MINISTÉRIO DA DEFESA

EXÉRCITO BRASILEIRO

DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA

INSTITUTO MILITAR DE ENGENHARIA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

CAMILA ANTONACCIO WANOUS

JOÃO LUIZ DO PRADO NETO

UMA FERRAMENTA PARA ANÁLISE ESTRUTURAL E PREDIÇÃO DE LIGAÇÕES EM BANCO DE DADOS ORIENTADOS A GRAFOS

RIO DE JANEIRO

2015

INSTITUTO MILITAR DE ENGENHARIA

CAMILA ANTONACCIO WANOUS

JOÃO LUIZ DO PRADO NETO

UMA FERRAMENTA PARA ANÁLISE ESTRUTURAL E PREDIÇÃO DE LIGAÇÕES EM BANCO DE DADOS ORIENTADOS A GRAFOS

Projeto de Final de Curso apresentado ao Curso de Engenharia de Computação do Instituto Militar (IME) de Engenharia, como parte das exigências do IME.

Orientadores: Profª. Maria Claudia Reis Cavalcanti, D.Sc., do IME e Profª. Claudia Marcela Justel - D.Sc., do IME

RIO DE JANEIRO

2015

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro - RJ CEP: 22290- 270

006.32 WANOUS, Camila Antonaccio
O48e Uma ferramenta para análise estrutural e predição de ligações em bancos de dados orientados a grafos/ Camila Antonaccio Wanous, João Luiz do Prado Neto; orientados por Maria Claudia Reis Cavalcanti e Claudia Marcela Justel - Rio de Janeiro: Instituto Militar de Engenharia, 2015.

47p. : il.

Monografias (PROFIC) - Instituto Militar de Engenharia, - Rio de Janeiro, 2015.

1. Curso de Engenharia de Computação - monografias. 2. Banco de Dados Grafos I. Wanous,Camila Antonaccio II. Prado, João Luiz do de III. Cavalcanti, Maria Claudia Reis IV. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

CAMILA ANTONACCIO WANOUS

JOÃO LUIZ DO PRADO NETO

UMA FERRAMENTA PARA ANÁLISE ESTRUTURAL E PREDIÇÃO DE LIGAÇÕES EM BANCO DE DADOS ORIENTADOS A GRAFOS

Projeto de Final de Curso apresentado ao Curso de Engenharia de Computação do Instituto Militar (IME) de Engenharia, como parte das exigências do IME.

Aprovado em 27/05/2015 pela seguinte Banca Examinadora:

Maria Claudia Reis Cavalcanti, D.Sc., do IME

Claudia Marcela Justel - D.Sc., do IME

Ronaldo Ribeiro Goldschmidt - D.Sc., do IME

Julio Cesar Duarte - D.Sc., do IME

RIO DE JANEIRO

2015

"Viver é a coisa mais rara do mundo. A maioria das pessoas apenas existe".

Oscar Wilde

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

RESUMO

A visualização de grafos pode dizer muito acerca das informações armazenadas. Atualmente as formas existentes para a visualização desse tipo de informação não são tão diversificadas a ponto de atender as necessidades de todos os usuários, como por exemplo prever ligações futuras. Tendo isto como base e outros trabalhos desenvolvidos na própria instituição pretende-se criar uma ferramenta que ofereça ao usuário a predição de ligações, além de um sistema que integre banco de dados e visualização dos dados. Um exemplo prático de aplicação é a rede de colaboração de pesquisadores, na qual pode-se ter nós que representam os autores, as publicações e os veículos no qual a publicação foi feita e por meio destas informações gerar-se predições a respeito de autores que poderiam publicar juntos ou de novos veículos de publicação para um autor.

Por meio da predição de ligações, pode-se obter informações mais detalhadas sobre os dados e de como eles interagem. A título, tem-se a possibilidade de avaliar a evolução da rede em sua formação de conexões ou de servir como ferramenta comercial para empresas que desejem fazer anúncios inteligentes; isto é oferecer seus produtos a pessoas com maior probabilidade de adquiri-los.

Palavras-chave: Predição de Ligações, Grafos, Banco de Dados, Visualização.

ABSTRACT

The visualization of graphs can show us a lot about the stored information. Currently the offered ways to display information are not diversified enough to content the needs of all users, such as predicting future connections. Using this like basis and other works developed in our institution, we aim to create a tool that offers a link prediction, as well as a system that integrates database and data visualization. A practical example is the collaboration network of researchers, which you can have nodes that represent the authors, publications and vehicles in which the publication was made and through this information to generate predictions about authors could publish together or publication of new vehicles for an author.

Through the link prediction, we can obtain more detailed information about the data and how they interact. For example, we have the possibility to check the evolution of the network connections in formation or can be used like business tool for companies who want make smart ads; it means offering their products to people with higher probability to buy them.

Keywords: Link Prediction, Graph, Database, Visualization.

1 INTRODUÇÃO

Nas duas últimas décadas, avanços tecnológicos proporcionaram um crescimento na quantidade de dados, sobre os mais diversos assuntos, os quais estão facilmente disponíveis para consulta de qualquer pessoa com acesso à Internet. Pode-se considerar um grande desafio aos pesquisadores da área de Ciência da Computação a elaboração de métodos que prevejam com precisão a existência de ligações entre estes dados [1]. A predição de ligações tem sido um assunto muito abordado pela comunidade científica e pode-se entendê-la como a previsão da existência de uma aresta entre dois nós de um grafo que representam os dados, enquanto as arestas representam ligações entre eles. Os nós podem ser homogêneos, ou seja, todos representam o mesmo tipo de dado, ou podem apresentar diferentes tipos de dados. Informações topológicas (métricas extraídas a partir da estrutura do grafo associado), e informações sobre os vértices/arestas são utilizadas como base para construção dos modelos preditivos [2].

A predição de ligações envolve diversas áreas do conhecimento como algoritmos em grafos, banco de dados, engenharia de software e mineração de dados, tornando seu estudo multidisciplinar. No contexto da Web, desenvolver técnicas para lidar com a enorme quantidade de dados, que tende a crescer, é mais uma frente a ser estudada pelos pesquisadores [2].

No estudo da predição de ligações, as redes sociais aparecem com destaque. Nelas a predição de ligações é vista como um problema de evolução temporal, ou sobre o aspecto das conexões faltantes. Dada uma rede social em um dado instante, a predição procura determinar, com a maior precisão possível, quais serão as ligações adicionadas em um momento futuro, além de detectar conexões implícitas na rede [3]. Diversas redes virtuais vêm se formando e se tornando populares nos últimos anos, aumentando a relevância de estudos relacionados à predição de ligações. Redes de relacionamento e sugestão de produtos que possam ser interessantes para o usuário são apenas alguns exemplos [4].

1.1 MOTIVAÇÃO

A predição de ligações é um tema corrente na área da Ciência da Computação, pois possui aplicações em diversas áreas. Previsões de ligações podem, por exemplo, ser utilizadas na realização da predição de colaborações em artigos (área abordada nesse trabalho), onde pode-se tentar predizer quais autores irão colaborar entre si na produção de artigos científicos [5]. Outros exemplos de aplicações das predições de ligações incluem sistemas de recomendação, onde uma ligação consiste em uma relação entre objeto e usuário, como o sistema da Amazon; sistemas de recuperação da informação; ou até mesmo sistemas de auxílio na análise da propagação de doenças, no qual o relacionamento entre indivíduos se caracteriza pelo contágio, ou seja, as relações entre nós indicam quem infectou quem [6].

Atualmente, existe uma área da computação destinada exclusivamente ao estudo do comportamento dos usuários dentro das redes sociais, chamada de Análise de Redes Sociais (SNA - *Social Network Analysis*), e dentro dessa área a predição de ligações é uma importante área de pesquisa. Estudos nessa área são relevantes em vários aspectos, como economia (análise de compras) e segurança (redes terroristas), visto que seus resultados permitem a realização de análises estratégicas baseadas em informações extraídas das redes [7].

Independentemente do algoritmo de predição de ligações aplicado sobre uma rede qualquer, a visualização dessas novas ligações, por parte do pesquisador, não é uma tarefa simples. Facilitar a visualização das ligações previstas, de modo que o pesquisador possa acompanhar visualmente tanto a evolução temporal da base de dados estudada, e compará-la a prevista, quanto à própria predição sob a ótica de um nó qualquer, auxiliaria os estudos sobre o tema.

A criação de uma ferramenta que auxilie ao pesquisador nesse estudo é importante devido à relevância atual do estudo da predição de ligação.

1.2 OBJETIVO

O trabalho tem como objetivo desenvolver uma ferramenta para predição de ligações que integre um Sistema de Gerenciamento de Banco de Dados e um Sistema Visualizador e Analisador de Grafos e possibilite o armazenamento e análise dos dados

Dado um conjunto de informações estruturadas num banco de dados, a ferramenta desenvolvida deverá ser capaz de:

- Representar e analisar as informações na forma de um grafo;
- Permitir o uso de filtros para selecionar um subconjunto de dados desejado;
- Aplicar sobre estas informações no formato de grafo um algoritmo de predição de ligações;
- Disponibilizar ao usuário uma interface de visualização das informações permitindo identificar arestas obtidas por um algoritmo de predição de ligações;
- Representar no banco de dados as informações atualizadas, com base nos resultados da predição de ligações sobre o grafo.

Existem disponíveis no mercado ferramentas de visualização, de manipulação, e de armazenamento de grafos, mas não há uma ferramenta que integre todas essas funcionalidades.

1.3 JUSTIFICATIVA

O trabalho está relacionado a diversas iniciativas e outros trabalhos de pesquisa da seção, já em andamento na forma de dissertação de mestrado, como pesquisa de banco de dados orientado a grafos ou mineração de dados e permite a integração de alunos da graduação e da pós graduação, a saber [7], o qual está inserido dentro do contexto de predição de

ligações, relacionado com sistema de recomendações e [8] que está associada ao crescimento exponencial do volume de dados captados junto aos mais diversos segmentos da sociedade, sendo estes disponibilizados pela Internet e Web. Nesse contexto, um dos grandes desafios é prever com precisão a existência de ligações entre os dados.

O desenvolvimento é o resultado de uma interação entre duas áreas de pesquisa complementares, a saber, banco de dados, e algoritmos em grafos. O intuito é desenvolver uma ferramenta para colaborar com a pesquisa na área de grafos e predição fornecendo novas formas de visualização de dados.

1.4 METODOLOGIA

A metodologia proposta para este trabalho consta de três fases. A primeira consistiu em um levantamento das ferramentas relacionadas ao armazenamento de estruturas de grafos e visualização de grafos disponíveis no mercado. Na segunda foram realizados testes com as ferramentas pesquisadas, para que seja avaliada a viabilidade de alcançar o objetivo do trabalho por meio das ferramentas analisadas, e dentre estas, qual a melhor opção. Também durante a segunda fase foi construída uma base de dados real para fazer o estudo de caso. A terceira fase consistiu na implementação da ferramenta proposta, incluindo um algoritmo básico de predição de ligações, assim como a criação da interface com o usuário.

Na primeira etapa, foram levantadas ferramentas de visualização de grafos e de banco de dados orientados a grafos. Os requisitos que devem atender a ferramenta selecionada foram:

- Garantir a persistência dos dados (estruturados sobre a forma de um grafo) para permitir acompanhar a evolução de dados no tempo;
- Possuir uma interface que permita a visualização do grafo de forma dinâmica, ou seja, que permita ao usuário interagir com a visualização do grafo por filtro de nós ou relacionamentos, e permita diferenciar relacionamentos previstos dos existentes;
- Exportar dados de forma que possam ser armazenados, novamente no banco de dados;

Durante a segunda fase, uma vez selecionadas as ferramentas, as suas formas de integração foram testadas no ambiente Windows. Além das ferramentas voltadas exclusivamente para a manipulação de grafos, selecionou-se uma plataforma de desenvolvimento compatível com as ferramentas pré-selecionadas. Os testes realizados com as ferramentas escolhidas foram:

- A comunicação/integração entre as diversas ferramentas e plataformas de desenvolvimento;
- Formatos de entrada e saída de dados;

Após a consolidação das duas primeiras fases foi elaborada a arquitetura final da solução, e iniciou-se a implementação do produto final. Para atender o objetivo deste trabalho ainda foi necessário completar as seguintes atividades:

- Escolher o algoritmo de predição de ligações a ser implementado;
- Melhorar a interface com o usuário;
- Realizar o estudo de casos para comprovar o funcionamento da ferramenta.

1.5 ESTRUTURA

A estrutura do restante do texto está organizada da seguinte maneira: no capítulo 2, será introduzida a fundamentação teórica do projeto. No capítulo 3, será mostrada a proposta do projeto a ser desenvolvido. Essa fundamentação é composta por uma introdução sobre banco de dados orientados a grafos, sobre ferramentas de bancos de dados e de visualização, além de conceitos básicos sobre predição de ligações. No capítulo 4, é descrito o processo de desenvolvimento do projeto. No capítulo 5 são apresentadas as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será realizada a fundamentação do projeto, sendo explicitados os conceitos básicos e descrita a pesquisa realizada para determinar as ferramentas mais adequadas para atingir o objetivo do projeto. Além disso, apresentamos a base conceitual utilizada para a construção do algoritmo de predição de ligações.

