Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

Algoritmo Multinível de Detecção de Comunidades em Redes Complexas

Steven Koiti Tsukamoto

Supervisores: Prof. Dr. Marcelo Finger
Dra. Ana Paula Appel

Colaborador: Prof. Dr. Estevam R. Hruschka Jr. (UFSCar)

São Paulo
Janeiro de 2015

Dedicatória

Aos meus pais,
Paulo e Marisa.
Ao meu irmão,
Hideki.
À minha namorada,
Yukie.
E aos meus amigos.

Agradecimentos

Gostaria de agradecer a todos que tornaram esse trabalho possível, principalmente à minha família pela ajuda e incentivo que me deram durante todos esses anos. E também à minha namorada, pelo grande apoio em todos os momentos.

Um grande agradecimento aos meus orientadores. Ao professor Marcelo Finger pelo apoio e orientação. À Ana Paula por tudo que me ajudou, acompanhando do começo ao fim o desenvolvimento deste projeto. E ao professor Estevam pelos conselhos e por me guiar durante este trabalho.

Os meus agradecimentos ao IME-USP pela oportunidade de crescer e aprender. Obrigado a todos os professores e colegas que diretamente ou indiretamente contribuíram no resultado deste trabalho de conclusão.

Aprender é a única coisa de que
a mente nunca se cansa, nunca
tem medo e nunca se arrepende.

LEONARDO DA VINCI

Resumo

O surgimento e crescimento de dados modelados em grafos, como as redes complexas, tornou possível a criação de novas áreas de pesquisa como a mineração de grafos. Nesta área os assuntos que se destacam são a detecção de comunidades, a extração de propriedades estatísticas e a predição de ligações.

A detecção de comunidade pode ser utilizada para o pré-processamento do grafo a fim de decompô-lo em grupos que podem ter suas propriedades mineradas mais facilmente. É um problema bem explorado mas ainda bastante complexo em que até agora não foram encontradas soluções que o resolvam de forma satisfatória.

Este trabalho apresenta a implementação de um algoritmo multinível para detecção de comunidades. O principal estudo de caso é o projeto "Leitura da Web", que visa a construção de uma ontologia para representar o conhecimento extraído da Web a partir de um algoritmo de aprendizado sem fim. A fim de avaliar a escalabilidade e eficiência do método proposto foram realizados experimentos no grafo obtido.

Abstract

The emergence and growth of data modeled in graphs as complex networks has made possible the creation of new research areas such as graph mining. In this area the highlighted issues are the community detection, extraction of statistical properties and link prediction.

The community detection can be used for the preprocessing of the graph in order to break it down into groups where their properties can be more easily mined. It is a well-explored problem but still quite complex in that so far no solutions that solve satisfactorily were found.

This work presents the implementation of a multilevel algorithm for community detection. The main case study is the "Read the Web" project, which aims to build ontologies to represent the knowledge extracted from the Web by a never-ending learning algorithm. In order to check the scalability and efficiency of the proposed method experiments were performed in the obtained graph.

Sumário

I Parte Técnica **9**

1 Introdução **9**

1.1 Motivação 9

1.2 Objetivos 10

1.3 Organização 10

2 Apresentação dos conceitos **12**

2.1 Grafos 12

2.2 Redes Complexas 12

2.2.1 Redes aleatórias 14

2.2.2 Modelo Watts-Strogatz 14

2.2.3 Modelo Barabási-Albert 15

2.3 Detecção de comunidades 17

2.3.1 Girvan-Newman 18

2.3.2 Algoritmo Clauset-Newman-Moore 19

2.3.3 Walktrap 19

2.3.4 Método de Louvain 19

2.3.5 Leading Eigenvector 20

2.3.6 Label Propagation 20

3 Projeto Leitura da Web **21**

3.1 Never-Ending Language Learning 21

3.2 Construção do grafo 22

3.2.1 Grafo sem url 23

3.2.2 Grafo transformado 24

3.2.3 Grafo transformado sem url 25

4 Algoritmo Multinível de Detecção de Comunidades **26**

4.1 Considerações iniciais 26

4.2 Algoritmo Multinível de Detecção de Comunidades 26

4.3 Implementação 27

4.3.1 Fase de Coarsening 28

4.3.2 Fase de Particionamento 29

4.3.3 Fase de Uncoarsening 30

5 Experimentos **31**

5.1 Conjunto de dados 31

5.2 Resultados 31

5.2.1 Fase de Coarsening 31

5.2.2 Fase de particionamento - grafo sem url 32

5.2.3 Fase de particionamento - grafo transformado e sem url 33

5.2.4 Grafos resultantes 34

6 Considerações finais **37**

6.1 Conclusão 37

6.2 Trabalhos futuros 37

Referências **39**

II Parte Subjetiva 42

1 Desafios Encontrados **42**

2 Disciplinas relevantes e aplicações dos conceitos **42**

Parte I

Parte Técnica

1 Introdução

1.1 Motivação

Podemos dividir a história da computação em diversas eras. Os sistemas tabulares surgiram no século 19 e perduraram até aproximadamente 1940. Nesta década começa a emergir a computação programável onde cientistas construíram os primeiros computadores eletrônicos programáveis, permitindo avanços em diversas áreas da ciência e tecnologia. Estamos entrando agora no que alguns pesquisadores dizem ser a "Era da Computação Cognitiva". As máquinas não irão pensar como os seres humanos, mas nos ajudarão na percepção dos problemas e na tomada de decisão. Com esse novo conceito surgem diversos desafios como a compreensão da linguagem natural. E assim, sistemas de aprendizado semântico como o *IBM Watson*[1] e o *NELL*[2] (*Never-Ending Language Learning*) tornam-se ferramentas importantes e que demonstram ser ótimas soluções na construção de sistemas mais inteligentes.

O *NELL* é um sistema de aprendizado contínuo que tem por objetivo percorrer a internet e extrair conhecimento de milhares de textos existentes na rede. Todo esse conhecimento adquirido pode ser modelado na forma de um grafo, podendo-se explorar melhor as conexões entre os fatos. Porém, a Web é um ambiente repleto de recursos e ao se extrair essa grande quantidade de informação chega-se a um ponto em que não conseguimos mais analisar individualmente as instâncias e necessitamos enxergar de uma outra maneira o problema, de modo mais rápido e fácil. Com os métodos de detecção de comunidades podemos analisar mais abrangentemente como essas instâncias se conectam e se comportam dentro do grafo.

Por se tratar de uma rede dinâmica que constantemente está aprendendo novos fatos e ser um sistema real suscetível à falhas, podem ocorrer erros na categorização dos objetos. Ao dividir o grafo em comunidades podemos analisar possíveis "falhas" na rede como um nó mal categorizado. Ou ainda, encontrar possíveis ligações entre instâncias ainda não rotuladas. Um dos maiores benefícios seria então poder compreender como o sistema evolui e de que maneira os subgrupos se aglomeram. Importantes informações começam a ser percebidas como quais categorias mais crescem ou as que mais devem ser exploradas.

[1] http://www.ibm.com/smarterplanet/us/en/ibmwatson/
[2] http://rtw.ml.cmu.edu/rtw/

1.2 Objetivos

O problema de detecção de comunidades é uma tarefa custosa de complexidade NP-completo, ou seja, não existem algoritmos que resolvam o problema em tempo polinomial. Para isso, implementaremos uma versão multínivel que visa uma maior eficiência no processamento das soluções, podendo combiná-lo com outros algoritmos em sua fase de particionamento.

O objetivo deste trabalho é encontrar métodos de detecção de comunidades que se adequem a sistemas escaláveis e dinâmicos. Como estudo de caso usaremos a base de conhecimento adquirido pelo sistema *NELL* e a partir da extração de seu grafo, fazer experimentos e analisar os resultados obtidos. Portanto, não só gostaríamos de identificar possíveis comunidades como também fazer um estudo e verificar quais técnicas geraram melhores soluções, levando-se em consideração tanto o tempo de processamento quanto a qualidade dos subgrupos particionados.

Podemos dividir os objetivos do trabalho em dois principais focos:

- Implementação de um Algoritmo Multinível de Detecção de Comunidades
- Análise comparativa das soluções obtidas

1.3 Organização

A estrutura do trabalho foi organizada da sequinte forma:

- *Seção 1* - **Introdução**
  Nesta primeira seção é apresentada ao leitor uma visão geral sobre o trabalho, mostrando os motivos e objetivos do tema pesquisado.
- *Seção 2* - **Apresentação dos conceitos**
  Contextualização dos principais assuntos abordados no trabalho.
- *Seção 3* - **Projeto Leitura da Web**
  Descreve as principais características do projeto de pesquisa e do sistema *NELL* (*Never-Ending Learning Language*).
- *Seção 4* - **Algoritmo Multinível de Detecção de Comunidades**
  Fornece uma base conceitual sobre o algoritmo multinível para particionamento de grafos, assim como detalhes da implementação do Algoritmo Multinível de Detecção de Comunidades.

- *Seção 5* - **Experimentos**
  Nesta seção são analisados e discutidos os resultados obtidos durante o desenvolvimento do trabalho.

- *Seção 6* - **Considerações finais**
  São explicitadas as considerações finais, assim como uma conclusão sobre os resultados.

2 Apresentação dos conceitos

2.1 Grafos

Grafos são definidos como um par ordenado $G = (V, E)$ que consiste de um conjunto de **vértices** V e uma coleção de **arestas** E. Os vértices correspondem aos objetos modelados e as arestas indicam uma relação entre os pares de vértices. Uma aresta pode ser definida como:

$$e = (u, v), \text{ onde } u, v \in V. \tag{1}$$

Grafos podem ou não ter suas arestas direcionadas. Neste trabalho, os grafos serão definidos como **não direcionados**, ou seja, se u está diretamente conectado com v, então v também está diretamente conectado com u.

O **grau** $d(v)$ é o número de arestas que são **incidentes** ao vértice v.

$$d(v) = |\{(i, j) \in E \mid i = v \text{ ou } j = v\}| \tag{2}$$

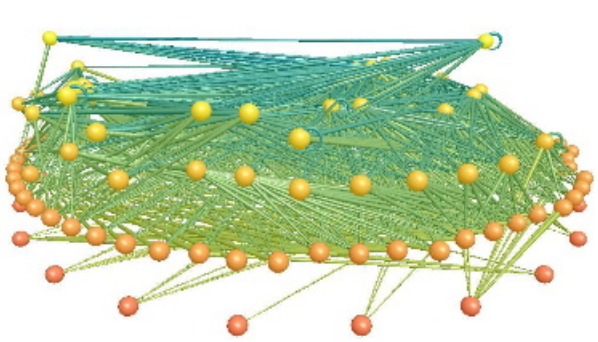
Dois vértices são considerados **adjacentes** (ou **vizinhos**) quando existe uma aresta que os conecta.

Seja um grafo $G = (V, E)$, um grafo $H = (V', E')$ é um **subgrafo** de G se $(V' \subseteq V)$ e $(E' \subseteq E)$; escrevemos $H \subseteq G$. O conjunto de subgrafos distintos que formam o grafo é chamado de partição. **Blocos** são os subgrafos que compõe a partição.

