*, vol. 42, pp. 413–451, 1995.

[73] A. Farquhar, R. Fikes, and J. Rice, "The ontolingua server: a tool for collaborative ontology construction," in *International Journal of Human-Computer Studies*, 1996.

[74] S. M. Paley, J. D. Lowrance, and P. D. Karp, "A generic knowledge-base browser and editor," in *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, AAAI'97/IAAI'97, pp. 1045–1051, AAAI Press, 1997.

[75] K. Mahalingam and M. N. Huhns, "A tool for organizing web information," *Computer*, vol. 30, pp. 80–83, June 1997.

[76] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "Webode: a scalable workbench for ontological engineering," in *Proceedings of the 1st international conference on Knowledge capture*, K-CAP '01, (New York, NY, USA), pp. 6–13, ACM, 2001.

[77] J. Domingue, E. Motta, and O. Corcho Garcia, "Knowledge modelling in webonto and ocml: A user guide," 2009.

[78] D. Faure and C. Nédellec, "ASIUM: Learning subcategorization frames and restrictions of selection," in *10th Conference on Machine Learning (ECML 98), Workshop on Text Mining*, 1998.

[79] B. McBride, "Jena: A semantic web toolkit," *IEEE Internet Computing*, vol. 6, pp. 55–59, November 2002.

[80] S. C. for Biomedical Informatics Research, "What is protégé?," 2010. Disponível em <http://protege.stanford.edu/overview/>, Acesso: 15 de dezembro de 2010.

[81] B. McBride, "An introduction to rdf and the jena rdf api," 2010. Disponível em <http://jena.sourceforge.net/tutorial/RDF_API/index.html>, Acesso: 19 de dezembro de 2010.

[82] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *Proceedings*

of the 13th international World Wide Web conference on Alternate track papers & posters, WWW Alt. '04, (New York, NY, USA), pp. 74–83, ACM, 2004.

[83] S.-m. Zhang, J.-y. Guo, Z.-t. Yu, C.-y. Lei, C.-l. Mao, and H.-x. Wang, "An approach of domain ontology construction based on resource model and jena," in *Proceedings of the 2010 Third International Symposium on Information Processing*, ISIP '10, (Washington, DC, USA), pp. 311–315, IEEE Computer Society, 2010.

[84] M. Grobe, "Rdf, jena, sparql and the 'semantic web'," in *Proceedings of the 37th annual ACM SIGUCCS fall conference*, SIGUCCS '09, (New York, NY, USA), pp. 131–138, ACM, 2009.

[85] K. K. Breitman, *Web semântica: a internet do futuro*, vol. 1. Rio de Janeiro: LTC, 2005. ISBN 85-216-1466-7.

[86] S. C. for Biomedical Informatics Research, "What is protégé-frames?," 2010. Disponível em <http://protege.stanford.edu/overview/protege-frames.html>, Acesso: 15 de dezembro de 2010.

[87] S. C. for Biomedical Informatics Research, "What is protégé-owl?," 2010. Disponível em <http://protege.stanford.edu/overview/protege-owl.html>, Acesso: 15 de dezembro de 2010.

[88] S. Bragaglia, F. Chesani, P. Mello, and D. Sottara, "A rule-based implementation of fuzzy tableau reasoning," in *Proceedings of the 2010 international conference on Semantic web rules*, RuleML'10, (Berlin, Heidelberg), pp. 35–49, Springer-Verlag, 2010.

[89] M. J. O'Connor, R. D. Shankar, S. W. Tu, C. I. Ny, and A. K. Dasulas, "Developing a Web-Based Application using OWL and SWRL," in *AAAI Spring Symposium*, 2008.

[90] M. O'connor, H. Knublauch, S. Tu, B. Grosof, M. Dean, W. Grosso, and M. Musen, "Supporting Rule System Interoperability on the Semantic Web with SWRL," pp. 974–986, 2005.

[91] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. T. B. Grosof, and M. Dean, "Swrl: A semantic web rule language combining owl and ruleml," 2004. Disponível em <http://www.w3.org/Submission/SWRL/>, Acesso: 20 de dezembro de 2010.

[92] M. J. O'Connor and A. K. Das, "SQWRL: a Query Language for OWL," in *OWL: Experiences and Directions (OWLED), Fifth International Workshop*, 2009.

[93] K. M. Borgwardt and X. Yan, "Graph mining and graph kernels," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, (New York, NY, USA), ACM, 2008.

[94] T. Yu, X. Jiang, and Y. Feng, "Semantic graph mining for e-science," *AAAI Library*, pp. 77–80, 2007.

[95] A. Cakmak and G. Ozsoyoglu, "Taxonomy-superimposed graph mining," in *EDBT '08: Proceedings of the 11th international conference on Extending database technology*, (New York, NY, USA), pp. 217–228, ACM, 2008.

[96] B. Aleman-Meza, M. Nagarajan, L. Ding, A. Sheth, I. B. Arpinar, A. Joshi, and T. Finin, "Scalable semantic analytics on social networks for addressing the problem of conflict of interest detection," *ACM Trans. Web*, vol. 2, pp. 7:1–7:29, March 2008.

[97] B. Aleman-Meza, M. Nagarajan, C. Ramakrishnan, L. Ding, P. Kolari, A. P. Sheth, I. B. Arpinar, A. Joshi, and T. Finin, "Semantic analytics on social networks: experiences in addressing the problem of conflict of interest detection," in *Proceedings of the 15th international conference on World Wide Web*, WWW '06, (New York, NY, USA), pp. 407–416, ACM, 2006.

[98] Q. Mei, D. Xin, H. Cheng, J. Han, and C. Zhai, "Generating semantic annotations for frequent patterns with context analysis," in *KDD'06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 337–346, ACM, 2006.

[99] K. M. K. G. Seno, M., "Pafi: A pattern finding toolkit," 2003. Technical Report.

[100] I. E. Francisco Junior and W. Francisco, "Proteínas: Hidrólise, precipitação e um tema para o ensino de química," 2006.

[101] S. Freeman, *Biological Science*. Prentice Hall, 2002.

[102] F. A. C. dos Anjos, "Visualização de motivos protéicos em ambiente tridimensional," 2009.

[103] A. Bairoch, "Uniprotkb/swiss-prot: New and future developments," in *Proceedings of the 5th international workshop on Data Integration in the Life Sciences*, DILS '08, (Berlin, Heidelberg), pp. 204–206, Springer-Verlag, 2008.

[104] C. M. Gould, F. Diella, A. Via, P. Puntervoll, C. Gemund, S. Chabanis-Davidson, S. Michael, A. Sayadi, J. C. Bryne, C. Chica, M. Seiler, N. E. Davey, N. Haslam,

R. J. Weatheritt, A. Budd, T. Hughes, J. Pas, L. Rychlewski, G. Trave, R. Aasland, M. Helmer-Citterich, R. Linding, and T. J. Gibson, "ELM: the status of the 2010 eukaryotic linear motif resource," *Nucl. Acids Res.*, pp. gkp1016+, Nov. 2009.