2.1 NOSQL E BANCO DE DADOS ORIENTADOS A GRAFOS

NoSQL são diferentes sistemas de armazenamento e gerência de dados (i.e. SGBDs) que vieram para suprir necessidades em demandas onde os bancos de dados tradicionais (relacionais) são ineficazes. Muitas dessas bases apresentam características muito interessantes como alta performance, escalabilidade, replicação, suporte a dados estruturados e sub colunas [9].

Os sistemas NoSQL surgiram da necessidade de uma performance superior e de uma alta escalabilidade. Os atuais bancos de dados relacionais são restritos quanto a isso, sendo necessária a distribuição vertical de servidores. Um grande utilizador desse conceito é o Google, que usa computadores de pequeno e médio porte para a distribuição dos dados; essa forma de utilização é muito mais eficiente e econômica [9].

Existem 4 categorias principais de sistemas NoSQL, a saber: baseados em chave-valor; em família de colunas; em grafos e em documentos. Cada uma desses sistemas pode ser usado preferencialmente para armazenar informações em contextos particulares [9].

Neste trabalho iremos tratar o caso de sistemas NoSQL orientados a grafos. Este tipo de sistema NoSQL é usado preferencialmente para armazenar informações que são obtidas a partir de redes sociais; ou quando trabalha-se com dados altamente relacionados [10]. Existem algumas ferramentas que implementam sistemas NoSQL baseados em grafos, a exemplo: Neo4J, Allegrograph e OrientDB.

Um sistema NoSQL orientado a grafos pode armazenar dados na forma de um conjunto de nós e relacionamentos. Quando combina-se nós e relacionamentos obtem-se um grafo. Neste caso, existem três campos de dados: nós, relacionamentos e propriedades [11].

Os nós do grafo, são em geral, representações de objetos do mundo real, como substantivos. Podem ser, por exemplo, pessoas, organizações, números de telefone, páginas Web,

computadores numa rede, ou células biológicas num organismo vivo. Os relacionamentos são conexões entre esses objetos e são representados por arestas. Consultas em grafos são semelhantes a percursos através dos nós do grafo. Exemplos deste tipo de consulta são:

- Qual o menor caminho entre dois nós do grafo?
- Quais os vizinhos de um dado nó que possuem propriedades específicas?

Os sistemas NoSQL orientado a grafos permitem fazer consultas aproveitando a estrutura de grafo definida entre os objetos que o compõem e permitem analisar relações entre esses objetos, além de visitar todos os objetos, de maneira otimizada.

2.1.1 NEO4J, ALLEGROGRAPH E ORIENTDB

A primeira versão do Neo4J foi lançada em fevereiro de 2010 pela empresa Neo Technology15. Sua implementação foi feita em Java e possui duas versões de licenciamento, uma aberta e outra proprietária. A última tem código fechado, oferece suporte aos usuários e possui mais recursos para garantir o desempenho de grandes aplicações.

O Neo4J, como dito anteriormente, é um sistema do tipo NoSQL orientado a grafos. Possui uma estrutura que o garante uma boa performance e requer menor poder de processamento para requisições complexas. Especificamente, os sistemas NoSQL baseados em grafos possuem um desempenho significativamente melhor para executar buscas de relacionamentos entre dados cadastrados, diminuindo o número de requisições e comandos para encontrar dados relacionados [12].

Em um SGBD relacional, para realizar buscas, deve-se navegar em diversas tabelas para encontrar as informações desejadas, o que pode carregar a aplicação, no caso de pesquisas complexas. Já no Neo4J, a busca realizada percorre apenas as entidades relacionadas, definindo-se previamente a profundidade da busca realizada, economizando processamento e melhorando a performance do tempo de resposta [12].

Os dados, no Neo4J, são registrados no formato chave-valor, ou seja, pode-se acrescentar propriedades aos dados registrados sem que haja a necessidade de modificar tabelas, como

acontece nos bancos de dados relacionais. Por exemplo, para acrescentar uma propriedade "idade" em um banco de dados relacional, deve-se adicionar uma coluna à tabela, modificando assim a estrutura do banco de dados, ocupando talvez, espaços desnecessários e aumentando o tempo de resposta das requisições. Para a mesma situação, utilizando-se o Neo4J, apenas seria adicionada a chave "idade" ao registro desejado. Uma vantagem de utilização do Neo4J em relação a outros bancos de dados, inclusive NoSQL, é a garantia de atender as propriedades do ACID (Atomicidade, Consistência, Integridade e Durabilidade), já que os bancos do tipo NoSQL geralmente não possuem essa propriedade visando o ganho de performance[13].

Quando se trata de redes sociais, há diversas vantagens para a utilização de NoSQL em relação aos SGBDs relacionais. Especificamente, os sistemas NoSQL baseados em grafos possuem um desempenho significativamente melhor para executar buscas de relacionamentos entre dados cadastrados, diminuindo o número de requisições e comandos para encontrar dados relacionados. Para realizar buscas, em sistema relacional, deve-se navegar em diversas tabelas, muitas delas desnecessariamente, para encontrar todos dados desejados.

O Neo4J possui a linguagem de consulta própria, *Cypher* [12]. A ferramenta possui grande popularidade, maturidade, boa documentação e grande atividade da sua comunidade de usuários o que facilita a interação com outros usuários com dúvidas comuns, outra vantagem dessa ferramenta consiste na gama de ferramentas que possui integração com o Neo4J, fator que contribui com a riqueza e facilita o desenvolvimento. Além disso, o Neo4J possui uma implementação da sua API em Java.

O AllegroGraph é um banco de dados que foi desenvolvido pela empresa Franz, Inc. Ele trabalha com o sistema operacional Linux e é utilizado para armazenar dados em forma de RDF e é indicado para aplicações que envolvam Web Semântica. A consulta neste banco de dados pode ser feita por meio de duas linguagens: SPARQL e PROLOG [14].

Assim como no Neo4J, o ACID (Atomocidade, Consistência, Isolamento e Durabilidade) é garantido no AllegroGraph [14]. A principal diferença entre o Neo4J e o AllegroGraph é que o primeiro é ultizado, quando se deseja explorar as propriedades do grafo armazenado, ao passo que o segundo e utilizado quando de deseja explorar o universo da Web Semântica, com RDF e SPARQL.

O OrientDB é um SGBD orientado a grafos aberto lançado comercialmente em 2011 pela empresa OrienTechnologies e possui uma licença livre. O banco de dados foi implementado em Java e é um banco de dados transacional, além de dar suporte à arquitetura centralizada e distribuída com replicação. O OrientDB oferece grafos de propriedades para a modelagem lógica. O armazenamento físico dos dados pode ser feito em memória e em disco. Como todos os sistemas, ele usa a lista livre de adjacências para viabilizar o processamento nativo

das consultas, porém, diferentemente dos outros ele usa recursos de banco de dados de documentos e de orientação a objetos para o armazenamento físico dos vértices para garantir uma maior flexibilidade na estrutura dos dados a serem armazenados na base [15].

Uma grande vantagem do OrientDB é sua velocidade, sendo possível o armazenamento de até 150 mil registros por segundo em um *hardware* comum. O banco de dados oferece suporte para a linguagem SQL para consulta em sua base de dados [15].

Em relação ao Neo4J, a principal diferença do OrientDB é em relação ao tipo de banco de dados. O Neo4J é puramente orientado a grafos, ao passo que o OrientDB é um híbrido de orientação a documentos e a grafos, o que confere ao OrientDB maior velocidade transacional em relação so Neo4J [12] [15].

Ambos os SGBD apresentam versões *Open Source*. Consultando a documentação das APIs do OrientDB [15], e do Neo4J[12], constatamos que ambas permitem tanto a importação, quanto a exportação dos dados nela armazenados. Contudo, o manual de uso do Neo4J se mostrou mais específico e detalhado. A API do Neo4J também mostrou-se mais completa, possuindo mais funcionalidades (classes e métodos) do que a do OrientDB. Por ser também orientado a documentos, o OrientDB possui API específicas, como a voltada para grafos, o que faz com que ela como um todo seja muito completa.

Outro ponto analisado foram as comunidades de ambos, na qual foi observado que a comunidade do Neo4J é mais ativa do que a do OrientDB. Além disso, por estar há mais tempo no mercado, o Neo4J é um software mais maduro, cuja quantidade de informações (manuais, tutoriais, e exemplos) acerca de seu funcionamento é maior que a do OrientDB.

Em um *ranking* de popularidade divulgado no portal Db-Engines, pode-se observar que o Neo4J ocupa a 23º posição, ao passo que o OrientDB, ocupa a 68º posição [16]. Para o cálculo dessa medida de popularidade o portal levou em conta: o número de menções dos SGBDs em *web sites* (pesquisa em portais de busca); a frequência da busca por elas no Google Trends; o número de perguntas relacionadas a eles em sites como Stack Overflow e DBA Stack Exchange, muito utilizados por programadores; número de ofertas de emprego em que os SGBD são mencionados; o número de profissionais cadastrados no LinkedIn que declaram ter domínio sobre o SGBD; entre outras[16].

O OrientDB e Neo4J são desenvolvidos em Java e possuem suporte para APIs. As APIs (Interface de Programação de Aplicativos) são conjuntos de interfaces e classe, normalmente documentadas, disponibilizadas para o programador, que permitem ao desenvolvedor que foque na lógica do programa, e em regras de negócio. As API disponibilizadas por softwares prontos como os SGBDs levantados facilitam a utilização das suas funcionalidades por aplicações que não pretendem envolver-se em detalhes da implementação do software, e sim apenas

usar seus serviços. De modo geral, a API é composta por uma série de funções acessíveis somente por programação, e que permitem a utilização características internas do software, invisíveis ao usuário tradicional [12] [15].

2.2 FERRAMENTAS DE VISUALIZAÇÃO

A visualização dos grafos pode dizer muito a respeito das informações nele armazenadas. Para este trabalho objetiva-se possibilitar que o usuário possua uma ferramenta que ofereça a visualização de dados associada a predição de ligações. O mercado atualmente oferece várias ferramentas que possibilitam a visualização e manipulação de dados em forma de grafo, possibilitando a integração destas com outras ferramentas de modo a possibilitar o complemento das funcionalidades das ferramentas de visualização. Dentre as ferramentas com maior destaque e maior facilidade de integração com um SGBD estão o Gephi e o D3.js.

2.2.1 GEPHI E D3.JS

O Gephi é uma plataforma *open source* para a visualização e manipulação de grafos dinâmicos, incluindo todos os tipos de redes e sistemas complexos. Existe ainda a possibilidade de se adicionar filtros para ressaltar alguns aspectos que sejam mais importantes para o usuário e exportar o resultado final nos formatos SVG, PNG ou PDF. Além disso, a aplicação permite a busca funcionalidades adicionais, que podem ser incorporadas ao *software* original por meio de *plugins*. O Gephi possui uma comunidade bem ativa e há muito material sobre a ferramenta disponível para pesquisa na Internet. O Gephi é capaz de importar a exportar múltiplos formatos de arquivos como CSV, GML, DOT, entre outros, além de permitir a visualização dos grafos em três dimensões [17].

Através da plataforma NetBeans, pode-se desenvolver um *plugin* em Java para incorporar um algoritmo de predição de ligações aplicado a um grafo armazenado no sistema Gephi. Já existem *plugins* desenvolvidos por usuários que realizam tarefas semelhantes, utilizando outros algoritmos como o '*Markov Cluster Algorithm* (MCL)' que implementa o algoritmo de Markov, e o '*Link Communities*' que implementa um algoritmo que se propõe a revelar

comunidades em grafos não-direcionados sem peso nas arestas. O Gephi também pode ser utilizado somente como ferramenta de visualização, uma vez que permite que os dados sejam importados tanto de SGBD relacionais, quanto de SGBD orientados a grafos como Neo4J e OrientDB [17]. Algumas funcionalidade de Gephi são listadas a seguir:

- Análise de Redes Sociais;
- Análises de Redes Biológicas;
- Possibilita cálculo de métricas que envolvem grafos;
- Permite a integração de *plugins*

A respeito do D3.js trata-se de uma biblioteca baseada em JavaScript, tendo uma função muito bem definida: permitir ao desenvolvedor manipular documentos baseados em dados, usando padrões HTML, SVG e CSS. D3 significa "Data Driven Documents". Com a D3.js, é possível associar dados arbitrários a Modelos de Objeto de Documento (DOM) e aplicar transformações a qualquer documento. Por exemplo: é possível utilizar o D3.js para gerar uma tabela em HTML a partir de uma sequência de números, ou gerar gráficos baseados nos mesmos dados com transições animadas e recursos interativos. Pode-se ainda criar elementos SVG com o D3.js utilizando estilos CSS externos, com filtros e efeitos diversos [18].