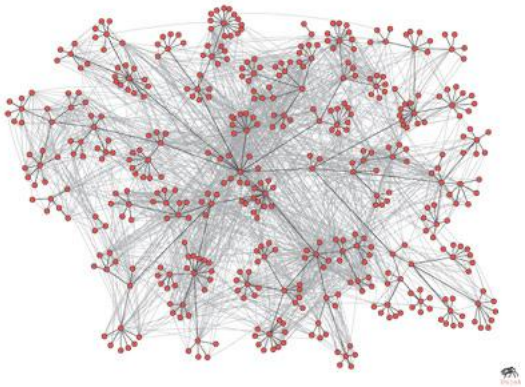
2.2 Redes Complexas

Nas últimas décadas testemunhamos o nascimento de um novo movimento de interesse e pesquisa com relação ao estudo de **redes complexas**, ou seja, redes cuja estrutura é irregular, complexa e que dinamicamente evolui conforme o tempo [11]. Estes sistemas apresentam características topológicas **não-triviais**, isto é, suas propriedades não ocorrem em estruturas simples como grafos aleatórios mas constantemente se apresentam em redes de sistemas reais. Apesar de possuirem propriedades irregulares, esses sistemas não são puramente randômicos, possuindo certos padrões de conexão entre seus elementos.

Estruturas complexas descrevem uma enorme quantidade de sistemas que recentemente têm instigado os cientistas a investigarem mecanismos que determinem a sua topologia. Podemos citar alguns exemplos como:



(a) Cadeia Alimentar do lago *Little Rock, Wisconsin* (Figura fornecida por Neo D. Martinez)



(b) Rede social mostrando a comunicação de email entre 436 funcionários do Laboratório de Pesquisa da Hewlett Packard (Imagem obtida em http://www-personal.umich.edu/ladamic/img/hplabsemailhierarchy.jpg)

Figura 1: Exemplos de redes

- Redes sociais
- Cadeia alimentar [15]
- Redes metabólicas [16]
- *World Wide Web* [17]
- Internet [18]
- Rede de co-autores e citações científicas [19]
- Redes de energia elétrica

Visualizando as figuras 1a e 1b, podemos observar a grande complexidade estrutural que as redes reais podem formar. Existe uma grande dificuldade em se analisar sucintamente tais tipos de rede como um todo pois certas partes são mais **densamente conectadas** do que outras, ou seja, em algumas redes notamos um núcleo central que reúne praticamente todas as conexões, e em outros momentos verificamos uma estrutura dividida em mútiplas regiões fortemente conectadas.

Portanto, a estrutura de uma rede é de fundamental importância para a compreesão de seu funcionamento. Por exemplo, a topologia de uma rede social mostra a forma como as informações se dispersam na rede, e a estrutura de uma rede de energia afeta a robustez e estabilidade da transmissão de eletricidade. Um dos primeiros passos para entender uma rede é analisar as características estruturais típicas que ela possui. Iremos conceituar alguns dos principais modelos de redes complexas já estudadas.

2.2.1 Redes aleatórias

Em 1959, os matemáticos húngaros **Paul Erdős** e **Alfred Rényi** desenvolveram um dos modelos mais simples de redes complexas [22]. A rede é construída a partir de um número n de vértices e uma probabilidade p que conecta os pares de vértices. O modelo **Erdős-Rényi** propõe percorrer todas as combinações dos pares de vértices e decide através da probabilidade p se deve ou não adicionar uma aresta.

Podemos construir a seguinte relação entre o número n de vértices e m de arestas da rede aleatória:

$$m = p\frac{n(n-1)}{2} \tag{3}$$

$\frac{n(n-1)}{2}$ indica o número máximo de ligações.

Apesar de Erdős e Rényi estarem focados essencialmente na matemática do problema, e não nas suas aplicações práticas em sistemas reais, o modelo consegue exemplificar diversas redes existentes no mundo, como por exemplo, de rodovias e estradas de ferro [23].

2.2.2 Modelo Watts-Strogatz

O psicólogo **Stanley Milgram** realizou em 1967 um experimento que consistia em tentar descobrir o número de conhecidos que separam duas pessoas escolhidas ao acaso. Para isso, Milgram conseguiu a ajuda de 160 voluntários que deveriam enviar uma carta com o intuito de chegar até uma pessoa alvo, e caso não conhecessem a pessoa, deveriam repassar a carta para algum conhecido. O processo de repasse de cartas deveria continuar até que se chegasse à pessoa alvo. No final, um total de 42 cartas chegaram ao alvo, podendo-se concluir que o grau de separação médio é de 6 pessoas. Esta propriedade ficou conhecida como **small world**, ou também **mundo pequeno**.

Diversas redes reais posicionam-se entre a regularidade e a aleatoriedade. Percebendo esse fenômeno os pesquisadores **Duncan Watts** e **Steven Strogatz** formularam um modelo mais sofisticado que pudesse representar este padrão e adicionar também a propriedade mundo pequeno.

O modelo **Watts-Strogatz**, também conhecido como **redes de mundo pequeno**, transita entre a estrutura de um grafo regular e um grafo aleatório. Podemos observar as características do modelo principalmente em redes sociais, onde as pessoas estão relacionadas com parentes, amigos e conhecidos, e estes também estão conectados com outras pessoas de

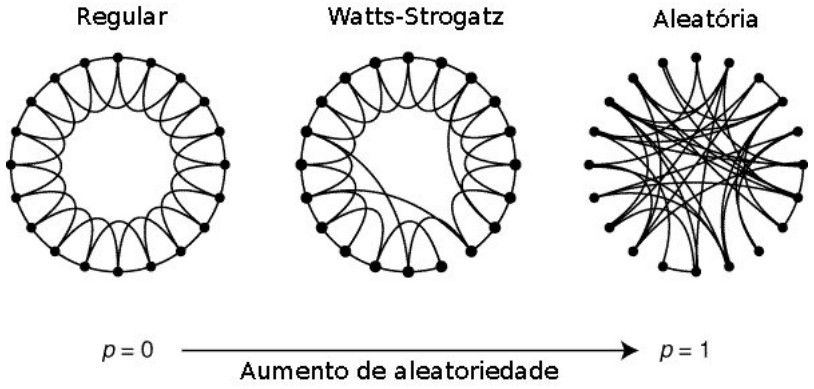


Figura 2: O modelo Watts-Strogatz interpola entre uma rede regular e aleatória, sem alterar o número de nós ou arestas. O exemplo acima ilustra um grafo com n=20 vértices, cada um deles ligado aos k=4 vizinhos mais próximos. [24]

diversas partes do mundo. O algoritmo para a construção de uma rede de mundo pequeno, ilustrado na figura 2, segue os seguintes passos:

1. *Estrutura regular*: Inicialmente teremos um grafo com n vértices e cada um deles ligados a k vizinhos. Para mantermos uma rede esparsa e também conectada, devemos considerar $n > k > \ln(n) > 1$.

2. *Aleatorizar*: Usando como parâmetro uma probabilidade p, cada uma das arestas é reconectada (excluem-se os casos de arestas ligando um vértice a ele mesmo, e também arestas duplicadas). Caso $p = 1$, então todas as arestas serão reconectadas, e teremos um comportamento idêntico às redes aleatórias. Já, se $p = 0$ teremos um grafo regular com caminhos mais longos.

2.2.3 Modelo Barabási-Albert

Em 1998, os pesquisadores **Albert-László Barabási** e **Réka Albert** perceberam que redes como a *World Wide Web* possuem uma distribuição de grau que segue uma *power-law* (ou lei de potência, vide figura 3). Ao contrário das redes aleatórias e do modelo Watts-Strogatz, os nós não possuem a mesma chance de receberem conexões, ou seja, podemos perceber a existência de certos vértices com alta conectividade dentro do grafo (os *hubs*).

O modelo **Barabási-Albert** foi baseado em duas propriedades na tentativa de explicar o surgimento dos *hubs* em **redes de livre escala**. A primeira é chamada de **crescimento** e indica que a rede cresce com o passar do tempo (a cada instante novos nós surgem e se conectam ao grafo). A segunda propriedade, conhecida como **ligação preferencial** (ou também *rich-get-richer*), diz que os nós "populares" (aqueles com maior grau) são mais atrativos e possuem uma probabilidade maior de receberem ligações. Páginas *web* mais

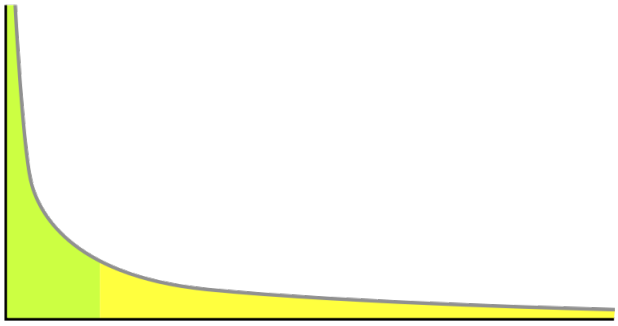


Figura 3: Exemplo de *power-law*, onde uma quantidade varia como potência da outra quantidade. (Imagem obtida em: http://en.wikipedia.org/wiki/File:Long_tail.svg)

apontadas são as mais prováveis a serem apontadas pelas novas páginas e artigos científicos mais citados são os mais propensos a serem citados novamente. Esta propriedade argumenta que quanto mais conhecida e famosa é uma instância, mais conexões ela receberá.

O modelo é construído a partir de n_0 vértices inicialmente conectados (m_0 arestas). O algoritmo segue 2 principais premissas:

1. *Crescimento*: A cada instante de tempo, adiciona-se um novo vértice ao grafo que se conecta a outros m ($m < n_0$) vértices já existentes.

2. *Ligação preferencial*: A probabilidade de um novo vértice escolher um determinado nó para conexão será proporcional ao número de arestas que este possui. Dessa maneira, os novos nós tendem a se ligar com os nós com mais ligações. A probabilidade P de um novo vértice se conectar a um nó i depende de seu grau k_i, podemos conferir a equação desta probabilidade a seguir:

$$P(i) = \frac{k_i}{\sum_j k_j} \tag{4}$$

Depois de t instantes de tempo, teremos como resultado uma rede com $N = n_0 + t$ vértices e $M = m_0 + (m \times t)$ arestas. Podemos observar pela figura 4 um exemplo de rede inicialmente com $n_0 = 3$ vértices e $m_0 = 3$ arestas, e cada vértice novo ligando-se a $m = 2$ vértices. Claramente podemos notar a prefência de conexão com os nós mais antigos e de maior grau, permitindo o surgimento de *hubs*.

Dentre as variações do modelo Barabási-Albert podemos destacar as redes cujas as novas ligações podem emergir enquanto a rede cresce, ou seja, ocorrer conexões internas entre duas instâncias antigas no grafo. Outra característica encontrada em sistemas reais que não ocorre nas redes de livre escala é de que alguns nós e arestas poderem desaparecer conforme o tempo. Por exemplo, em redes sociais podemos observar que um relacionamento não perdure para

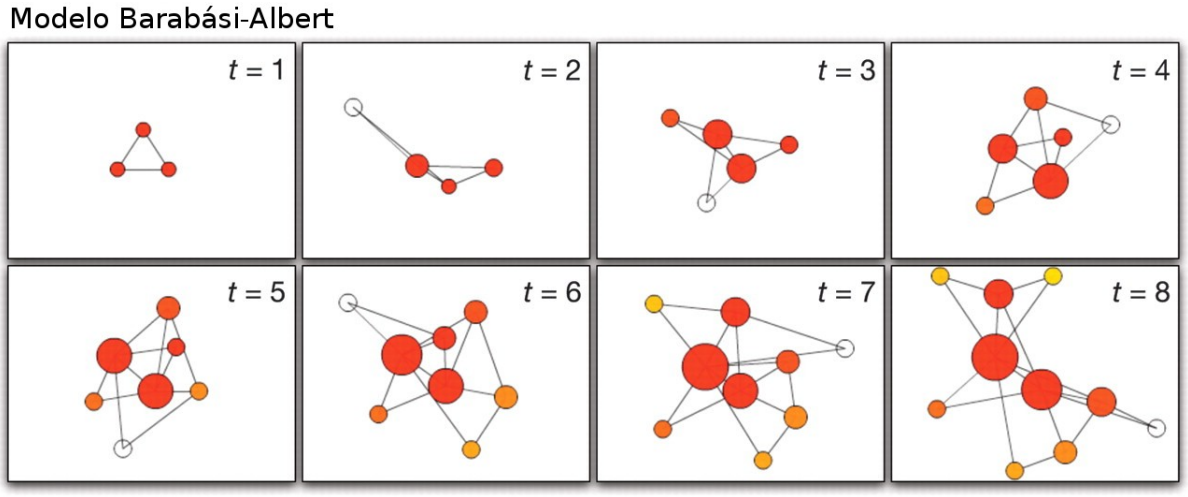


Figura 4: Construção do modelo Barabási-Albert. Os novos vértices preferem se ligar aos nós mais conectados. Devido às propriedades de *crescimento* e *ligação preferencial* podemos perceber o surgimento de *hubs*. (Imagem obtida em: http://www.sciencemag.org/content/325/5939/412/F1.large.jpg)

sempre, podendo ser rompido após um certo tempo. Conexões internas, reconecções, remoção de nós e arestas, envelhecimento ("*aging*"), efeitos não-lineares, e entre outras propriedades são características que alteram a evolução e crescimento da rede, mudando a quantidade e tamanho dos *hubs*. Diversos modelos são criados na tentativa de construir as diferentes topologias encontradas em redes reais.

2.3 Detecção de comunidades

Comunidades, também conhecidas como *clusters* ou módulos, são grupos de vértices que compartilham propriedades em comum e/ou desempenham funções semelhantes dentro do grafo. Já a **detecção de comunidades** visa identificar esses módulos apenas usando as infomações da topologia do grafo. Na figura 5, podemos notar um exemplo de uma rede em que são destacadas as suas comunidades.

Muitas das redes sociais apresentam uma **estrutura de comunidade** [20], ou seja, possuem uma alta densidade de conexões entre os vértices de um mesmo grupo e uma baixa densidade de ligações entre vértices de grupos distintos. Podemos perceber que as pessoas dentro de uma rede social tendem a se juntar em grupos de mesmo interesse, idade, gênero, ocupação e assim por diante.

Outra aplicação bastante concreta ocorre na rede *Web*, onde as comunidades são suscetíveis a agrupar páginas referentes a um mesmo tópico. Identificar *cluster* de consumidores com interesses similares em uma rede de compras de uma loja de varejo (como, por exemplo,

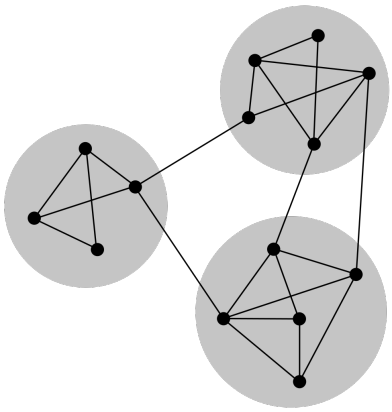


Figura 5: Esboço de uma pequena rede exibindo uma **estrutura de comunidade**, com três grupos de nós com densa conexão interna e uma esparsa conexão entre os grupos. (Imagem obtida em: http://en.wikipedia.org/wiki/File:Network_Community_Structure.svg)

Amazon [3]) possibilitam desenvolver um eficiente sistema de recomendação [21], que melhor guia os clientes pelas diversas ofertas de produtos e que aumentam as oportunidades de négocio.

Detectar comunidades em uma rede arbitrária é uma tarefa computacionalmente difícil. Tipicamente o número de comunidades é desconhecido, e os tamanhos e densidades dos módulos são diferentes. Diante dessas dificuldades, diversos métodos foram desenvolvidos e cada um deles com variados níveis de sucesso.

Descreveremos a seguir os principais conceitos de alguns métodos de detecção de comunidades:

2.3.1 Girvan-Newman

Usando um processo de **decomposição hierárquica**, o método remove aresta por aresta em ordem decrescente em relação a **medida de intermediação** até que o grafo seja dividido em comunidades. A centralidade de intermediação quantifica o número de vezes que uma aresta age como ponte entre caminhos mínimos de dois vértices distintos. A intermediação possui resultados mais eficazes do que apenas a conectividade de um vértice pois a medida consegue abranger de forma global o grafo enquanto a segunda tem somente um efeito local.

O algoritmo Girvan-Newman é bastante popular e gera resultados com boa qualidade. Porém, a grande desvantagem é sua complexidade computacional de $O(nm^2)$ em uma rede de n vértices e m arestas, mostrando-se inviável sua aplicação em grafos com mais do que alguns milhares de nós. O método é bastante lento pois o cálculo da medida de intermediação

[3] http://www.amazon.com/

é bastante custosa e a cada aresta removida, deve-se calcular novamente a medida para todo o grafo.

2.3.2 Algoritmo Clauset-Newman-Moore

Empregando também uma abordagem hierárquica, o algoritmo *Clauset-Newman-Moore* procura otimizar de maneira gulosa a qualidade da função de **modularidade**. A modularidade é uma medida usada para mensurar a força de divisão de uma rede em módulos. Grafos com modularidade alta têm uma densa conexão entre os nós de um mesmo grupo, mas com ligações esparsas entre nós de grupos diferentes.

Inicialmente, todo vértice pertence a uma comunidade isolada, e então as comunidades são unidas iterativamente de modo que cada união seja localmente ótima, isto é, de forma a maximizar o valor de modularidade. O algoritmo termina quando não é mais possível aumentar a modularidade, resultando em um agrupamento ótimo. O método é bastante rápido e frequentemente utilizado como uma primeira aproximação pois não há parâmetros a serem ajustados. Porém, o método sofre de um **limite de resolução**, ou seja, comunidades abaixo de um tamanho limiar (dependendo do número de vértices e arestas do grafo) serão sempre unidos com as comunidades vizinhas. Portanto, o algoritmo é incapaz de detectar pequenas comunidades.

2.3.3 Walktrap

Consiste em um algoritmo baseado em **passeios aleatórios**. A ideia geral é de que realizando passeios aleatórios em um grafo, então esses passeios são mais prováveis de permanecerem dentro de uma mesma comunidade pois existem apenas algumas arestas que o levam para fora do *cluster*.

O algoritmo executa pequenos passeios aleátorios (em torno de 3-4-5 passos, dependendo dos parâmetros) e usa o resultado desses passeios para unir comunidades separadas. Este método é um pouco mais lento que a abordagem gulosa anterior, mas com resultados bastante precisos.

2.3.4 Método de Louvain

O método de *Louvain* é uma das abordagens mais populares em maximização de modularidade devido à sua simplicidade, eficiência e fácil implementação. Assim como o método de *Clauset-Newman-Moore*, o algoritmo de *Louvain* procura otimizar localmente as comunidades até que a modularidade global não consiga mais ser melhorada.

Essa otimização é realizada em duas etapas. A primeira consiste em procurar pequenas comunidades através da otimização de modularidade local. Na segunda etapa, são unidos os nós pertencentes à mesma comunidade e é construído uma nova rede em que os nós são as pequenas comunidades. Essas etapas são repetidas até que a modularidade máxima seja atingida e uma hierarquia de comunidade seja produzida. Assim como o outro método, este algoritmo também sofre de limite de resolução e falha em detectar pequenas comunidades.

2.3.5 Leading Eigenvector

Trata-se também de um método que procura otimizar a função de modularidade. Em cada etapa, o grafo é dividido em duas partes de forma que a separação resulte em um significante aumento na modularidade. Esse corte é determinado por uma avaliação do **autovetor principal** (*leading eigenvector*) da chamada matriz de modularidade. A condição de parada previne que os grupos altamente conectados sejam divididos.

Devido ao cálculo do autovetor, existem alguns casos em que o algoritmo não funciona em grafos degenerados onde o resolvedor apresenta certa instabilidade. Em grafos não-degenerados, é mais provável que sejam gerados resultados com um valor de modularidade maior do que no método *Clauset-Newman-Moore*, porém com tempo de execução um pouco mais lento.

2.3.6 Label Propagation

O algoritmo utiliza uma simples abordagem em que todos os vértices são marcados com um rótulo. O método então procede iterativamente e re-atribui rótulos para os nós de modo que cada nó possua o rótulo de maior frequência entre seus vizinhos, sendo esse procedimento executado de maneira síncrona. O algoritmo termina quando o rótulo de cada nó seja uma das etiquetas mais frequentes de sua vizinhança.

É um método rápido mas que gera resultados diferentes dependendo da sua configuração inicial (que é gerado aleatoriamente). Portanto, deve-se executá-lo inúmeras vezes para então construir uma rotulação inicial adequada.

3 Projeto Leitura da Web

3.1 Never-Ending Language Learning

O **NELL** (*Never-Ending Language Learning*) é um sistema de **aprendizado de máquina semântico** desenvolvido por pesquisadores da Universidade Carnegie Mellon (CMU) e da Universidade Federal de São Carlos (UFSCar), sendo financiado por organizações como Google, Yahoo!, DARPA (*Defense Advanced Research Projects Agency*) e NSF (*National Science Foundation*). O sistema faz parte do projeto "**Leitura da Web**" que tem como objetivo criar um sistema computacional capaz de ler a rede *web* e aprender com as informações adquiridas ao longo do tempo.

O *NELL* possui 3 principais objetivos:

1. Ser um estudo de caso em **aprendizagem contínua**.
2. Avançar o estado da arte em **processamento de linguagem natural**.
3. Desenvolver a maior base de **conhecimento estruturado**.