Essa biblioteca permite que você manipule documentos baseados em dados usando padrões e navegadores abertos na Web. Isto significa que pode-se criar visualizações complexas sem depender de software proprietário. Pode ser encarada como uma alternativa clara ao Flash, com a grande diferença que todos os desenvolvimentos estão abertos e podem ser reutilizados por outros desenvolvedores [18].

2.3 PREDIÇÃO DE LIGAÇÕES

A predição de ligações consiste em prever um futura existência de uma ligação entre dois nós com base nos atributos dos outros vértices e das ligações observadas na rede [19]. Esse objetivo pode ser encarado de duas maneiras. A primeira é a predição de ligações faltantes

que podem ter sido perdidas durante o processo de construção da rede, o que em geral é aplicável nas redes criminosas e na World Wide Web. A segunda, é a predição de ligações que poderão existir no futuro, devido a evolução da rede no tempo, como acontece, por exemplo nas redes sociais [19].

Nesse último contexto, a definição de predição de ligações dada por [20] e que tem sido bastante usada na literatura ao servir como referência para diversos trabalhos é: dada uma região de algum domínio, a predição de ligações tenta prever com precisão quais ligações serão adicionadas a rede no intervalo de tempo entre t e um tempo futuro $t+1$.

O processo de predição de ligações pode-se basear em duas fontes de dados: as características dos membros da rede (conteúdo ou semântica), e as informações estruturais (topológicas) da rede. Embora sejam distantes, essas fontes de dados muitas vezes estão implicitamente relacionadas. Assim, diferentes métodos de predição de ligações utilizam tais fontes de dados para o cálculo de métricas que possam representar padrões de associação ou características dos elementos de uma forma de rede mensurável [21].

Segundo [21] as técnicas para resolver a predição de ligações podem ser divididas em três categorias: índices baseados na similaridade, métodos baseados ma máxima verossimilhança e modelos probabilísticos. Porém, existem também um grupo de métodos, chamados métodos híbridos, caracterizados pela combinação das propostas anteriores. Para este trabalho usaremos um dos métodos de predição de ligações baseados em similaridade que atribuem uma pontuação a pares de nós, baseados no grafo ao qual pertencem. Dessa forma, produzem uma lista ordenada na ordem decrescente de valores de pontuação. Assim, são obtidas medidas de proximidade ou de similaridade entre os nós a partir da topologia do grafo.

Métodos baseados em vizinhança de nós: alguns métodos de predição de ligações são baseados na premissa de que dois nós tem maior probabilidade de estar ligados no futuro se as vizinhanças deles possuim uma interseção grande.

Uma abordagem para resolver o problema de predição de ligações é utilizar índices baseados em similaridade. Nesta abordagem, consideramos um grafo G=(V,E), sendo V o conjunto de vértices e E o conjunto de arestas do grafo e para cada par de vértices (x,y) que não são ligados por uma aresta em E, é atribuída uma pontuação$S_{x,y}^{CN}$ a qual é definida como a similaridade (ou proximidade) entre x e y. Os pares de vértices com maior pontuação são tidos como ligações com alta probabilidade de existência.

A similaridade estrutural existente entre os vértices de uma rede é aproveitada para criar um grupo de índices chamados índices de similaridade estrutural (ou topológicas). Esses índices baseiam-se unicamente na estrutura da rede e, por sua vez, podem ser divididos em índices de similaridade local e global.

Os índices de similaridade local caracterizam-se por usar informações correspondentes ao par de vértices. Isso permite que o cálculo desses índices tenha um baixo custo computacional, embora o desempenho de predição possa muitas vezes não ser o melhor. Existem vários índices locais, no entanto para este trabalho usaremos o índice dos vizinhos comuns.

O índice dos vizinhos comuns ou *common neighbors,* notado CN foi proposto em [21] é definido da seguinte maneira: dois vértices x e y têm maior probabilidade de ter um futuro relacionamento se eles tem muitos vizinhos em comum. Assim, a pontuação é calculada da como:

$$S_{x,y}^{CN} = |\Gamma(x) \bigcap \Gamma(y)|$$

Sendo: $S_{x,y}^{CN}$ a similaridade entre os nós x e y ,

$\Gamma(x)$ grau de nó x e

$\Gamma(y)$ grau do nó y.

O índice de CN foi usado por [22] no estudo de redes sociais de colaboração, uma rede de co-autoria, apresentando uma correlação positiva entre os números de vizinhos comuns e a probabilidade de que dois cientistas possam colaborar no futuro. No algoritmo implementado, foi incluído a opção de o usuário dizer o número mínimo de vizinhos comuns a partir da qual é relevante a análise.

3 ESPECIFICAÇÃO DO PROTÓTIPO DE ARMAZENAMENTO, ANÁLISE, VISUALIZAÇÃO DE DADOS E PREDIÇÃO DE LIGAÇÕES

Esta seção apresenta a proposta do trabalho promovendo uma visão geral, por meio de diagramas e esquemas que possam mostrar o objetivo do desenvolvimento, bem como a contribuição para a comunidade científica.

3.1 VISÃO GERAL DO PROJETO E CONTRIBUIÇÕES

A ideia geral desse trabalho é aliar um Sistema de Gerência de Dados (SGBD), um Sistema Visualizador e Analisador de Grafos (SVAG) e a funcionalidade de predição de ligações. A aplicação final será o resultado da integração entre ferramentas já existentes e *plugins*.

Na aplicação devem existir dois tipos de usuários: o usuário administrador de banco de dados e o usuário analítico, o primeiro interage com o SGBD e faz cargas no banco de dados e o segundo tem como função interagir com o visualizador de grafos e requisitar a predição de ligações. Inicialmente o usuário administrador fará a carga da base de dados original. Esta base será armazenada em um banco de dados gerenciado por um SGBD, que além de garantir a persistência dos dados, facilita a realização de alterações no conteúdo armazenado e, consequentemente, na estrutura do grafo.

Dependendo do SGBD e da plataforma na qual se desenvolve a aplicação, mecanismos diferentes de integração podem ser utilizados, mas quaisquer que sejam estes, podem ser usados módulos de extensão (*plugins*) para importação do conteúdo da base para a aplicação, para predição de ligações e para a exportação.

Na importação e exportação os *plugins* têm como principal função garantir que nenhuma informação seja perdida ou alterada durante a importação dos dados. Dentro do *plugin* cria-se uma conexão entre o banco de dados e o visualizador, permitindo que os dados da base sejam importados para o visualizador por meio de uma cópia de dados, sem alterar o conteúdo original da base.

Uma vez que os dados estejam dentro do visualizador, eles são estruturados de maneira a facilitar a aplicação de um algoritmo de predição de ligações. O visualizador então executa o algoritmo gerando uma lista de arestas que foram inseridas entre determinados nós (ligações

previstas), criando assim um novo grafo.

Dentro do visualizador o usuário poderá estudar melhor o grafo com as ferramentas que este oferece, como visualização diferenciada de acordo com certas propriedades e cálculos de métricas, como medida de centralidade.

Por fim, o grafo pode ser exportado para a base de dados por meio do *plugin* de exportação, dessa forma, uma nova instância do banco de dados é criada a partir do novo grafo que contém as arestas previstas. A figura 3.1 ilustra a arquitetura da aplicação proposta.

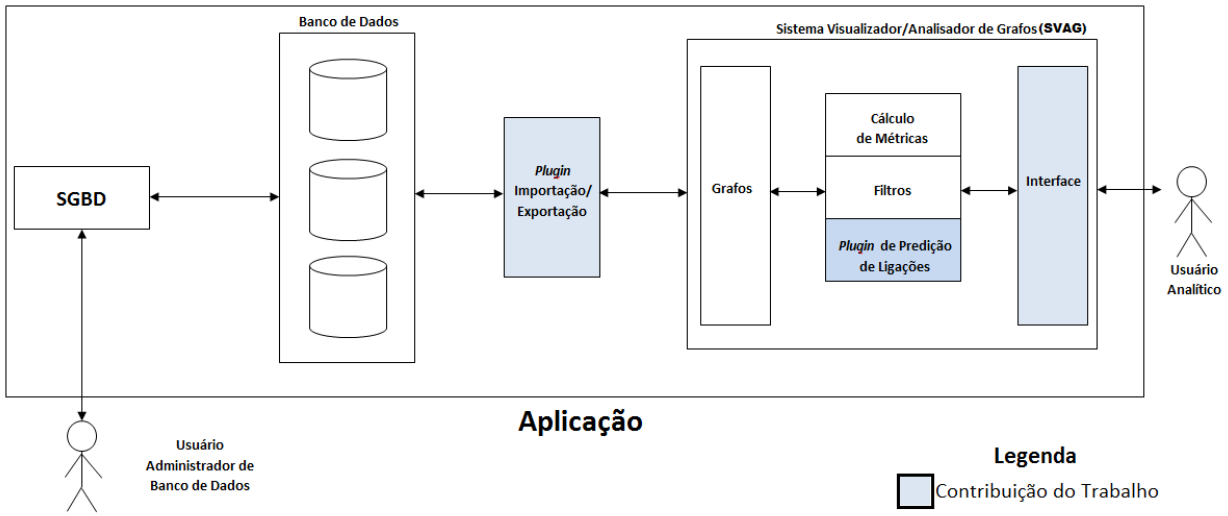

Figura 3.1: Visão geral do projeto

Como contribuições do trabalho destacamos o *plugin* de predição de ligações para um grafo qualquer, a integração da ferramentas de banco de dados e a da ferramenta de visualização, por meio da importação e exportação de dados e a interface com o usuário.

3.2 DETALHAMENTO DO FUNCIONAMENTO DA APLICAÇÃO

3.2.1 CASOS DE USO

Para aplicação temos os sequintes casos de uso:

1 - Importação da base.

Descrição Sucinta: Importar as informações da base de dados para o SVAG.

Atores: 1. Usuário analítico.

Pré-Condições: 1 .Ter o SVAG aberto.

Fluxo Básico:
1. O usuário analítico seleciona a opção importar informações.
2. O sistema exibe a lista de instâncias de banco de dados.
3. O usuário analítico seleciona a instância de banco de dados desejada.
4. O sistema fornece a lista de tipo de nós e arestas.
5. O usuário escolhe o tipo de informação que deseja importar.
6. O sistema importa os dados.
7. O caso de uso é encerrado.

2- Predição de Ligações.

Descrição Sucinta: Realizar a predição de ligações sobre um grafo.

Atores: 1. Usuário analítico

Pré-Condições: 1 .Ter executado o caso de uso importação da base.

Fluxo Básico:

1. O usuário requisita a predição de ligações.

2. O sistema exibe a janela de opções para definição do número mínimo de vizinhos comuns.

3. O usuário fornece o valor do número mínimo de vizinhos comuns.

4. O sistema realiza a predição de ligações.

5. O caso de uso é encerrado.

3- Exportar a base.

Descrição Sucinta: Exportar as informações do para uma instância de banco de dados.

Atores: 1. Usuário analítico.

Pré-Condições: 1. Ter o SVAG aberto.

Fluxo Básico:

1. O usuário requisita a exportação da informação para uma instância de banco de dados.

2. O sistema exibe a lista de instâncias de banco de dados.

3. O usuário escolhe a instância de banco de dados, para qual deseja exportar as informações.

4. O sistema exibe a janela para nomeação das arestas previstas.

5. O usuário nomea as arestas.

6. O sistema exporta as informações para a instância de banco escolhida.

7. O caso de uso é encerrado.

3.2.2 DIAGRAMA DE SEQUÊNCIA

A aplicação tem como principal função tornar possível a execução de predição de ligações a partir de um grafo e permitir a visualização das arestas previstas para o usuário. A figura 3.2 descreve o processo de importação das informações do banco de dados para ferramenta de visualização, onde são realizadas as análises e aplicada a predição de ligações.

Inicialmente o usuário seleciona na aplicação uma instância do banco de dados, sob a qual

ele deseja realizar a predição. As informações sobre o grafo são repassadas para o *plugin* de banco de dados (importação), que as repassa para o banco de dados, que por sua vez envia informações sobre as arestas e nós para o *plugin*, sendo estas repassadas para o visualizador, que as exibe ao usuário. Este seleciona qual parte do grafo deseja de fato importar, escolhendo os tipos de nós e de relacionamentos. Então, baseando-se nas escolhas do usuário analítico o *plugin* requisita a importação apenas daquelas arestas e nós cujo tipo foram selecionados, e por fim, a informação é disponibilizada no visualizador. Para o diagrama de sequência da figura 3.2 considerou-se banco de dados como objeto.

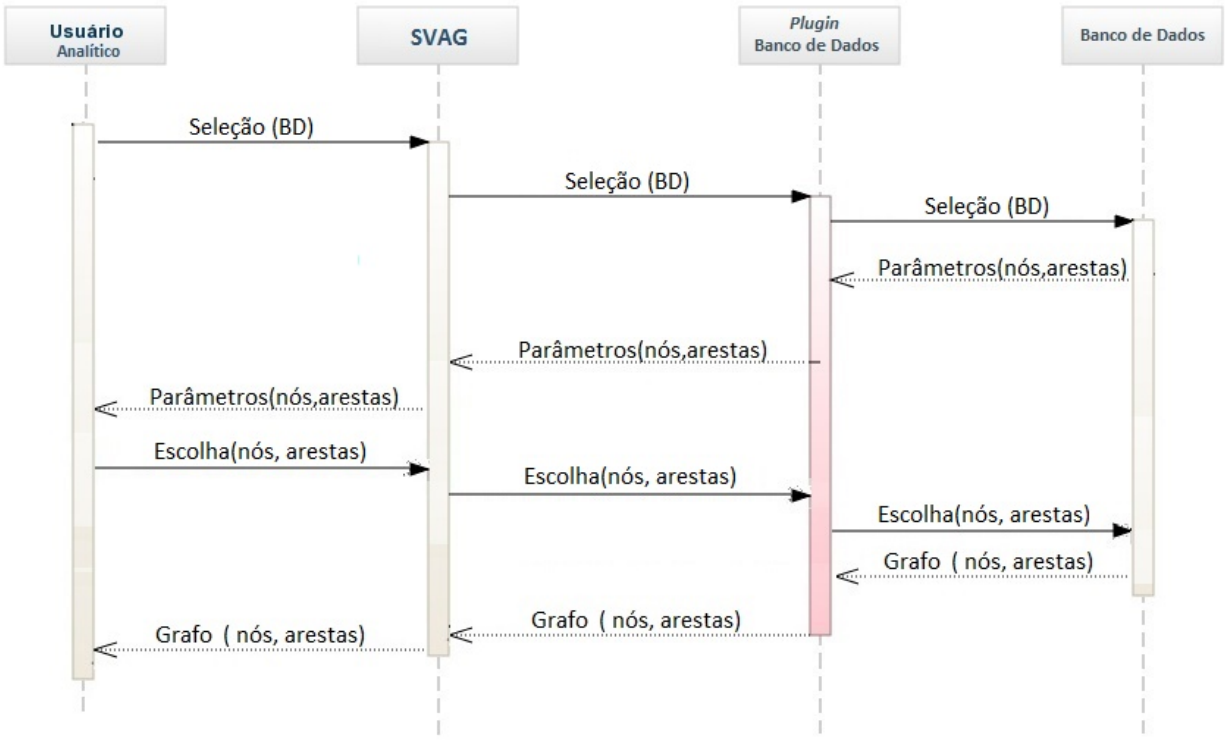

Figura 3.2: Diagrama de sequência da interação usuário analítico SGBD

Na figura 3.3, descreve-se o processo de predição de ligações. Tendo a representação do grafo no visualizador o usuário analítico pode solicitar a aplicação da predição de ligações. Uma vez solicitada a funcionalidade de predição, o visualizador repassa ao *plugin* de predição uma lista com os nós e arestas do grafo (parâmetros). O *plugin* aplica o algoritmo e retorna para o visualizador o grafo atualizado com os as ligações previstas. Para armazenar essa informação no banco de dados, o usuário requisita a exportação das informações. O visualizador repassa ao *plugin* de exportação os dados sobre o novo grafo, que por sua vez realiza a

exportação para o banco de dados. Para o diagrama de sequência da figura 3.3 considerou-se banco de dados como objeto.

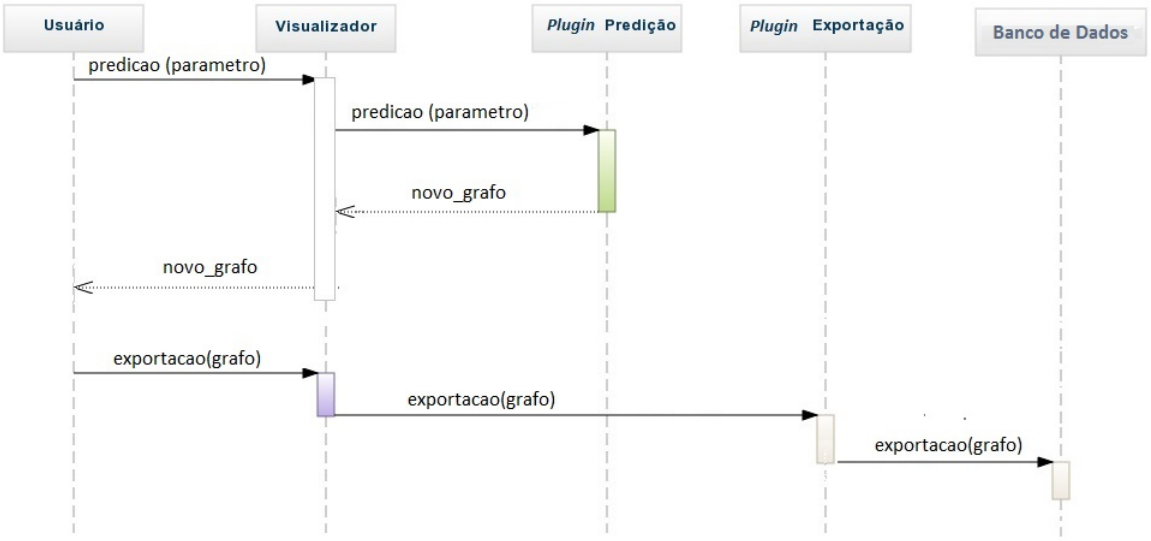

Figura 3.3: Diagrama de sequência do usuário e aplicação de predição de ligações

4 IMPLEMENTAÇÃO DO PROTÓTIPO

O projeto consiste em uma aplicação que visa integrar as funcionalidades de um SGBD e de um SVAG, e oferecer a função de predição de ligações. Este capítulo tem como objetivo mostrar o processo de desenvolvimento do trabalho, possibilitando sua reprodução ou continuidade posterior.

4.1 ESCOLHA DE FERRAMENTAS

Para escolha de ferramentas levou-se em consideração as informações da seção 2.1 deste trabalho, a qual faz uma breve explicação e diferenciação entre as ferramentas de banco de dados e de visualização pesquisadas para desenvolvimento deste trabalho.

De acordo com estas informações, descartou-se o AllegroGraph de início por armazenar os dados na foma de RDF e sobretudo pela limitação com relação ao sistema operacional, uma vez que trabalha apenas em Linux [14]. Em relação às outras duas ferramentas analisadas compilou-se as informações existentes apresentadas anteriormente, na tabela 4.1.

Critério	Neo4J	OrientDB
DB-ENGINES(Popularidade)	Posição no Ranking DB-ENGINES:23º	Posição no Ranking DB-ENGINES: 68º
Desenvolvedor	Neo Technology	Orient Technologies LTD
Lançamento	2007	2010
Linguagem de Implementação	Java	Java
Sistemas Operacionais	Linux ou Windows	Todos suportam Java VM
Modelo Lógico	SGBD Orientado a Grafos	SGBD Orientado a Grafos e Documentos
SQL	Não Suporta	Suporta
Java API	Possui	Possui
Java como linguagem de desenvolvimento	Sim	Sim
Comunidade	Ativa	Moderadamente Ativa
Documentação	Grande quantidade	Média quantidade

Tabela 4.1: Comparação Neo4J e OrientDB

Tendo como base critérios como quantidade de documentação disponível, número de usuários ativos, liguagem de implementação, tipo de licença e popularidade no *Ranking* Db-Engines[16], optou-se por utilizar o Neo4J, sobretudo pela quantidade de documentação existente a respeito da ferramenta e pela comunidade ativa.

Em relação às ferramentas de visualização comparando o Gephi com o D3.js, o segundo oferece apenas recursos visuais e não oferece possibilidade de manipulação dos dados como adição de filtros ou possibilidade de *plugins* para incrementar a ferramenta. Já o Gephi oferece a possibilidade de integração com o Neo4J, possibilitando novas formas visualização não oferecidas pelo interface gráfica do Neo4J. A tabela 4.2 fornece uma comparação entre as duas ferramentas.

Critério	Gephi	D3.js
Linguagem e Extensibilidade	Java e Permite a Adição de *Plugins*	JavaScript e permite modelagem da visualização
Integrabilidade	Permite integração com banco de dados e oferece grande quantidade de opções de visualização	Permite integração com banco de dados e oferece escassa quantidade de opções de visualização
Recursos	Oferece a possibilidade de filtros	Não oferece a Possibilidade de Filtros
Comunidade	Ativa	Moderadamente Ativa

Tabela 4.2: Comparação Gephi e D3.js

Com base nas pesquisas realizadas, optou-se por usar o Gephi como ferramenta de visualização, devido a sua facilidade de integração com o SGBD selecionado (Neo4J), pela quantidade de recursos visuais e pelo fato de possuir uma comunidade ativa, fato que facilita o desenvolvimento.

4.2 INTEGRAÇÃO DE FERRAMENTAS

Uma vez tendo selecionado as ferramentas que serão utilizadas, verificou-se qual plataforma melhor se adaptaria ao desenvolvimento do projeto, no quesito de integrabilidade com do SGBD e o SVAG. A primeira opção foi integrar o Neo4J com o Eclipse. Nesta opção utilizou-se a API do Neo4J para Java na plataforma de desenvolvimento Eclipse.

Nesta solução a base de dados foi exportada do Neo4J para uma aplicação a ser desenvolvida na plataforma Eclipse (recomendada pelo NetBeans) em Java.

A aplicação utilizou a API Java do Neo4J para acessar a instância do banco de dados a ser importada, informada a aplicação através de uma *string* dentro do código que aponta o diretório do Neo4J em que base está armazenada. Os dados da base são lidos da fonte, e estruturados dentro aplicação, mas não há nenhuma conexão ativa entre o Neo4J e a aplicação.

Ao término da manipulação dos dados pelo algoritmo, as novas arestas foram criadas sobre a estrutura que originalmente representava o grafo dentro da aplicação. Este novo grafo então foi exportado, e uma nova instância do banco de dados foi criada pela aplicação. Por meio de uma string, a aplicação aponta o diretório dentro do Neo4J em que a nova

instância do banco foi criada.

A segunda opção foi integrar o Neo4J e o Gephi. Nesta opção foi utilizada a API do Gephi para Java na plataforma de desenvolvimento NetBeans.

Nesta solução a base de dados foi exportada primeiramente do Neo4J para o Gephi, por meio de um *plugin*. Os dados sobre a base foram lidos da fonte, e armazenados no Gephi, que possui sua própria estrutura de armazenamento de grafos.

A aplicação de predição nesse caso poderia ser desenvolvida como um *plugin* do Gephi na plataforma NetBeans (recomendada pelo Gephi) em Java. Este *plugin* acessa o grafo armazenado no Gephi, que consiste em uma cópia importada do banco de dados, deixando os dados originais inalterados. Ao término da manipulação dos dados pelo algoritmo de predição, as novas arestas são criadas sobre a estrutura que originalmente representava o grafo dentro do Gephi. Este novo grafo estará automaticamente disponível dentro do Gephi.

O mesmo *plugin* utilizado para importar a base no Neo4J pode ser utilizado para exportar o grafo do Gephi para a base de dados.

O Gephi foi desenvolvido na linguagem Java utilizando-se o Netbeans para o desenvolvimento. A aplicação possui código aberto, oferecendo a possibilidade de edição do código da ferramenta diretamente no Netbeans podendo-se criar novas versões do programa com melhoras. Além disso, o Gephi apresenta a possibilidade da inserção de *plugins*, ou seja pode-se criar complementos à ferramenta e modelar o funcionamento desta de acordo com as necessidade do usuário.