Desde o começo de 2010, o sistema *NELL* vem processando durante 24 horas por dia, 7 dias por semana, uma enorme quantidade de dados. Todos os dias ele executa dois tipos de tarefas:

1. *Leitura*: Extrair dados encontrados em textos de páginas *web* e popular uma base de conhecimento com fatos estruturados.
2. *Aprendizado*: a cada dia conseguir melhorar sua capacidade de leitura, extraindo fatos cada vez mais precisos e exatos.

Um dos grandes desafios encontrados é de como compreender a linguagem natural. Diversos outros sistemas como o *IBM Watson* são desenvolvidos com o intuito de serem capazes de responder perguntas feitas pelos humanos. É necessário saber reconhecer a semântica das palavra e também diferenciar uma ambiguidade ou conflito entre duas fontes de informações.

Sabendo que existe uma vasta redundância de informação na *web* (ou seja, um mesmo fato é repetido diversas vezes em diferentes páginas e textos), o sistema utiliza-se desse recurso para melhorar seu mecanismo de aprendizado e aumentar a certeza sobre os conhecimentos adquiridos. A confiança sobre um fato aumenta a medida que temos uma maior frequência

das mesmas palavras. Além disso, o *NELL* conta com um agente de conversação no *Twitter* [4] em que usuários podem contribuir e dizer se uma crença está correta ou errada, melhorando a qualidade da base de conhecimento do sistema.

Entre algumas habilidades do *NELL*, podemos citar: (1) capacidade de classificar palavras por categorias; (2) classificar pares de palavras por relações; (3) descobrir padrões e regras para predizer novas instâncias de relação; (4) descobrir novas relações para estender o conjunto de ontologias iniciais; (5) aprender a atribuir âmbito temporal para crenças.

3.2 Construção do grafo

Com o intuito de transformar informações não-estruturadas em informações estruturadas, o *NELL* possui um formato específico para armazenar o conhecimento adquirido. Atualmente, podemos acessar a base de conhecimento a partir da página do projeto "Leitura da Web" [5]. O arquivo disponibilizado possui todas as instâncias que o *NELL* considera ser verdade. Cada crença é uma tupla (entidade, relação, valor), como por exemplo, (George Harrison, tocaInstrumento, guitarra). Para diferenciar instâncias de mesmo nome mas de categorias diferentes, os pesquisadores do projeto atribuíram para cada instância um *conceito*. A palavra "*Apple*" tanto pode se referir a uma fruta quanto ao nome de uma empresa norte-americana de tecnologia, portanto teremos dois conceitos presentes na base de dados: conceito:fruta:apple e conceito:empresa:apple. O nome literal da palavra juntamente com sua respectiva categoria possibilita a diferenciação das ambiguidades entre instâncias.

O grafo do conhecimento do *NELL* é construído da ligação entre as entidades e os valores. No exemplo anterior em que havia uma relação entre as instâncias "George Harrison" e "guitarra", teríamos uma aresta ligando o vértice conceito:músico:george-harrison com o vértice conceito:instrumentomusical:guitarra. Por ser um sistema **dinâmico** e que está constantemente lendo e aprendendo novos fatos, diversas versões são disponibilizadas com o conhecimento mais recente. Na versão **NELL.08m.860** temos um grafo com as seguintes características:

- Número de vértices: 679.292
- Número de arestas: 441.329
- Número de componentes conexas: 310.983
- Tamanho da maior componente: 51.233

[4] https://twitter.com/cmunell
[5] http://rtw.ml.cmu.edu/rtw/resources

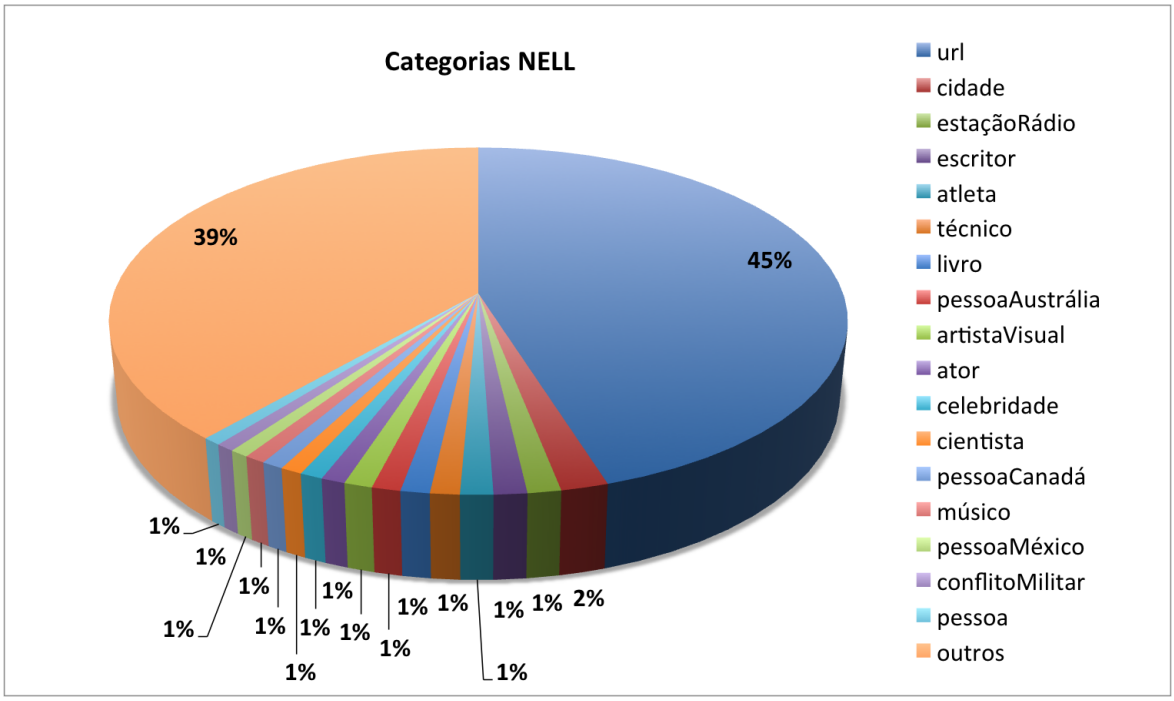


Figura 6: Categorias da base de conhecimento do *NELL*.

Podemos notar um grafo altamente **esparso**, ou seja, possuindo um pequeno número de arestas em relação à quantidade de nós. Além disso, a grande quantidade de componentes conexas revelam um elevado nível de **fragmentação** do grafo. Na figura 6, podemos observar a distribuição de categorias e notar a predominância de instâncias do tipo *url* (45% do total). Diante desses aspectos, foram propostas 3 novas versões do grafo:

3.2.1 Grafo sem url

Na primeira modificação do grafo retiramos os vértices relacionados com a categoria *url*. Pelo gráfico da figura 7, notamos agora uma distribuição mais equalitária entre as categorias.

- Número de vértices: 69.837
- Número de arestas: 134.700
- Número de componentes conexas: 8.156
- Tamanho da maior componente: 48.937

Podemos ver uma redução enorme do grafo que ficou com um tamanho de apenas 10% do original. Examinando melhor as instâncias deixadas de lado, tínhamos o seguinte formato se repetindo para as ligações: instância → url_da_instância(wikipedia)

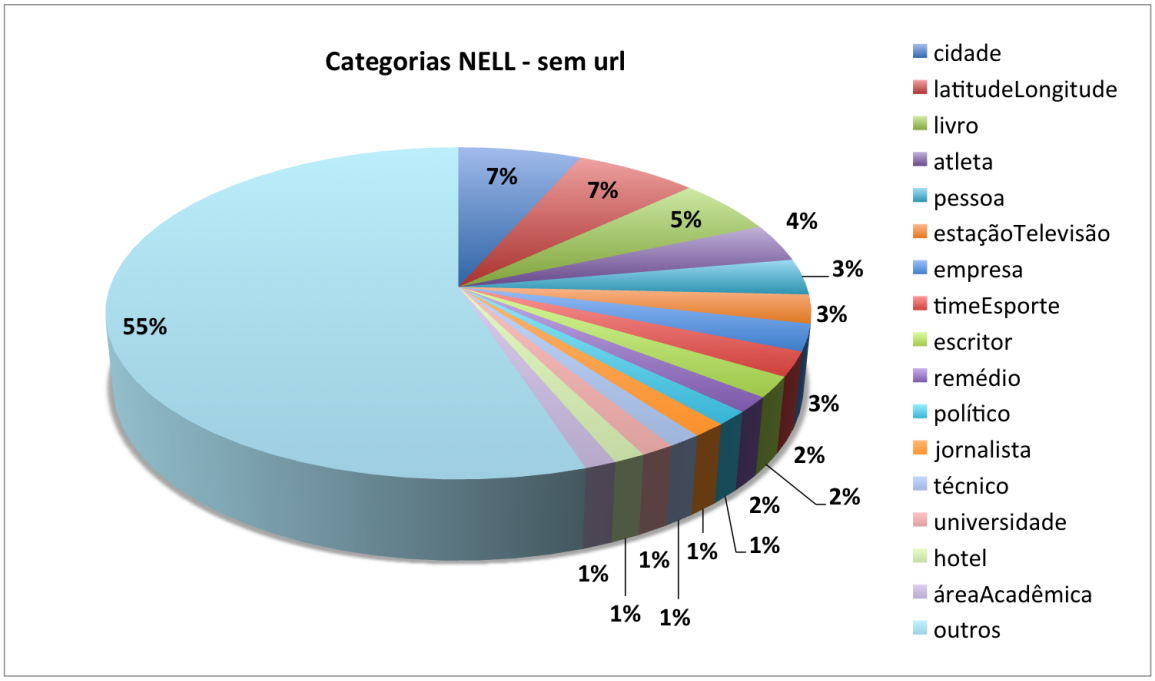


Figura 7: Categorias da base de conhecimento do *NELL* sem urls.

Exemplos:
conceito:país:camarões → conceito:url:http://en.wikipedia.org/wiki/Cameroun
conceito:produto:macbook → conceito:url:http://en.wikipedia.org/wiki/Macbook

Apesar da remoção de um elevado número de vértices e arestas, a transformação foi uma solução bastante positiva pois eliminou-se uma enorme quantidade de elementos desnecessários e que atrapalhavam a obtenção dos resultados. Podemos dizer que este grupo isolado de *urls* pode ser considerado uma comunidade e tratado como um subgrupo dentro do grafo do *NELL*.

3.2.2 Grafo transformado

Na segunda abordagem, as relações entre instâncias foram transformadas em vértices do grafo. Ou seja, visando um grafo mais conexo, uma das soluções encontradas foi tornar as arestas de ligações do grafo original também em vértices. Esta transformação foi possível pois todas as arestas possuem valores que guardam a relação a que se referem. Podemos notar na figura 8 que inicialmente tínhamos dois pares de instâncias ligados separadamente por uma aresta de relação (indicado pela flecha mais clara em cinza). Depois da transformação, todas as instâncias foram conectadas pela relação em comum que agora se tornou um vértice do grafo transformado. Dessa forma, vértices que antes compartilhavam de uma aresta com valor de relação em comum, agora se tornaram conectados.

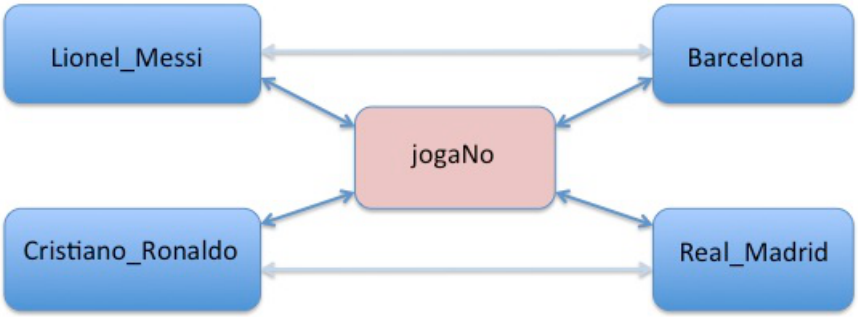


Figura 8: Exemplo da transformação do grafo

Depois da transformação, obtemos o grafo com as seguintes especificações:

- Número de vértices: 679.588
- Número de arestas: 1.323.987
- Número de componentes conexas: 2
- Tamanho da maior componente: 679.576

Nota-se que o número de componentes conexas diminuiu para apenas duas componentes, sendo que a maior componente representa praticamente todo o grafo. Ou seja, temos uma grafo quase 100% conectado, garantindo-se melhores condições para a detecção de comunidades.

3.2.3 Grafo transformado sem url

A última abordagem é a fusão das duas estrégias descritas acima: um grafo transformado e com a remoção dos elementos relacionados à categoria *url*. Esta tornou-se a versão mais versátil deste trabalho, pois além de garantir um grafo não viesado para uma determinada categoria (*url*), também apresentou ser uma solução bastante apropriada para a detecção de comunidades (grafo bastante conectado e não fragmentado).

- Número de vértices: 70.132
- Número de arestas: 404.100
- Número de componentes conexas: 6
- Tamanho da maior componente: 69.765

4 Algoritmo Multinível de Detecção de Comunidades

4.1 Considerações iniciais

O **algoritmo multinível** é uma técnica existente na literatura para o **particionamento** de grafos com um grande número de vértices e arestas [26].

Considere inicialmente um grafo $G_0(V_0, E_0)$. A estratégia multinível executa uma série de reduções no grafo G_0 até que se atinja um tamanho ideal para a **fase de particionamento**. O particionamento divide o grafo em k partes distintas, e então cada um desses blocos é expandido até que se retorne ao grafo G_0 inicial.

O algoritmo multínivel pode ser dividido em 3 fases:

Fase de Coarsening: o grafo G_0 é transformado sucessivamente em grafos menores $G_1, G_2, ..., G_x$ através da contração de suas arestas.

Fase de Particionamento: o grafo $G_x(V_x, E_x)$ é particionado em k partes, dividindo o conjunto de vértices V_x em k subgrupos distintos.

Fase de Uncoarsening: os blocos de G_x são projetados para cada um dos grafos intermediários gerados na fase de *coarsening* até que se retorne ao grafo original G_0.

São ilustadas na figura 9 as fases do método multinível. Neste exemplo, o grafo inicial G_0 é reduzido sucessivamente até atingir a configuração do grafo G_4. Na fase de particionamento o grafo reduzido é dividido em 6 partes. Na etapa seguinte os blocos são expandidos nos grafos intermediários G_3, G_2 e G_1 até que se retorne ao grafo G_0 original.

4.2 Algoritmo Multinível de Detecção de Comunidades

Uma das implementações da estratégia multinível existente na literatura é o programa **Metis**[6]. Desenvolvido por George **Karypis** e Vipin **Kumar**, o *Metis* tem como característica construir particionamentos balanceados, ou seja, os blocos possuem tamanho aproximado de $|V|/k$, sendo $|V|$ o número inicial de vértices e k a quantidade de blocos desejado.

Apesar de ser uma boa solução multinível já existente, o *Metis* é bastante restrito no particionamento. Ao aplicá-lo no problema de detecção de comunidades, obtemos como resultado sempre blocos de tamanho bem próximo (o que muitas vezes não condiz com a

[6] http://glaros.dtc.umn.edu/gkhome/views/metis

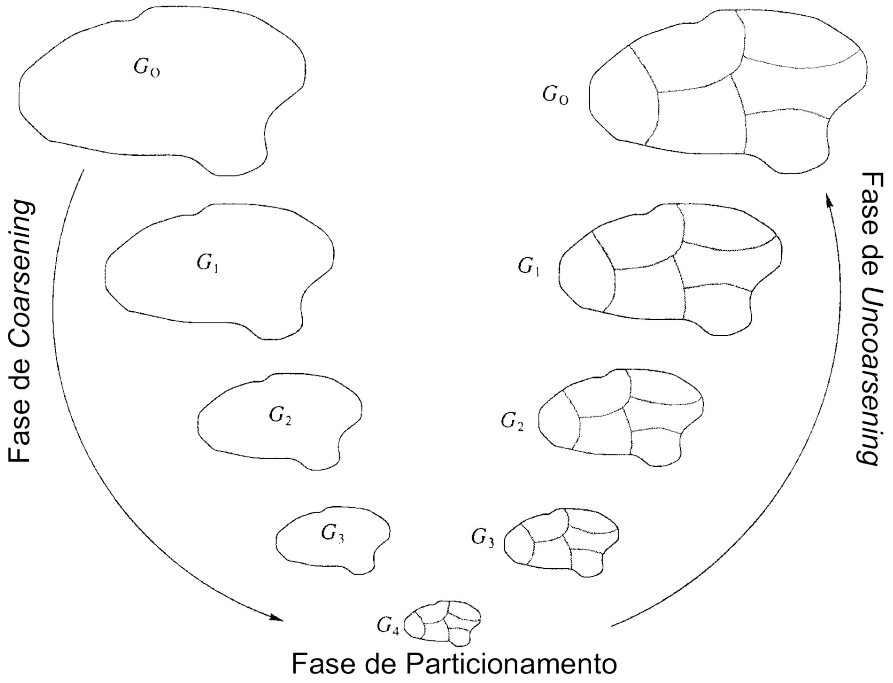


Figura 9: Esboço das fases do algoritmo multinível. [25]

realidade pois as comunidades são de tamanho variado). Além disso, o número k de blocos é um dos parâmetros do algoritmo, devendo-se testar o algoritmo para diferentes valores de k ou já ter um conhecimento prévio de um número ideal para k .

Tendo em vista essas limitações, foi implementada uma versão própria da estratégia multinível. A grande diferença dessa nova versão ocorre na fase de particionamento, onde utilizaremos algoritmos de detecção de comunidades já implementadas em bibliotecas para grafos. As principais vantagens dessa nova abordagem são:

1. Há uma grande variedade de opções para o particionamento, podendo-se aplicar métodos já bem estudados na literatura.
2. Aplicação de algoritmos custosos na fase de particionamento, pois teremos uma redução no tamanho do grafo na fase de *coarsening*.
3. Os blocos resultantes são de tamanho variado.
4. Não há necessidade de saber o número k de blocos.

4.3 Implementação

O pseudocódigo da implementação da estratégia multinível pode ser vista no algoritmo 1. A primeira parte do algoritmo consiste na redução do grafo G_0 até atingir um tamanho

ideal. Este critério de encerramento da fase de *coarsening* ocorre quando o grafo reduzido G_i atinge uma certa porcentagem do original. Em seguida temos o particionamento do grafo reduzido. E por último, temos a expansão do grafo reduzido até retornar ao grafo G_0 inicial. A saída é o particionamento P_0 obtido das expansões dos grafos intermediários.

Algoritmo 1 Algoritmo Multinível de Detecção de Comunidades

Entrada: G_0 - grafo inicial
Saída: P_0 - particionamento de G_0
$i \leftarrow 0$
repita
  $G_{i+1} \leftarrow$ Coarsening(G_i)
  $i \leftarrow i + 1$
até *critério de encerramento coarsening*;
$P_i \leftarrow$ Paticionamento (G_i)
repita
  $(G_{i-1}, P_{i-1}) \leftarrow$ Uncoarsening(G_i, P_i)
  $i \leftarrow i - 1$
até $G_i = G_0$;
retorna P_0

A seguir temos uma explicação mais detalhada da implementação de cada umas das fases do algoritmo multinível.

4.3.1 Fase de Coarsening

São descritas a seguir as 4 heurísticas implementadas na fase de *coarsening*: *Random Matching* (RM), *Heavy-Edge Matching* (HEM), *Modified Heavy-Edge Matching* (MHEM) e *Light-Edge Matching* (LEM).

RM: A heurística *Random Matching* começa rotulando todos os vértices como não-marcados. O algoritmo visita aleatoriamente todos os nós até que todos tenham sido marcados ou que o critério de redução tenha sido satisfeito. Ao visitar um vértice v não-marcado, o algoritmo verifica se existe algum vizinho u não-marcado para fazer o *matching*. Caso encontre, o par (v, u) é definido como marcado e entra na lista de *matching*. Caso contrário, apenas o vértice v é rotulado como marcado pois não foram encontrados vizinhos livres.

HEM: Assim como o RM, o algoritmo *Heavy-Edge Matching* inicializa rotulando todos os vértices como não-marcados e visita aleatoriamente cada um deles até que o critério

de redução seja atingido. Porém, a grande diferença está no momento da escolha do par de *matching*. O HEM procura pela aresta de maior valor (peso) dentre os vizinhos que ainda não foram marcados. O objetivo dessa estratégia é buscar um conjunto de vértices em que a lista de *matching* tenha o maior peso possível.

MHEM: A heurística MHEM é uma versão modificada da HEM. A estratégia é extremamente parecida, porém na fase de procura na vizinhança para o *matching* é escolhido o vizinho mais similar e que possua mais vértices adjacentes em comum. Ou seja, considere v o vértice escolhido e H o conjunto de vértices vizinhos a v que ainda não foram marcados. Para cada vértice $u \in V$, é calculado a soma de pesos das arestas entre u e os vértices adjacentes a u e v. O par (v, u) que obtiver a maior soma é escolhido para entrar na lista de *matching*.

LEM: A heurística *Light-Edge Matching* utiliza um critério oposto ao HEM. Ao invés de procurar pela aresta de maior peso entre os vizinhos, o LEM escolhe a aresta de menor peso.

O algoritmo 2 mostra o pseudocódigo utilizado na implementação da fase de *coarsening*. Primeiramente, a função *Matching* aplica no grafo uma das 4 heurísticas (RM, HEM, MHEM, LEM) para se obter o conjunto de *matchings*. Então, para cada aresta da lista de matchings é feito uma fusão de seus nós, transformando-os em um único **super-vértice**. Em seguida é calculado o novo peso das arestas entre o super-vértice e os vizinhos dos vértices geradores. No fim, retorna-se o grafo reduzido.