Para comparar os modelos descritos acima os seguintes testes de integração foram realizados avaliando-se os seguintes critérios :

- A integração entre as ferramentas e as plataformas de desenvolvimento: em ambas as soluções não houve problemas na criação de um canal de comunicação entre o Neo4J e o Eclipse, o Neo4J e o Gephi, e o Gephi e o NetBeans;

- A criação de um novo grafo a partir do grafo original: ambas as soluções possuem ferramentas de exportação eficientes, contudo o *plugin* do Gephi é mais prático, pois encapsula muitas tarefas que devem ser feitas manualmente (ou encapsuladas pela ferramenta a ser desenvolvida);

- A adição de arestas resultantes da predição de ligações: ambas as soluções possuem essa funcionalidade;

- A persistência dos dados após as importações: no caso da solução com o Eclipse, a persistência é garantida tanto na importação quanto na exportação, já na solução com

o Gephi, há perda do tipo do nó, que não é comtemplado na estruturação interna do Gephi, mas as demais informações persistem;

- A entrada de parâmentos por parte do usuário, que podem ser valores numéricos, ou nós: no caso da solução com o Eclipse toda uma interface teria que ser desenvolvida para que houvesse qualquer interação com o usuário, contudo na solução com o Gephi, como há o desenvolvimento de um *plugin*, a interface do próprio Gephi pode ser utilizada como base;

- Informações sobre as API: ambas as API são bem documentadas quanto às funcionalidades das mais diversas classe, métodos e interfaces, contudo, a do Gephi possui um manual informando quais são as etapas do desenvolvimento de um *plugin*, que informa ao usuário quais interfaces são essenciais na implementação deste, entre outras informações que facilitam a construção da aplicação por parte do desenvolvedor;

- Ferramentas de simulação: a solução do Gephi emula a própria ferramenta de dentro do NetBeans. Ao executar o código do *plugin* em desenvolvimento (a partir do NetBeans), com a API do Gephi é possível simular a interação entre o Gephi, Neo4J e *plugin*. Já na solução com o Eclipse, não há como testar a interação com o Neo4J dinamicamente;

- A quantidade de exemplos de aplicações/*plugin* utilizando as mesmas APIs disponíveis: não foi possível encontrar muitas aplicações que seguissem a arquitetura proposta pela solução que utiliza o Eclipse. Já a própria ferramenta Gephi possui uma plataforma Web voltada exclusivamente para o desenvolvimento de *plugin*, que possui desde tutoriais, a diversos exemplos de *plugins* com as mais diversas funcionalidades, e cujo código fonte está disponível na Web (via Github) para qualquer desenvolvedor.

Ao final obtou-se pela segunda solução já que a quantidade de documentação encontrada a respeito foi maior, bem como observou-se uma comunidade mais ativa nas ferramentas referentes à segunda solução.

Em relação à integração entre a ferramenta de banco de dados e o visualizador observou-se que os dados são representados de maneiras diferentes no Gephi e no Neo4J [23][24].

Para o Neo4J o modelo de grafo consiste em Nós, Relacionamento e Propriedades. Relacionamentos representam arestas entre os nós. Propridedades são pares de valores chave e podem armazenar informações. As propriedades são sempre *strings*, logo não são aceitos valores arbitrários ou nulos. Já os nós e relacionamento possuem id própria, a qual é atri-

buída pelo próprio Neo4J no momento da criação. A presença de propridedades em nós e em relacionamentos é facultativa [23].

O Gephi, em sua representação de grafo possui os seguintes itens: arestas, nós e atributos, além disso o grafo ainda pode ser estruturado de duas formas: modelo atributo e modelo grafo. No modelo grafo, apenas manipulações de adição e remoção de nós e arestas podem ser feitos. Já no modelo atributo, atributos podem ser criados e manipulados [24].

No Gephi as arestas são responsáveis por conectar nós. Os nós e as arestas são representados no modelo atributo por meio de tabelas, cada nó ou aresta possui seus atributos armazenados em uma outra tabela, possuindo uma linha de tabela para cada atributo. O Gephi oferece um *plugin* de integração com o Neo4J. O objetivo deste *plugin* que pode ser baixado e instalado pelo próprio Gephi é criar uma cópia dos dados do Neo4J de modo que os dados possam ser visualizados e analisados de forma mais detalhada no Gephi. No entanto, a forma como os dados são encarados no Neo4J e no Gephi são diferentes, logo faz-se necessário um mapeamento dos dados para que não haja perda de informação durante o processo de importação dos dados [23][24].

Para observar como os dados são mapeados do Gephi para o Neo4j pode-se observar a representação fictícia de um grafo no Neo4J. Os círculos representam os nós e os relacionamentos são as setas direcionais. As propriedades, seja dos nós ou relaciomentos estão representadas dentro dos retângulos [23].

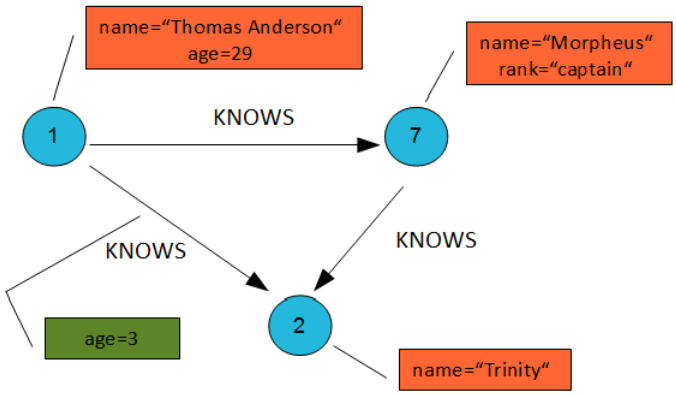

Figura 4.1: Exemplo de representação de um grafo no Neo4J.

As tabelas 4.1 e 4.2 mostra, a representação do grafo no Gephi.

Aresta(String)	**Idade(int)**
”KNOWS”	-
”KNOWS”	-
”KNOWS”	3

Tabela 4.3: Atributos das arestas no Gephi.

Id(int)	**Name(String)**	**Rank(String)**	**Age(int)**
1	”Thomas Anderson”	-	29
2	”Morpheus”	”Capitan”	-
7	”Trinity”	-	-

Tabela 4.4: Atributos dos nós no Gephi.

Primeiramente os nós são processados. Para cada nó no Neo4J deve existir um nó no Gephi e para cada relacionamento no Neo4J cria-se uma aresta no Gephi. Por fim, para cada propriedade do grafo Neo4J, deve existir no Gephi uma coluna com o nome atributo e seu respectivo valor.

A figura 4.2 e 4.3 juntamente com as tabelas 4.3 e 4.4 mostram o processo de conversão das informações contidas no Neo4J para o Gephi.

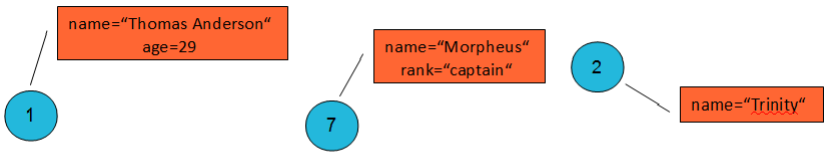

Figura 4.2: Nós e propriedades no Neo4J.

Id(int)	Name(string)	Rank(string)	Age(int)
1	”Thomas Anderson”	-	29
7	”Morpheus”	”capitan”	-
2	”Trinity”	-	-

Tabela 4.5: Dados dos nós no Gephi.

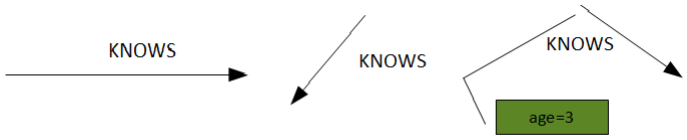

Figura 4.3: Relaciomentos no Neo4J.

EdgeType	Age(int)
”KNOWS”	-
”KNOWS”	-
”KNOWS”	3

Tabela 4.6: Dados dos relacionamentos no Gephi.

A arquitetura do Gephi é modular e cada um de seus recursos está implementada em módulos separados. Estes módulos por sua vez dependem uns dos outros, similar aos pacotes Java. Para se desenvolver uma nova funcionalidade, pode-se simplesmente criar novos códigos e adicionar como dependência de outros módulos do Gephi [24].

A figura 4.4 apresenta um esquema da aquitetura do Gephi. No primeiro nível observa-se a linguagem (Java) usada para desenvolvimento. No segundo nível tem-se a plataforma usada para a criação do programa, no terceiro e quarto nível observam-se as APIs do Gephi e seus *plugins*, o quais acrescentam funções adicionais à aplicação [24].

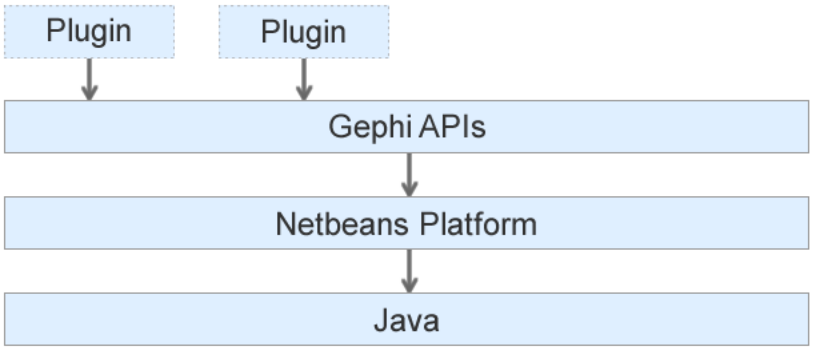

Figura 4.4: Arquitetura Gephi

A razão pela qual se decidiu por utilizar um *plugin* par realizar a tarefa de predição de ligações foi porque *plugin* ou módulo de extensão é um programa de computador usado para adicionar funções a outros programas maiores, provendo alguma funcionalidade especial ou muito específica. Geralmente é pequeno e leve.

Uma aplicação pode utilizar tal técnica por diversos motivos, como permitir que desenvolvedores de software externos estendam as funcionalidades do produto, suportem funcionalidades antes desconhecidas, reduzam o tamanho do programa ou, até mesmo, separem o código fonte de diferentes componentes devido à incompatibilidade de licenças de software [25].

No Gephi, podemos ter *plugins* do tipo filtro, o qual gera um subgrafo a partir do grafo carregado em memória, respeitando as condições definidas pelos usuário. Criando um paralelo com banco de dados relacionais o filtro do Gelphi funciona como uma *view* e mostrando apenas os dados que interessam ao usuário, condição que auxilia na análise de um grande volume de dados [26].

4.3 *PLUGIN* PREDIÇÃO DE LIGAÇÕES E FILTRO PARA BASE DE DADOS

O *plugin* de predição de ligações terá como função prever futuras ligações em uma rede de grafos, criando novas conexões entre os nós de acordo com o algoritmo descrito na seção 2.3 deste trabalho. Como dito anteriormente o princípio dos vizinhos comuns diz que: quanto maior o número de vizinhos comuns entre dois nós, maior a probabilidade de haver uma futura conexão entre eles. Para efeito de otimizar o algoritmo, usou-se um parâmetro para que o usuário entre com um valor mínimo de vizinhos comuns a partir do qual a predição será considerada. Baseando-se nessas premissas o código segue a sequência para gerar a predição de ligações.

- O usuário define a quantidade mínima de vizinhos em comum entre dois nós, para realização da predição de ligações.
- O *plugin* computa a quantidade de vizinhos em comum entre cada par de nós.
- O *plugin* conecta os nós que possuem um número maior ou igual de vizinhos comuns do que o mínino estabelecido pelo usuário.
- O *plugin* devolve o novo grafo, já com as novas arestas.

Na figura 4.5 pode-se ver o fluxograma que representa o processo de predição de ligações.

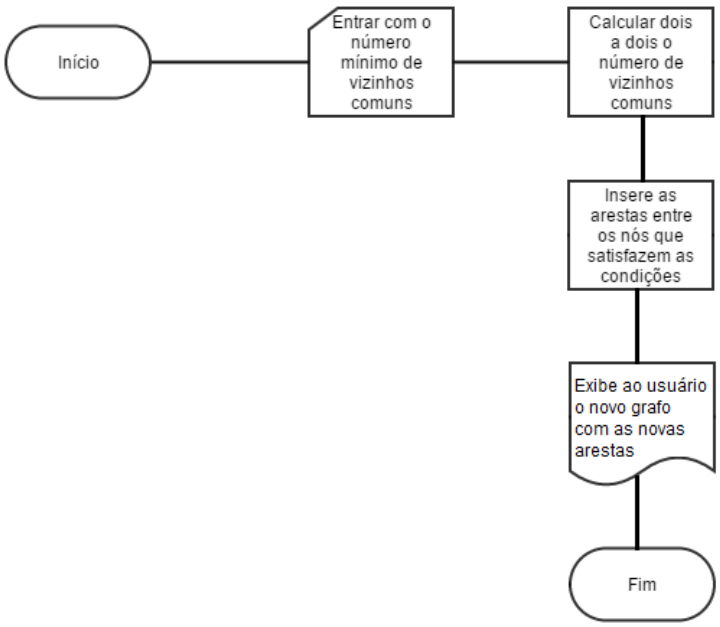

Figura 4.5: Fluxograma da predição de ligações

O desenvolvimento do *plugin* de predição de ligações foi realizado na linguagem Java, fazendo uso da plataforma NetBeans, seguindo o algoritmo proposto.

Objetivando-se a realização dos testes de implementação do *plugin*, seria necessário a cada modificação no código a criação de um novo arquivo para instalação no SVAG, no caso o Gephi. A melhor forma encontrada para evitar que a cada alteração fosse necessária uma nova instalação foi emular o Gephi dentro do Netbeans de modo que o *plugin* em desenvolvimento pudesse ser testado imediatamente a cada modificação.

Para base de dados de teste utilizou-se um pequeno exemplo sob o qual havia-se previsibilidade de modo a verificar se a predição estava sendo realizada de maneira adequada. Tendo chegado a uma versão estável do *plugin*, este foi instalado no Gephi e testado novamente para ratificar o funcionamento em um ambiente real.