Algoritmo 2 Fase de *coarsening*

Entrada: G_i - grafo a ser reduzido
Saída: G_{i+1} - grafo reduzido
$M_i \leftarrow Matching(G_i)$
para cada $aresta(u, v) \in M_i$ **faça**
  $SuperVertice \leftarrow Contrair(u, v)$
  para todo *vértice* x *adjacente a* u *e* v **faça**
    pesoAresta $(SuperVertice, x)$ $\leftarrow$ pesoAresta (u, x) + pesoAresta(v, x)
  fim
fim
retorna G_{i+1}

4.3.2 Fase de Particionamento

Com o grafo reduzido durante a fase de *coarsening*, temos agora a oportunidade de aplicar algoritmos custosos que antes não eram possíveis de serem executados. Utilizando métodos

já implementados nas bibliotecas **Snap**[7] e **Igraph**[8], foram utilizados os seguintes algoritmos para o particionamento do grafo (descritos na Seção 2.3):

- *Girvan-Newman*
- *Clauset-Newman-Moore (CNM)*
- *Walktrap*
- *Louvain*
- *Leading Eigenvector*
- *Label Propagation*

Nessa etapa temos como entrada o grafo reduzido G_i e como saída teremos um particionamento P_i que emprega um dos algoritmos listado acima.

4.3.3 Fase de Uncoarsening

A última etapa da estratégia multinível consiste em fazer o processo reverso ao realizado na fase de *coarsening*. A partir do grafo reduzido G_i são realizadas expansões até se retornar ao grafo inicial G_0.

No algoritmo 3 podemos ver o pseudocódigo da fase de *uncoarsening*. Para cada etapa de expansão, os super-vértices do grafo G_{i+1} são divididos nos vértices geradores u e v, retornando-se a estrutura anterior do grafo G_i. Nesse processo, o particionamento P_{i+1} obtido do grafo reduzido é projetado sobre os vértices do grafo expandido, ou seja, os vértices são colocados na mesma partição que o seu super-vértice.

Algoritmo 3 Fase de *uncoarsening*

Entrada: G_{i+1} - grafo reduzido
Saída: G_i - grafo expandido
para cada $SV \in G_{i+1}$ **faça**
  $(u, v) \leftarrow Separar(SV)$
  $P_i(u) = P_i(v) = P_{i+1}(SV)$
fim
retorna G_i

[7] http://snap.stanford.edu/snap/
[8] http://igraph.org/

5 Experimentos

5.1 Conjunto de dados

O conjunto de dados utilizados nos experimentos são os grafos extraídos da base de dados do sistema *NELL* (descritos na seção 3.2). Analisaremos tanto as informações do grafo original quanto duas outras versões modificadas: o "Grafo sem *url*" (subseção 3.2.1) e o "Grafo transformado sem *url*" (subseção 3.2.3).

5.2 Resultados

Nesta seção serão mostrados os resultados do algoritmo multinível de detecção de comunidade aplicados no conjunto de dados descritos na seção anterior.

Os experimentos foram testados em uma máquina com processador Intel Core i5 (2.3 GHz) e 4 GB de memória. As implementações foram feitas usando a linguagem de programação Python.

A seguir teremos mais detalhes dos resultados obtidos na aplicação do algoritmo multinível.

5.2.1 Fase de Coarsening

Na subseção 4.3.1 foram apresentados alguns detalhes da implementação na fase de *coarsening*. Na Figura 10 temos os resultados dessa fase aplicada nas diferentes versões do grafo do *NELL*.

Analisando o grafo original notamos que o tempo da fase de *coarsening* é extremamente alto, principalmente pelo seu grande tamanho (aproximadamente 680.000 vértices e 440.000 arestas). No canto superior direito do gráfico podemos observar que o tempo de redução para um tamanho de 70% do grafo original levou aproximadamente 6500 segundos. Um tempo bastante elevado, sendo inviável reduções para tamanhos relativos menores.

Para o grafo sem *url* temos os resultados da variação de tempo desde 5% até 70% do seu tamanho relativo, enquanto que para o grafo transformado e sem url temos de 40% até 70%. Percebemos que o tempo de processamento neste segundo é maior. Apesar de ambos possuírem uma quantidade de vértices bem próxima (cerca de 70.000 nós), a versão com o grafo transformado tem 404.100 arestas enquanto que a outra apenas 134.700, explicando essa diferença de tempo.

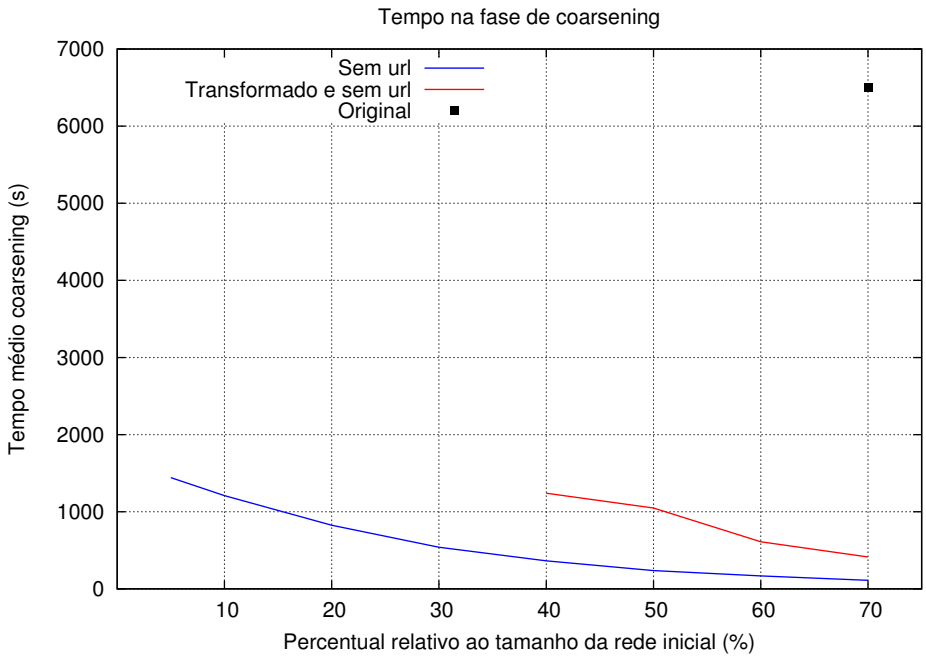


Figura 10: Gráfico da variação de tempo na fase de *coarsening* em função do porcentual do tamanho do grafo.

5.2.2 Fase de particionamento - grafo sem url

Na subseção 4.3.2 foi explicada a fase de particionamento e os diferentes algoritmos utilizados na detecção de comunidades. Na figura 11 podemos observar a variação de tempo para cada um desses métodos. Claramente vemos que o algoritmo *Girvan-Newman* (localizado no canto superior esquerdo) demorou um tempo extremamente elevado para particionar o grafo com apenas 5% de seu tamanho original (aproximadamente 3400 vértices), enquanto que outros algoritmos levaram menos de alguns segundos para processar o mesmo grafo reduzido. Ou seja, fica inviável aplicar o método de *Girvan-Newman* para grafos cada vez maiores.

Também é importante notar que certos algoritmos mantêm o tempo de particionamento constante, como o método de *Louvain* que demora menos de 1 segundo para todos os níveis de redução do grafo. Ao mesmo tempo que métodos como o *CNM* e *Walktrap* começam a apresentar uma curva de subida mais acentuada, demorarando cerca de 6 à 7 minutos para particionar grafos com 70% do tamanho (48.000 vértices).

Na figura 12 temos a porcentagem de comunidades corretamente detectadas em função do tamanho relativo do grafo inicial. Percebemos que grandes reduções no grafo resultam em baixas taxas de detecções corretas, ou seja, temos um índice de acerto de apenas 30-40% quando o tamanho do grafo é de apenas 5% do inicial. Quando a redução se aproxima de 70% do tamanho do grafo, os métodos começam a convergir para valores entre 50% e 60%.

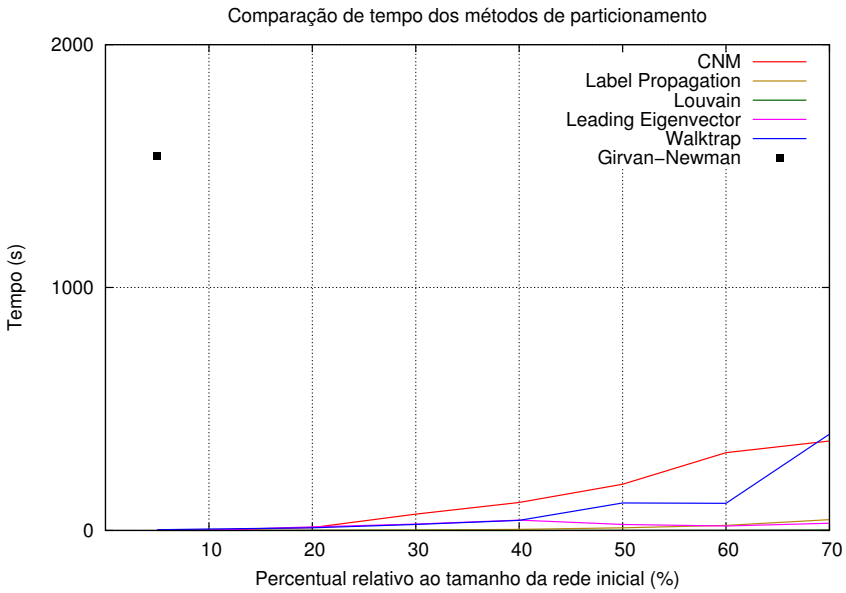


Figura 11: Gráfico dos tempos na fase de particionamento em função do porcentual do tamanho do grafo.

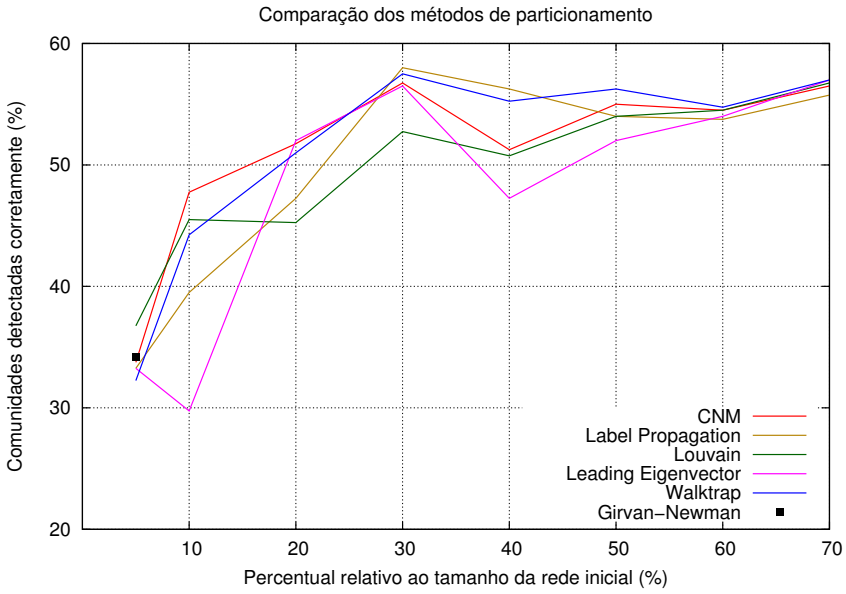


Figura 12: Gráfico da porcentagem das comunidades corretamente detectadas em função do porcentual do tamanho do grafo.