O processo de seleção dos dados é realizado pelo *plugin* de importação e exportação de dados entre Neo4j e Gephi, e tem como função além de importar e exportar, ser capaz de transformar o estrutura de grafo do Neo4j para uma estrutura de grafo que possa ser lida e modificada no Gephi e posteriormente voltar a ser armazemazenado em uma instância de banco de dados. Como ambos representam o grafo de maneiras diferentes, como citado na

seção 4.2 deste trabalho, primeiramente realizou-se o mapeamento entre os itens do grafo do Neo4j e itens do grafo Gephi e em seguida a conversão de um formato para o outro. Para este desenvolvimento adaptou-se um *plugin* de importação e exportação já existente de modo que atendesse as necessidades do projeto. Esta adaptação do *plugin* será explicada mais adiante.

O processo de importação do grafo consiste basicamente em 2 abordagens: importar todo o grafo ou apenas parte do grafo. A importação de todo o grafo, como sugere o nome, traz para o Gephi o grafo completo. A desvantagem desse tipo de importação consiste de se deixar a interface do Gephi lenta ou de se poluir a análise da estrutura do grafo, já que a quatidade de informações de um grafo pode ser muito grande. Já a opção de importação de parte do grafo, permite importar apenas parte da estrutura. Neste caso, a seleção é feita por meio das propriedades presentes nos nós e por meio dos tipos de relacionamentos presentes na base de dados. O valor da propriedade inserida pelo usuário é comparado ao valor da propriedade dos nós presentes no grafo usando operador escolhido pelo usuário para efeito de comparação. Os operadores de filtragem são combinados usando a operação lógica "E", a lista de operadores oferecidos são: = (igual a), < (menor que), <= (menor ou igual a), > (maior), >= (maior ou igual a) e ! (diferente de). Um exemplo de uso das operações para filtragem seria supondo que alguns nós possuíssem a propriedade idade e quiséssemos todos nós que possuem essa propriedade menor que 38. Então, teríamos a condição idade <38, o que traria para o visualizador apenas nós que respeitassem esta condição.

Já no caso da seleção da base pelo tipo de relacionamento, as opções de relacionamentos contidos na base são expostas para o usuário, o qual as seleciona. Este recurso pode ser usado juntamente com a seleção feita por meio das propriedades dos nós. Um exemplo de uso seria caso o grafo possuísse relacionamentos do tipo "Publicou_Com" e "Publicou". Caso fosse selecionado o tipo "Publicou_Com", seriam trazidos apenas nós que se relacionassem por meio deste tipo de relacionamento. Os filtros por operações e por relacionamentos se somam, ou seja, uma vez aplicado um deles o outro será aplicado sobre o grafo obtido pelo primeiro filtro.

No processo de exportação é feito o processo inverso ao importador. O modelo de grafo do Gephi é convertido para o modelo de grafo do Neo4J de modo a garantir a consistência dos dados e a manutenção das informações. Durante o processo de exportação oferece-se a opção de nomear as arestas recém criadas no processo de predição de ligações de modo que elas se tornem um novo tipo de aresta.

No entanto, durante o desenvolvimento do trabalho o *plugin* de importação e exportação disponibilizado *online* por [27] apresentou problemas de funcionamento, permitindo apenas a importação da base completa para o visualizador. Após um trabalho de investigação

descobriu-se que a causa do não funcionamento do *plugin* disponibilizado era uma mudança feita na estrutura do Neo4J.

O princípio de funcionamento do *plugin* importação/exportação se pautava em uma biblioteca aberta do Neo4J chamada *NeoStore*, a qual permitia que o *plugin* de importação selecionasse um nó aleatório dentro da base de dados e a partir deste buscasse em toda base os tipos de relacionamento contidos nesta e trouxesse para o usuário em forma de lista para que este pudesse selecionar a informação desejada. No entanto, a partir do fim de 2014 a biblioteca em questão foi fechada, por meio de um *upgrade* da versão gratuita do Neo4j, invibializando o funcionamento do *plugin* já existente.

A solução encontrada para estes problemas foi modificar o *plugin* de importação orginal e toda a parte que dependia da biblioteca *NeoStore* de modo que este permitisse que o usuário inserisse uma identificação do nó por onde deveria começar a buscar pelas informações do grafo para que estas pudessem ser trazidas para o usuário analítico.

Atualmente o *plugin* disponibilizado na rede não funciona devido às modificações feitas na biblioteca do Neo4J, a qual é usada no processo de importação. Logo, pode-se citar o *plugin* de importação, refatorado durante o desenvolvimento, como um contribuição para a comunidade científica.

A seguir temos um diagrama de *storyboard* que mostra como a aplicação se comporta. Na figura 4.6 podemos observar o processo de importação da base de dados para a aplicação. O primeiro passo consiste em solicitar a importação dos dados e o segundo passo em escolher a base a ser importada.

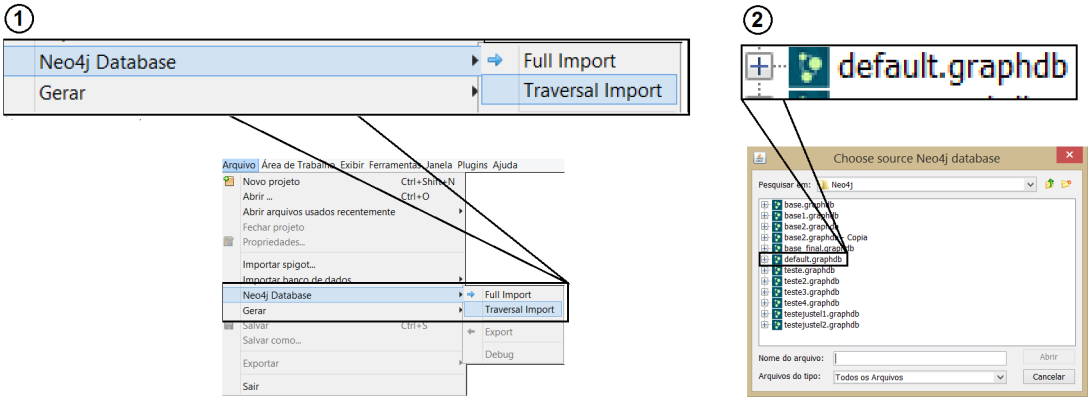

Figura 4.6: Importação da Base

Caso o usuário deseje filtrar a base de dados para trabalhar apenas o tipo de informação desejada ou para otimizar o desempenho da aplicação, evitando carregar dados que não serão utilizados na predição, ele possui as opções de selecionar os dados de acordo com propriedades

dos nós ou pelo tipo de relacionamento contido na base. Para utilização destes recursos faz-se necessário a inserção da identificação de um nó inicial, o qual servirá como referência para busca e seleção da base de dados. O processo de seleção pode ser observado na figura 4.7 .

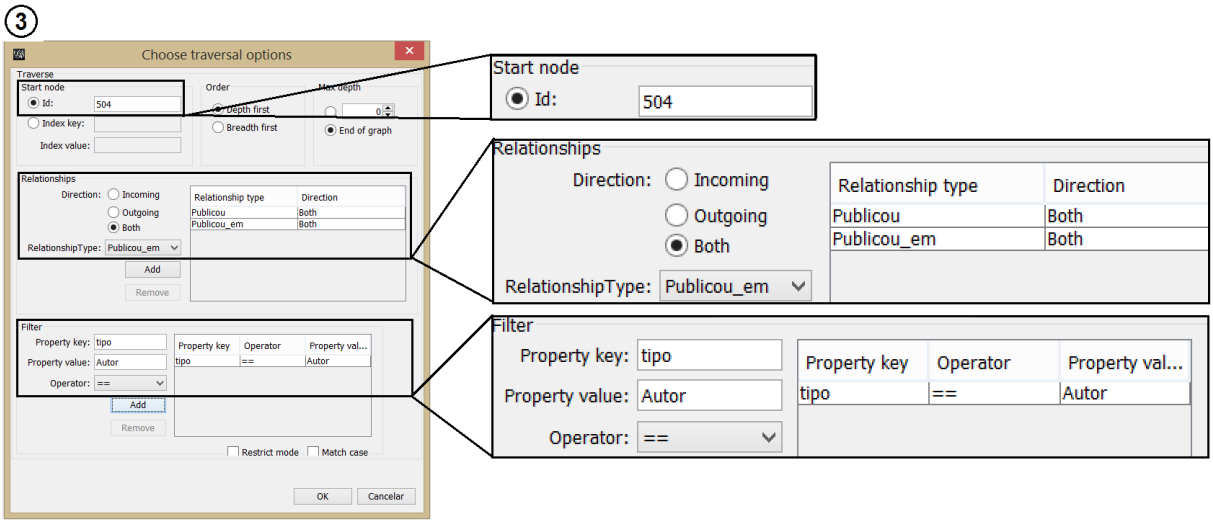

Figura 4.7: Seleção de dados para importação

Os dados podem ser visualizados no Sistema Analisador e Visualizador de Grafos de duas formas: por meio da representação gráfica de um grafo como pode ser observado no passo 4 da figura 4.8 ou por meio de listas, que consistem em uma lista de nós e uma lista de arestas como pode ser observado no passo 5 da figura 4.8.

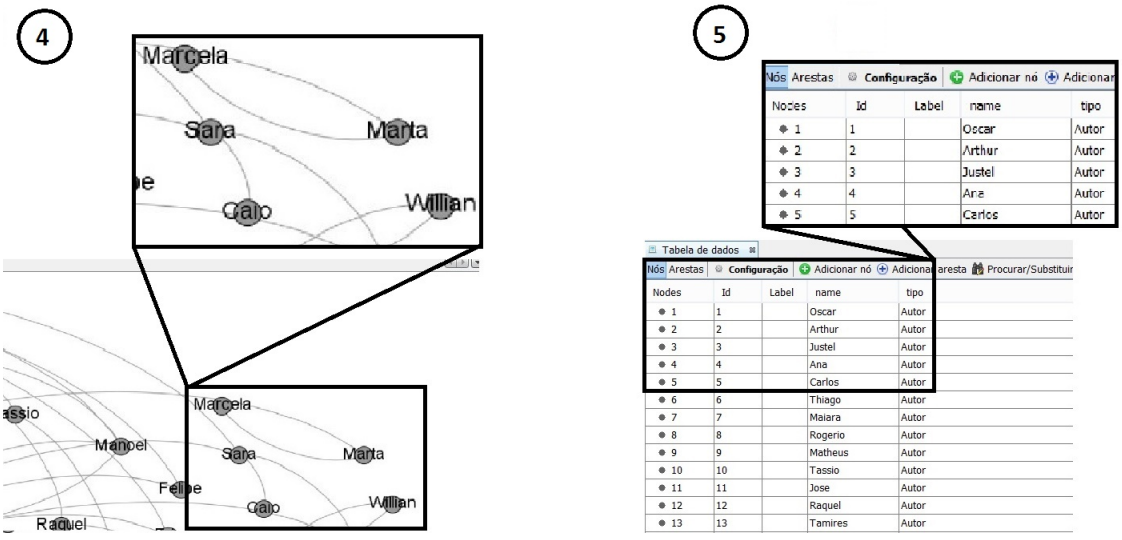

Figura 4.8: Base importada

A aplicação da predição de ligações será realizada somente se o grafo já estiver carregado no Gephi. Neste processo se faz necessário que o usuário forneça o número mínimo de vizinhos comuns entre os nós, como mostra o passo 7 da figura 4.9 . Este parâmetro foi introduzido pensando em trazer resultados mais relevantes ao usuário, já que muitas vezes deseja-se que a predição seja feita apenas a partir de uma certa quantidade de vizinhos comuns.

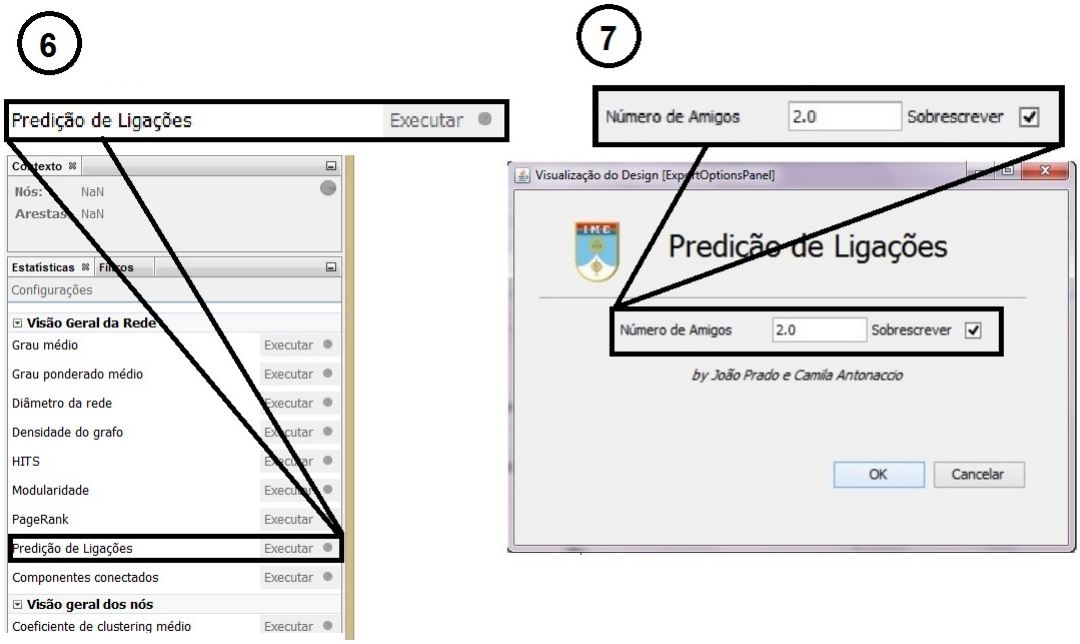

Figura 4.9: Predição de Ligação

Na figura 4.10 pode-se observar o resultado de uma predição de ligações, dois nós que anteriormente não estavam conectados agora estão, devido à predição de ligações.

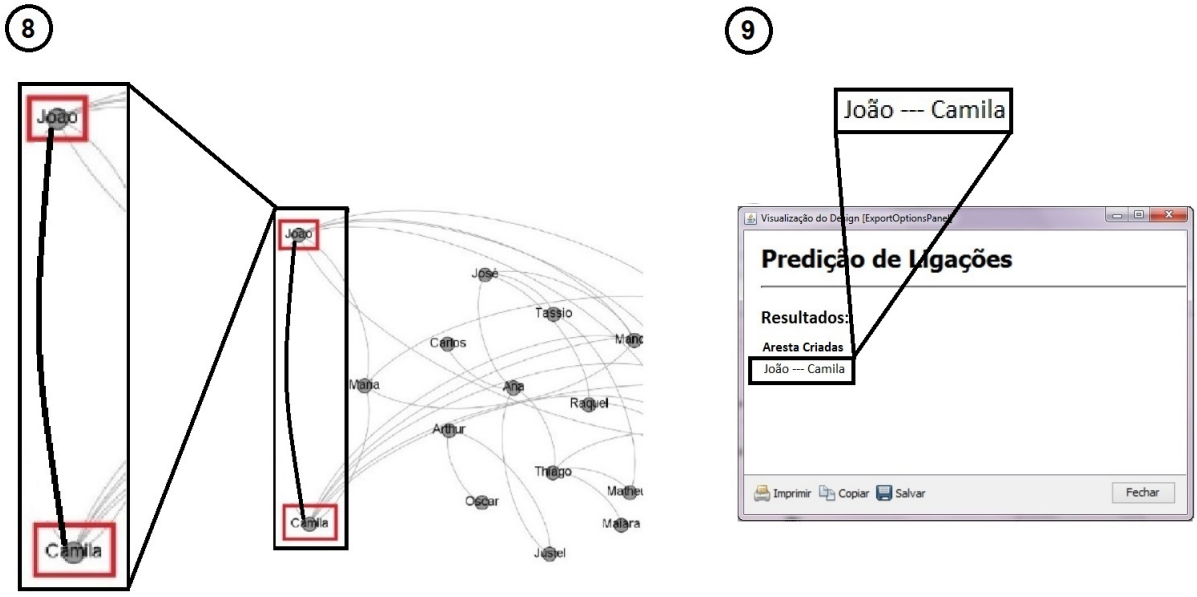

Figura 4.10: Resultado da predição de ligações

No processo de exportação da base para o banco de dados, o *plugin* terá o papel de converter os dados do formato do Gephi para o formato do Neo4J, nesta fase existe a opção de se nomear as arestas criadas durante a predição de ligações, esta função é ilustrada no passo 10 da figura 4.11 .

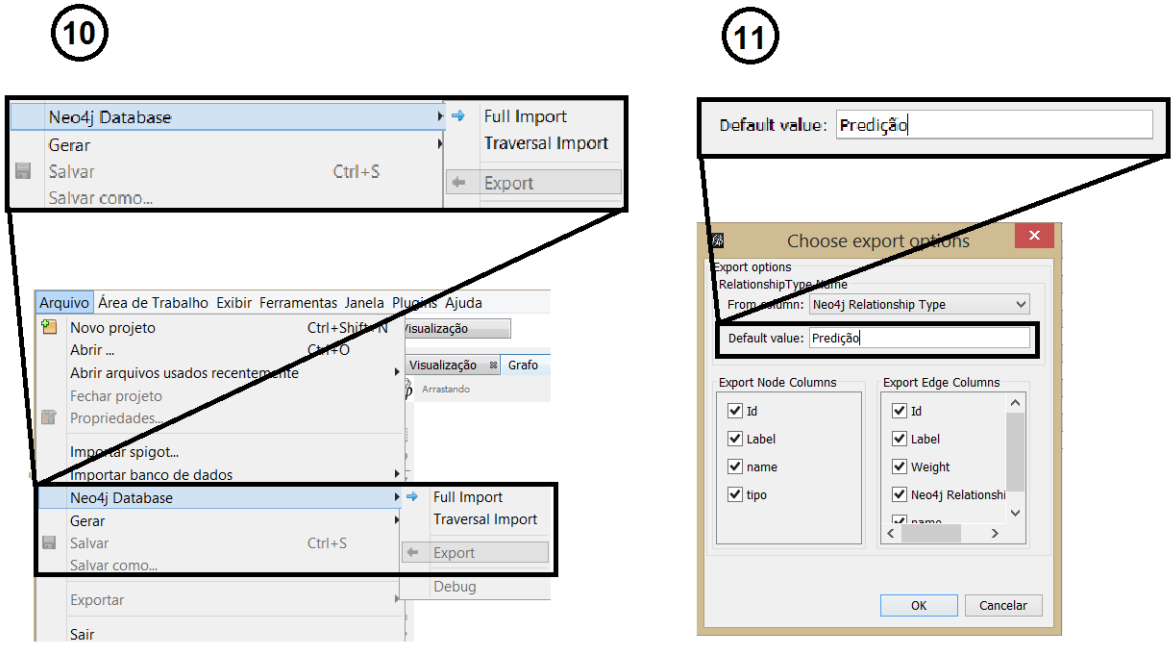

Figura 4.11: Exportação da base

4.4 ESTUDO DE CASO

Para o estudo de caso usou-se uma base de dados criada a partir de dados reais, os quais foram extraídos de informações provenientes do CNPq e anteriormente usadas em [28] a respeito do autores, co-autores, título do artigo publicado, veículo publicado e ano de publicação. No caso da base criada, foram selecionados três autores e suas respectivas publicações realizadas até o ano de 2010.

A modelagem das informações para o formato de grafo foi feita de modo a se ter variedade de tipo de nós, neste caso foram criados três tipos de nós: nó tipo autor, tipo publicação e tipo veículo. O primeiro possui como propriedades nome do autor e tipo do nó, o segundo apresenta o nome da publicação realizada e tipo do nó e o último contém o nome do veículo no qual o artigo foi publicado e tipo do nó. Os nós do tipo autor se conectam entre si, caso tenham pelo menos uma publicação em comum, por meio de uma aresta do tipo Publicou_Com, os nós do tipo autor e publicação se conectam por meio de uma aresta do tipo Publicou, no caso do autor ter participação na publicação e por fim os nós publicação se relacionam com os nós do tipo veículo por meio das arestas do tipo Publicou_Em.

Como citado na seção 4.2 deste trabalho, o Neo4J possui uma tipagem para cada tipo de nó, a qual pode ser inserida pelo usuário administrador no momento da criação da base, no entanto esta tipagem não é reconhecida pelo Sistema Visualizador e Analisador de Grafos (Gephi), por isso inseriu-se esta informação como propriedade de cada nó, o que possibilitou a permanência da informação durante o processo de importação e exportação. O esquema de modelagem da base em grafos pode ser visto na figura 4.12.

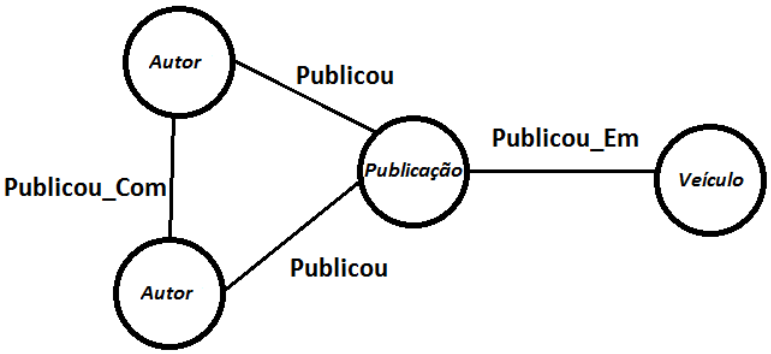

Figura 4.12: Esquema de modelagem da base de dados

Para verificação do funcionamento da aplicação deve-se testar todas as funcionalidades

implementadas, além da consistência da informação no processo de importação e exportação, para isso as seguintes funcionalidades devem ser testadas:

- Importação da base completa;
- Importação da base utilizando filtros de tipo de nó e tipo de relacionamento;
- Algoritmo de predição de ligações;
- Exportação da base.

Visando verificar as funcionalidades citadas os seguintes testes foram executados:

- Importação de todos os dados da base, realização da predição de ligações, verificação da corretude da predição e exportação dos dados atualizados;
- Importação de dados filtrados por tipo de nó, realização da predição de ligações e exportação dos dados filtrados atualizados;
- Importação de todos os dados de uma base modificada, na qual foi removida uma ligação de grande importância para verificação a capacidade do *plugin* de recuperar esta aresta.

No caso da predição de ligações a partir do grafo gerado com a base completa, foram obtidos os resultados esperados. Foi obtida a ligação entre um veículo de publicação e um autor (nós do grafo). Posteriormente, foi atualizada a base de dados, exportando os dados da nova ligação. No exemplo, sugeriu-se que o autor Figueiredo, C. M. H. publicasse no veículo *Discret Applied Mathematics*, pelo fato deste autor ter publicado com outros 8 autores que já publicaram neste veículo, o resultado da predição de ligações pode ser observado na tabela 6. No processo de exportação pode-se inserir um tipo para arestas criadas durante o algoritmo de predição, os dados exportados são inseridos em uma nova instância de banco de dados, garatindo o teste fim a fim dos dados. Podemos observar que esta ligação não está de acordo com a modelagem de grafo proposta neste trabalho, no entanto isto pode ser entendido como uma sugestão de veículo para publicação de um autor ou como um veículo de preferência, ou seja, como a criação de um novo tipo de relacionamento na base de dados. O resultado obtido pode ser observado na tabela 4.7 e na figura 4.13.

Autor	Veículo
Figueiredo, C. M. H.	*Discret Applied Mathematics*

Tabela 4.7: Resultado das Predição de Ligações para base completa

Na figura 4.13 pode-se observar o resultado da predição de ligações para os dados importados da base completa. No passo 1 observam-se os dados antes da predição de ligação e no passo 2 pode-se observar a ligação criada após a predição de ligações.

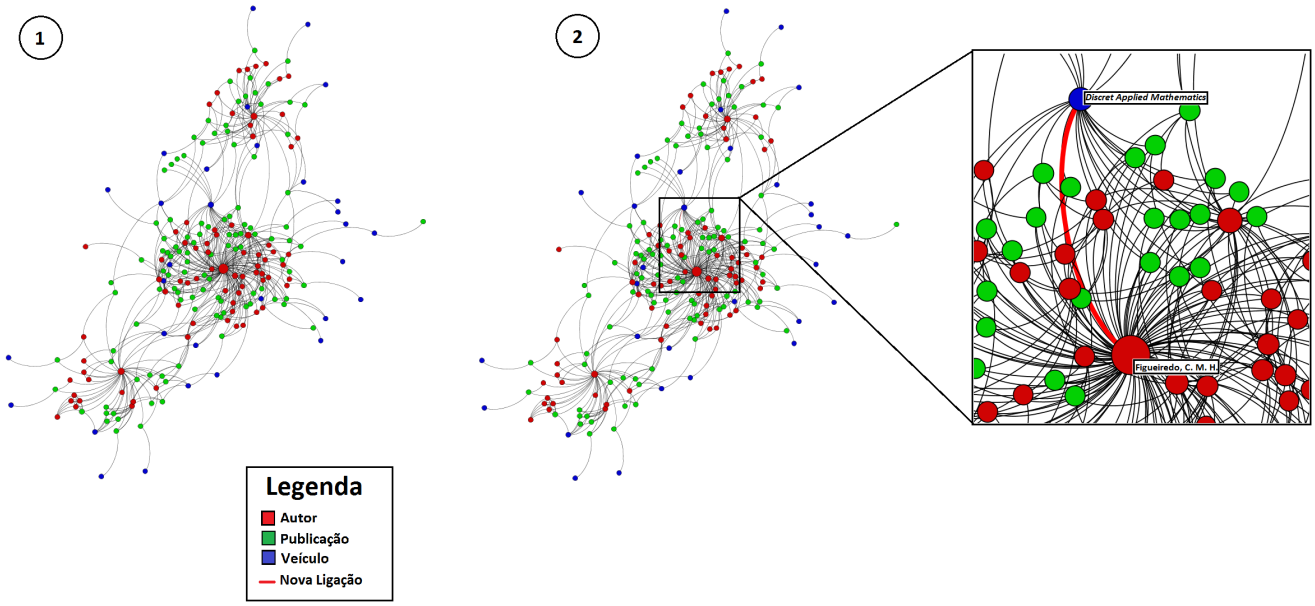

Figura 4.13: Resultado da predição de ligações

A predição de ligações pode ser utilizada para recuperação de ligações ausentes. Visando testar esta funcionalidade modificou-se a base original removendo-se uma ligação. O resultado obtido foi a recuperação da ligação removida, sendo criada uma aresta entre autores que possuíam 16 co-autores em comum. A figura 4.14 mostra o grafo antes e depois da inserção da aresta obtida pelo algoritmo de predição de ligações, e a tabela 4.8 identifica o par de autores sugeridos pelo algoritmo de predição.