5.2.3 Fase de particionamento - grafo transformado e sem url

Nesta subseção apresentaremos os resultados obtidos na fase de particionamento para o grafo transformado e sem instâncias relacionadas à categoria *url*. Diferente da versão anterior, não analisamos os resultados para o método *Girvan-Newman* devido a seu elevado custo de

execução e tempo de processamento.

Na imagem 13 temos a variação de tempo para cada um dos métodos de particionamento. Os resultados se mostraram similares à versão anterior, onde certos métodos como o *Louvain* mantiveram tempo constante extremamente baixo na fase de particionamento enquanto que outros como o *CNM* e o *Walktrap* com tempos cada vez maiores.

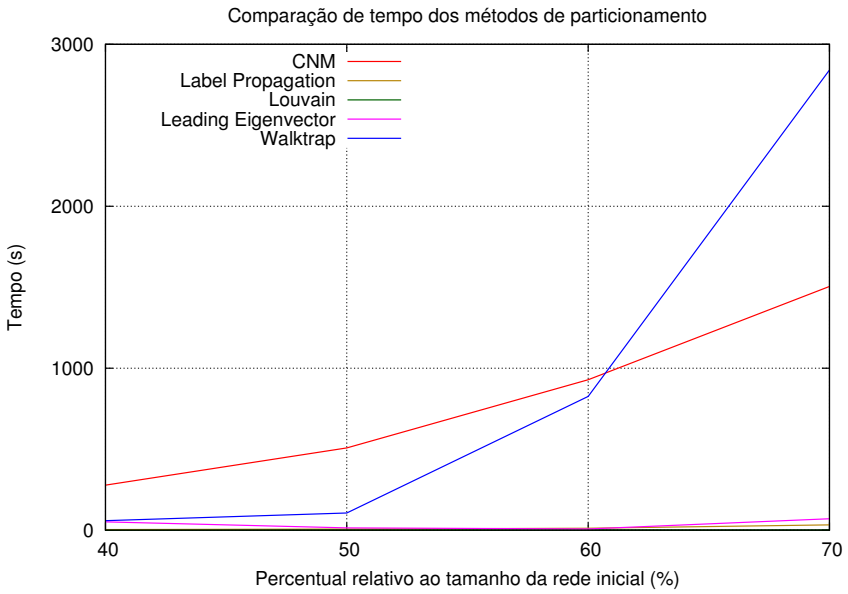


Figura 13: Gráfico dos tempos na fase de particionamento em função do porcentual do tamanho do grafo

A figura 14 mostra o gráfico com a porcentagem de comunidades corretamente detectadas. Também foram obtidos resultados bastante similares à versão anterior. Para o grafo com 70% de seu tamanho a porcentagem de acertos também ficou entre 50% e 60%. Como o intervalo do porcentual relativo à rede nessa versão é menor (variando de 40% à 70%, e anteriormente entre 5% até 70% do tamanho do grafo), não ficou tão evidente que grandes reduções no grafo resultam em baixas taxas de detecções.

5.2.4 Grafos resultantes

Utilizando a ferramenta de visualização de grafos **Gephi**[9], ilustramos alguns grafos resultantes dos experimentos.

Nas figuras 15a e 15b temos as imagens do grafo sem *url*. À esquerda o grafo foi colorido em relação as suas categorias (como cidade, jogador, música) e à direita pelo resultado do algoritmo multinível de detecção de comunidades (mais especificamente, com o método

[9]http://gephi.github.io/

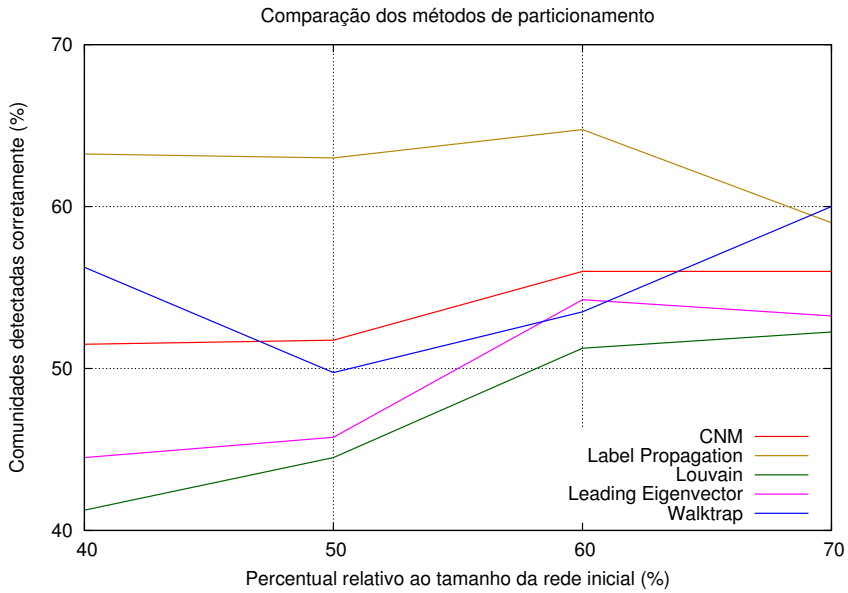
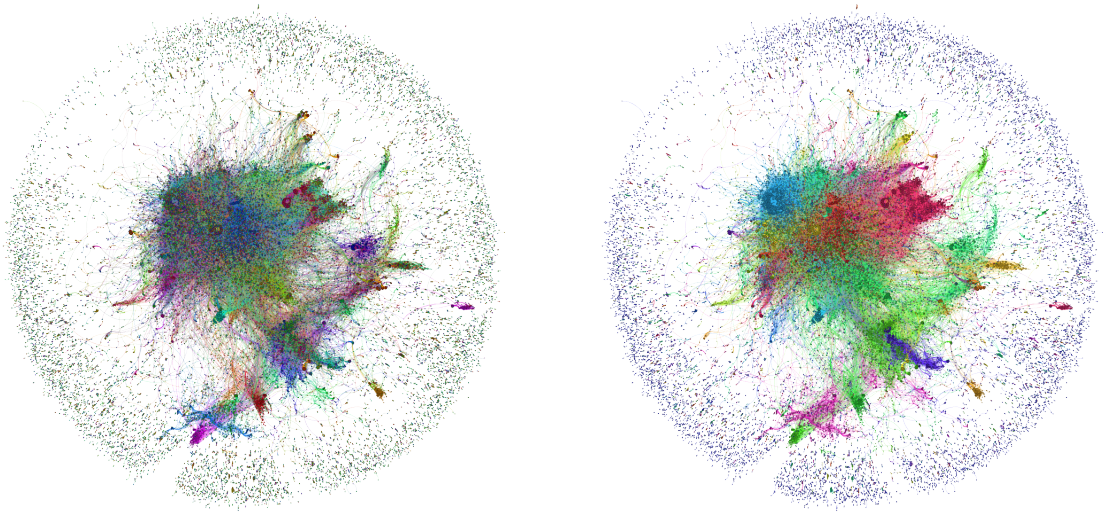


Figura 14: Gráfico da porcentagem das comunidades corretamente detectadas em função do porcentual do tamanho do grafo.

Louvain). Com um total de 274 categorias, a intenção era poder separar as instâncias do grafo nessas 274 partições de acordo com seu tipo. Neste exemplo, foram detectadas 1765 comunidades e em média as comunidades possuíam cerca de 55% de predomínio em uma categoria.



(a) Grafo colorido pelas categorias.

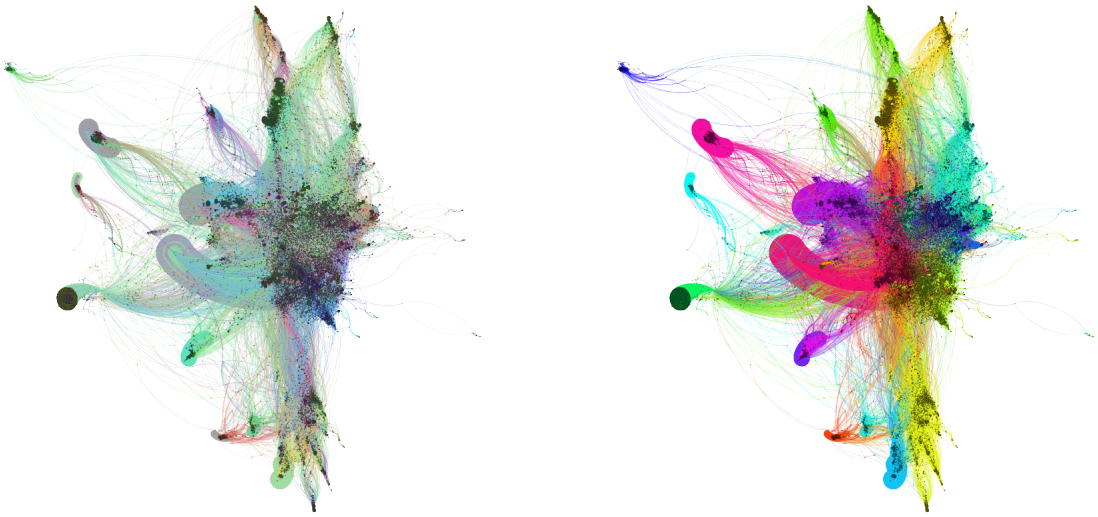
(b) Utilizando algoritmo multinível de detecção de comunidades.

Figura 15: Grafo sem instâncias relacionadas às *urls*

Já na versão transformada podemos observar algumas difenças. Assim como no grafo

anterior, à esquerda (figura 16a) temos o grafo colorido pelas categorias e à direita (figura 16b) utilizando estratégia multinível para detecção de comunidades (método de *Louvain* na fase de particionamento).

Uma das grandes características do grafo transformado foi obter uma rede mais conectada. Essa diferença teve um grande impacto nos resultados obtidos. Com um grafo mais densamente conectado os algoritmos na fase de particionamento tenderam a criar comunidades menores. De um total de 274 categorias, foram detectadas apenas 27 comunidades. Apesar de ser um número baixo, o índice de categorias dominantes nas comunidades foi de aproximadamente 50%.



(a) Grafo colorido pelas categorias.

(b) Utilizando algoritmo multinível de detecção de comunidades.

Figura 16: Grafo Transformado e sem instâncias relacionadas às *urls*

6 Considerações finais

6.1 Conclusão

Um dos grandes intuitos de se utilizar a estratégia multinível era poder aplicar algoritmos custosos para a detecção de comunidades no grafo do *NELL* (grande quantidade de vértices e arestas). Entretanto, o tempo de processamento ainda se mostrou bastante elevado, tornando-se inviável utilizar métodos como o *Girvan-Newman* na fase de particionamento.

Certas soluções mostraram-se bastante satisfatórias como o método de *Louvain* na fase de *coarsening*. Mas em geral, o tempo de processamento do algoritmo multinível é bastante elevado, principalmente por ser dividido em três etapas e cada uma delas podendo ter um custo de execução caro.

O maior desafio no problema de detecção de comunidades é entender que tipo de resultados são esperados. Há inúmeras maneiras de se avaliar a qualidade das comunidades detectadas. Este trabalho mostra uma solução de algoritmo multinível aplicando diversos métodos em sua fase de particionamento que visam maximizar conexões entre vértices de um mesmo grupo e minimizar ligações com vértices de grupos distintos.

6.2 Trabalhos futuros

Para os próximos passos, temos os seguintes pontos:

- **Refinamento na fase de uncoarsening**: Na fase de *uncoarsening* utilizamos uma estratégia simples, onde os nós que compõe um super-vértice são colocados ambos na mesma partição. Um próximo passo seria aplicar uma etapa de refinamento, possibilitando a mudança de partição desses vértices. O objetivo do refinamento seria poder maximizar a medida de agrupamento (como a modularidade) conforme o grafo é expandido.

- **Análise qualitativa**: Na análise dos resultados do algoritmo multinível foram levados em consideração apenas os aspectos estruturais da rede. Uma outra forma de análise seria entender que tipos de instâncias as pessoas consideram similares. Sabendo quais categorias possuem valores próximos, poderemos usar uma outra abordagem na análise dos resultados.

- **Aprendizagem supervisionada**: Assim como no item anterior, seria interessante poder detectar comunidades de forma qualitativa, não levando em consideração apenas

a topologia do grafo. Uma solução seria a implementação de um algoritmo de detecção de comunidades que utilizasse aprendizado de máquina (supervisionado) para auxiliar na divisão da rede. Levando em consideração tanto os aspectos estruturais quanto qualitativos isso possibilitaria uma melhoria na qualidade dos resultados.

Referências

[1] Adamic, L. A., Huberman;, B. A., Barabási, A., Albert, R., Jeong, H., and Bianconi;, G. **Power-law distribution of the world wide web**. Science, 287(5461):2115a+. (2000)

[2] Albert, R., Jeong, H., and Barabasi, A.-L. **The diameter of the world wide web**. (1999)

[3] Appel, A. P. and Hruschka Jr., E. R. **Centaurs a component based framework to mine large graphs**. In XXV Brazilian symposium on databases, pages 1–8, Belo Horizonte, MG, Brazil. (2010)

[4] Betteridge, J., Carlson, A., Hong, S. A., Hruschka Jr., E. R., Law, E. L. M., Mitchell, T. M., and H.Wang, S. **Toward never ending language learning**. In AAAI Spring Symposium. (2009).

[5] Chakrabarti, D., Wang, Y., Wang, C., Leskovec, J., and Faloutsos, C. (2008). **Epidemic thresholds in real networks**. ACM Trans. Inf. Syst. Secur., 10(4):1–26.

[6] Clauset, A., Moore, C., and Newman, M. E. J. (2008). **Hierarchical structure and the prediction of missing links in networks**. Nature, 453(7191):98–101. [Fortunato, 2010] Fortunato, S. Community detection in graphs. Physics Reports, 486(3-5):75–174. (2010).

[7] Leskovec, J., Backstrom, L., Kumar, R., and Tomkins, A. **Microscopic evolution of social networks**. In KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 462–470, New York, NY, USA. ACM. (2008).

[8] Leskovec, J., Kleinberg, J., and Faloutsos, C. **Graphs over time: densification laws, shrinking diameters and possible explanations**. In eleventh ACM SIGKDD, pages 177–187, New York, NY, USA. ACM Press. (2005).

[9] Liben-Nowell, D. and Kleinberg, J. **The link prediction problem for social networks**. In CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management, pages 556–559, New York, NY, USA. ACM. (2003).

[10] Milgram, S. **The small world problem**. Psychology Today, 2:60-67. (1967).

[11] Boccaletti, S., Latorab, V., Morenod, Y., Chavez, M., Hwang, D.U.: **Complex networks: structure and dynamics**. Physics Reports 424, 175-308. (2006).

[12] Newman, M. E. J. **The structure and function of complex networks**. SIAM Review, 45:167–256. (2003).

[13] Taskar, B., Wong, M., Abbeel, P., and Koller, D. **Link prediction in relational data**. (2004)

[14] Mitchell, T. M., Betteridge, J., A. C. E. R. H. R. W. **Populating the semantic web by macro-reading internet text**. In Proceedings of the 8th ISWC. (2009)

[15] Cohen, J. E., Briand, F. and Newman, C. M. **Community Food Webs: Data and Theory** (Springer, Berlin, 1990)

[16] Jeong H., Tombor, B., Albert, R., Oltavi, Z, N. and Barabási, A.-L. **The large-scale organization of metabolic networks**. Nature 407, 651–654 (2000).

[17] Broder, A. et al. **Graph structure in the web**. Comput. Netw. 33, 309–320 (2000).

[18] Faloutsos, M., Faloutsos, P. and Faloutsos, C. **On power-law relationships of the internet topology**. Comp. Comm. Rev. 29, 251–262 (1999).

[19] Newman, M. E. J. **The structure of scientific collaboration networks**. Proc. Natl Acad. Sci. USA 98, 404–409 (2001).

[20] Wasserman, S. and Faust, K., **Social Network Analysis**, Cambridge University Press, Cambridge (1994).

[21] Reddy, K. P., M. Kitsuregawa, P. Sreekanth, and S. S. Rao: **Proceedings of the Second International Workshop on Databases in Networked Information Systems** (Springer-Verlag, London, UK), pp. 188–200. In DNIS '02. (2002)

[22] Erdős, P. and Rényi, A. **On random graphs**. Publicationes Mathematicae, 6:290–297 (1959).

[23] Barabási, A.-L., editor. **Linked: How Everything is Connected to Everything Else and what it Means for Business, Science, and Everyday Life**. Plume (2003).

[24] D.J. Watts and S.H. Strogatz. **Collective dynamics of 'small-world' networks**. Nature 393, 440-442 (1998).

[25] Almeida, Leonardo J. **Detecção de comunidades em redes complexas utilizando estratégia multinível**. ICMC-USP (2009)

[26] Karypis, G. e Kumar, V. **Multilevel k -way partitioning scheme for irregular graphs**. Journal of Parallel and Distributed Computing, página 10 pp. (1998)

[27] Albert, R. and Barabási, L.A., **Statistical Mechanics of Complex Networks**, Rev. Mod. Phys. 74, 47-97 (2002)

[28] Rodrigues, David M. de S., **Detecção de Comunidades no sistema de correio electrónico universitário**. Instituto Superior de Ciências do Trabalho e da Empresa. (2009)

[29] Fortunato, Santo, **Community Detection in Graphs**. Physics Reports 486, 75-174 (2010)

[30] Barabási, A.-L., Albert R. and Jeong H., **Mean-field theory for scale-free random networks**, Physica A, 272, 173-187 (1999)

[31] Barabási, A.-L., Albert R. and Jeong H., **Scale-free characteristics of random networks: the topology of the world-wide web**. Physica A: Statistical Mechanics and its Applications 281 (1), 69-77 (2000)

Parte II

Parte Subjetiva

1 Desafios Encontrados

Provavelmente um dos grandes desafios daqueles que cursam a disciplina de conclusão de curso é conseguir conciliar o projeto juntamente com a graduação, vida pessoal e outras atividades como estágio. Comigo não foi diferente, mas nesse ponto fui sortudo em poder alinhar meu trabalho de conclusão com assuntos relacionados ao meu estágio. Por estar no setor de pesquisas da *IBM*, mais precisamente no grupo de análise de dados sociais, pude ter contato com projetos de manipulação de grafos, mineração de dados e análise de sentimento. O conhecimento que adquiria na leitura de artigos e pesquisa na literatura contribuíam tanto para as tarefas do estágio quanto para o trabalho de conclusão, o que tornava os esforços mais centrados.

Por ser um projeto extenso e de duração de quase 1 ano é extremamente difícil se manter disciplinado e focado durante todo o tempo. Fazer reuniões a cada 2 semanas propiciaram um trabalho contínuo que evoluía gradativamente. Compromentendo-me a mostrar pequenos resultados em intervalos curtos ajudava a sempre manter uma parte da mente ocupada com alguma tarefa do trabalho.

A redação da monografia se mostrou uma das atividades mais díficeis no desenvolvimento deste trabalho. Durante todo o curso de ciência da computação estamos acostumados mais na parte de programação e são raros os momentos em que temos de redigir um texto científico formal. Foi um desafio que me fez sair da zona de conforto, mas que teve um ponto extremamente positivo. No momento em que sentamos e começamos a pensar profundamente sobre um tópico é que se consolidam melhor as ideias. Escrever a monografia me ajudou a entender melhor sobre o problema a ser resolvido pois as ideias não ficavam apenas na cabeça mas se concretizavam em forma de texto.

2 Disciplinas relevantes e aplicações dos conceitos

Descrição das disciplinas mais relevantes para o trabalho:

MAC0110 - Introdução à Computação

MAC0122 - Princípios de Desenvolvimento de Algoritmos
MAC0323 - Estruturas de Dados

Matérias essenciais dentro da grade curricular do Bacharelado em Ciência da Computação do IME-USP. Com esse conhecimento inicial e fundamental pude aprender sobre os principais conceitos da computação e despertar meu interesse na programação. Serviram de base para todo o conhecimento aplicado no trabalho.

MAC0328 - Análise de Algoritmos

Saber sobre a complexidade dos algoritmos me ajudou a compreender melhor sobre o tempo de execução dos métodos utilizados. Pelo fato do problema ser bastante difícil e ainda não ser resolvido de forma satisfatória, com o conhecimento adquirido na disciplina foi possível entender quais soluções eram possivelmente viáveis.

MAC0460 - Aprendizagem Computacional

Apesar de não ter aplicado nenhum tipo de algoritmo de aprendizado de máquina neste trabalho, uma das intenções futuras é poder implementar uma versão utilizando aprendizagem supervisionada para a detecção de comunidades. Além disso, o sistema NELL é um sistema de aprendizado de máquina semântico, o que facilitou na compreensão sobre parte de seu funcionamento.

MAC0328 - Linguagens Formais e Autômatos

O conhecimento obtido sobre expressões regulares foram extremamente úteis na criação de scripts para o parseamento de valores e relações da base de dados.

MAC0328 - Algoritmos em Grafos

Uma das disciplinas mais importantes no trabalho dentre todas aqui descritas. Pelo fato das redes complexas serem modeladas em grafos, os algoritmos de detecção de comunidades são também aplicados na estrutura de grafos. A todo momento foram utilizados conceitos estudados nesta disciplina.