Autor	Autor
Figueiredo, C.M.H.	Faria, L.

Tabela 4.8: Resultado da predição de ligações para aresta ausente

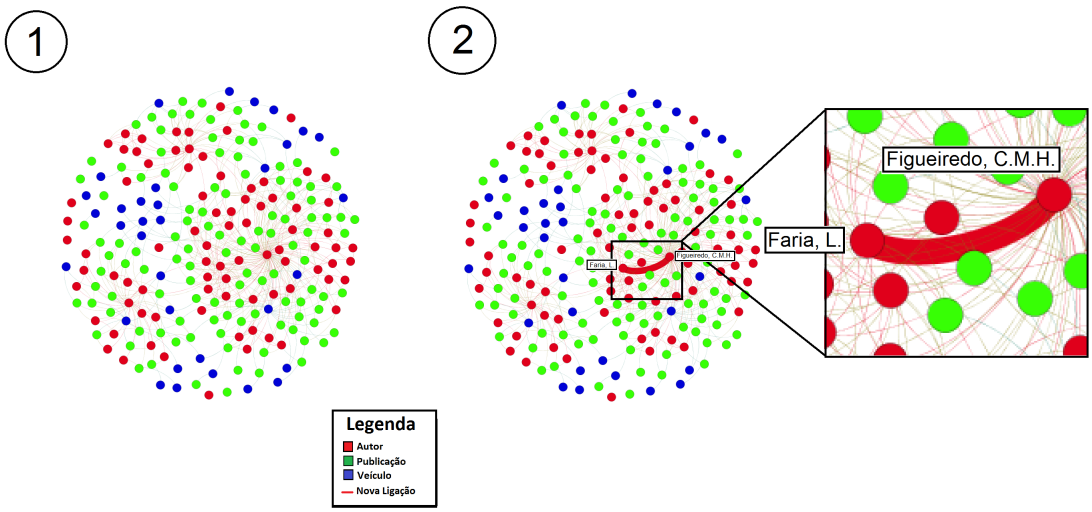

Figura 4.14: Resultado da predição de ligações para recuperação de aresta

Quando a predição de ligações é feita utilizando algumas informações da base de dados, ou seja o caso da importação utilizando os filtros, escolheu-se importar os nós do tipo autor devido a maior quantidade de dados deste tipo na base. O resultado obtido na predição foi de acordo com o esperado. Neste teste foram criadas arestas para os pares de autores que possuíam 3 co-autores em comum. A figura 4.15 mostra as arestas criadas neste experimento, e a tabela 4.9 identifica os pares de autores sugeridos pelo algoritmo de predição. No processo de exportação é possível inserir um tipo para arestas criadas pelo algoritmo de predição, os dados exportados são inseridos em uma nova instância de banco de dados, garatindo o teste fim a fim dos dados.

Autor	Autor
Lucchesi, C.L.	Kawarabyashi, K.
Lucchesi, C.L.	Figueiredo, C.M.H.
Alcon, L.	Gutierrez, M.
Dantas, S.	Faria, L.

Tabela 4.9: Resultado da predição de ligações com filtro de nós tipo autor

Mais especificamente, na figura 4.15 pode-se observar o resultado da predição de ligações para os dados importados da base filtrada antes da importação, neste caso importou-se apenas os nós do tipo autor. No passo 1 observam-se os dados antes da predição de ligação, no passo 2 pode-se observar a ligação criada após a predição de ligações.

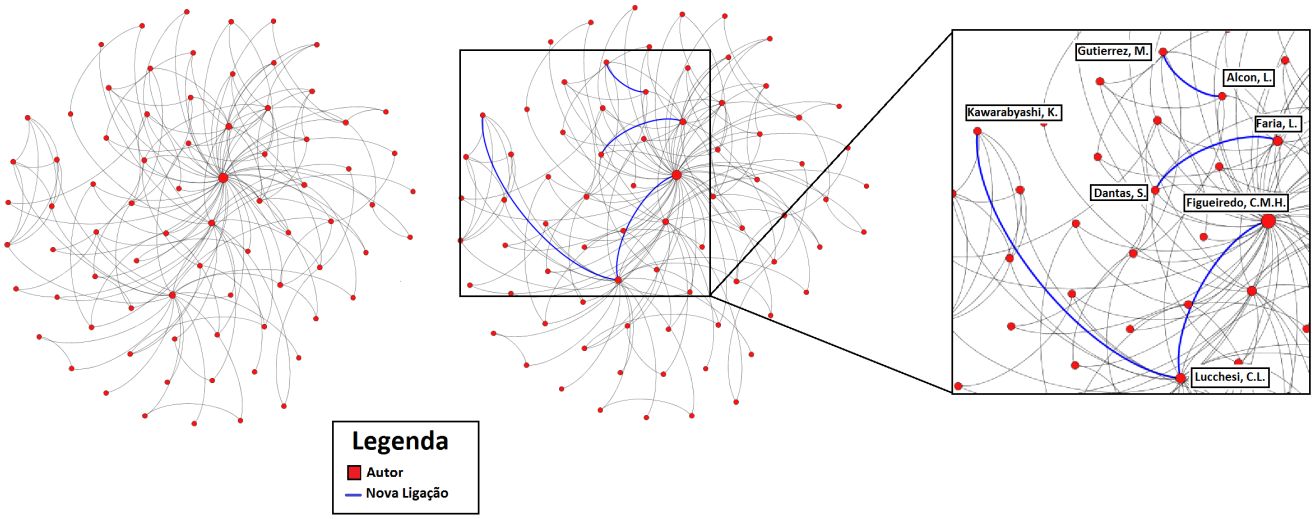

Figura 4.15: Resultado da predição de ligações com filtro de nós do tipo autor

A predição de ligações em dados filtrados pode ser considerada como uma contribuição deste trabalho, uma vez que no processo de filtragem do banco de dados orientado a grafos Neo4j não funciona corretamente. Nossa contribuição no *plugin* de exportação permite exportar as arestas tipadas.

5 CONCLUSÃO

Nas últimos tempo, os avanços tecnológicos proporcionaram um crescimento na quantidade de dados, sobre os mais diversos assuntos, os quais estão facilmente disponíveis para consulta de qualquer pessoa com acesso à Internet e elaboração de métodos que prevejam com precisão a existência de ligações entre estes dados pode ser encarada como um desafio. Nesse contexto, a predição de ligações tem sido um assunto muito abordado pela comunidade científica e pode-se entendê-la como a previsão da existência de uma aresta entre dois nós de um grafo que representam os dados, enquanto as arestas representam ligações entre eles.

Seguindo esta linha, o objetivo deste trabalho foi o desenvolvimento de um *plugin* para predição de ligações que integrasse um Sistema de Gerenciamento de Banco de Dados e um Sistema Visualizador e Analisador de Grafos e possibilitasse o armazenamento e análise dos dados.

Como contribuições deste trabalho pode-se citar a integração das ferramentas de gerência de banco de dados e de visualização de dados, a implementação do *plugin* de predição de ligações e a adaptação do *plugin* de importação/exportação de dados, tendo em vista que a versão existente do *plugin* de importação/exportação disponível *online* não funciona adequadamente.

Por fim, observou-se a possibilidade de utilizar a predição como recurso de previsão de futuras ligações ou para recuperação de ligações faltantes, além da possibilidade da extração de informações de preferências que podem ser interpretadas a partir da análise dos resultados da predição.

Para futuros trabalhos sugere-se a implentação de novas formas de se realizar a predição de ligações, tendo como base o trabalho desenvolvido até o presente momento. Uma possível vertente seria inserir novos algoritmos de predição de ligação e comparar os resultados. As novas formas de predição podem ser inseridas como *plugin*s aproveitando a estrutura já existente.

6 REFERÊNCIAS

[1] FIRMINIO, Fabrício de Faria; FRANÇA, Tiago Cruz. (2014) Big Social Data: Princípios sobre Coleta, Tratamento e Análise de Dados Sociais. SCB 1º ed.

[2] LIBEN-NOWELL, D.; KLEINBERG, J. (2003). The link prediction problem for social networks, New Orleans, LA, USA.

[3]BRINGMANN, B. et al. (2010). Learning and Predicting the Evolution of Social Networks. IEEE Intelligent Systems, 25(4), 26-35.

[4] MATHEUS, Renato Fabiano; SILVA, Antonio Braz de Oliveira. (2006). Análise de redes sociais como método para a Ciência da Informação. Revista Ciência da Informação, v.7 nº2 .

[5] NEWMAN, M. E. J. (2000). Who is the best connected scientist?: a study of scientific coauthorship networks. Santa Fé: The Santa Fé Institute. Paper 00-12-064.

[6]HUANG, Z.; LIN, D. K. J. (2009). The Time-Series Link Prediction Problem with Applications in Communication Surveillance. INFORMS J. on Computing 21, 2, 286-303.

[7] JUNIOR, Geraldo de Souza.(2014). Sistemas de recomendação em redes sociais baseado em análise de grafos.

[8] CAVALCANTI, Maria Cláudia Reis; JUSTEL, Claudia M.; et al.(2014) Investigando Abordagens para o Problema de Predição de Ligações e sua Aplicação em Diferentes Contextos. Projeto MCTI/CNPq.

[9]http://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando/ acesso em 12 de setembro de 2014.

[10] LITT, Steve.(2007).NoSQL: The Unix Database.

[11] DAN MC CREARY, Ann Kelly. (2014). Making sense of NoSQL: A guide for managers and the rest of us. Manning Publications Co. 2014.

[12] ROBBINSON, Ian;WEBBER, Jim ; EIFREM, Emil, O'Reilly.(2013) Graph Databases.

[13]http://www.dbmaker.com.tw/reference/manuals/tutorial/tutorial.html acessado em 02 de agosto de 2014.

[14] http://www.ranz.com/agraph/allegrograph/ acessado em 15 de outubro de 2014.

[15] http://www.orientechnologies.com/orientdb/ acessado em 05 de agosto de 2014.

[16] http://db-engines.com/en/ranking_definition acessado em 20 de outubro de 2014.

[17] https://gephi.github.io/, acesso em 07 de agosto de 2014.

[18] http://d3js.org/, acesso em 07 de agosto de 2014.

[19] LIBEN-NOWELL, D.; KLEINBERG, J. (2007). The link-prediction problem for social networks. Journal of the American Society Information Science and Technology.

[20] CARAGEA, C.; et al. (2009). Learning link-based classifiers from ontology-extended textual data.

[21] NEWMAN, M.E.J. (2001). Clustering and preferential attachment in growing networks. Phys. Rev. E.

[22] LORRAIN, F. ; WHITE, H. [1971]. Structural equivalence of individuals in social networks. Journal of Mathematical Sociology.

[23] http://neo4j.com/api_docs/2.0.3/ acessado em 20 de outubro de 2014.

[24] https://wiki.gephi.org/index.php/Specification_-_GSoC_Adding_support_for_Neo4j_in_Gephi acessado em 13 de outubro de 2014.

[25] https://wiki.gephi.org/index.php/Build_Gephi acessado em 01 de outubro de 2014

[26] https://wiki.gephi.org/index.php/Plugins_portal acessado em 01 de outubro de 2014

[27] https://marketplace.gephi.org/plugin/neo4j-graph-database-support/ acessado em 2 de fevereiro de 2015.

[28] BARBOSA,Diego A. B. L. ; AVELINO, Leonardo B.; SOUZA, Rarylson F. Utilização de parâmetros espectrais em redes sociais. 2011. Trabalho de Conclusão de Curso. (Graduação em Engenharia de Computação) - Instituto Militar de Engenharia.