Redes Complexas

Gustavo Fernandes Carneiro de Castro[1] **- 11369684**
Mateus Miquelino da Silva[1] **- 11208412**

[1]Bacharelado em Ciência da Computação - Departamento de Computação e Matemática
Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto (FFCLRP - USP/RP)
Av. Bandeirantes, 3900 - Monte Alegre - Ribeirão Preto - SP - CEP: 14040-901

gus.castro@usp.br, m.miquelino@usp.br

Abstract. *In this paper will be described, caracterized and analized a complex network created using the database from FOLDOC, a dictionary of jargons, acronyms, names and nomeclatures of words in the area of Computation and Information Technology, including from terms to company names. First, the network will me indroduced and described. Then, it will be done a data characterization and, lastly, it will be done an analysis with final considerations about this network.*

Resumo. *Neste documento será descrita, caracterizada e analisada uma rede complexa retirada do banco de dados do FOLDOC, um dicionário de jargões, acrônimos, nomes e nomenclaturas de palavras da área de Computação e Tecnologia da Informação, incluindo desde termos a nomes de empresas. Primeiramente, a rede complexa será introduzida e descrita. Depois, será feita uma caracterização dos dados desta rede complexa e, por último, será feita uma análise com considerações finais sobre esta rede.*

1. Introdução

O FOLDOC é um dicionário de jargões, acrônimos, nomes e nomenclaturas de palavras da área de Computação e Tecnologia da Informação. A rede complexa analisada foi retirada da relação de descrição das palavras descritas neste dicionário. Ou seja, supondo um arco formado de duas palavras (X, Y), X está conectado a Y se Y se encontra na descrição de X. Qualquer análise feita dentre um grupo de palavras pode ser útil para uma maior compreensão das linguagens naturais. Portanto, ao analisar uma rede complexa deste tipo, é possível descobrir informações úteis para o futuro processamento de linguagem natural, levando em conta os tópicos de um área específica, neste caso a área de Computação.

2. Caracterização da rede

A rede complexa $foldoc.net$ é representada por um grafo e foi analisada pela biblioteca $NetworkX$ na linguagem $Python$. O grafo foi criado originalmente como um multigrafo direcionado ponderado. Para um maior acesso as funções e algoritmos da biblioteca, foram sacrificados 3 arestas para deixá-lo apenas como um grafo direcionado ponderado. Algumas características desta rede após a transformação são:

2.1. (a) Tipo de rede

O grafo é um grafo direcionado e ponderado.

2.2. (b) Número de Nós e Arestas

O número total de nós é de 13355, e o número total de arestas é de 120235 (120238 como um multigrafo direcionado, com perda de 3 arestas na transformação).

2.3. (c) Grau máximo, mínimo e médio

Sabendo que o grafo analisado é direcionado, os graus são separados em grau de entrada e grau de saída. Para este grafo direcionado e ponderado, os graus de entrada e saída máximos, mínimos e médios, para o grafo ponderado, são:

Grau de entrada máximo do Grafo: 819.0
Grau de entrada mínimo do Grafo: 0
Grau de entrada médio do Grafo: 9.375

Grau de saída máximo do Grafo: 289.0
Grau de saída mínimo do Grafo: 3.0
Grau de saída médio do Grafo: 9.375

2.4. (d) Densidade

A densidade da rede é de 0.000674.

2.5. (e) Número médio de triângulos

A função do $NetworkX$ não funciona para grafos direcionados, portanto foi calculado a média de triângulos para a rede transformada em um grafo não direcionado, utilizando $to\_undirected()$. O número médio, nessas condições, é: 23.4.

2.6. (f) Média do coeficiente de agrupamento (clustering)

A média do coeficiente de agrupamento é igual a 0.02034

2.7. (g) Diâmetro da rede

O diâmetro da rede não é calculável para este grafo pois ele não é fortemente conectado, ou seja, nem todos os vértices tem acesso a todos os outros e, portanto, não é possível calcular o diâmetro utilizando a função do $NetworkX$

2.8. (h) Número de componentes conexos

As funções $node\_connectivity$ e $average\_node\_connectivity$, que devolvem a conectividade de cada nó e a média das conectividades dos nós, respectivamente, são funções cujo tempo de execução é extremamente alto, impedindo delas devolverem a conectividade. Além disso, a função $k\_components$ não funciona para grafos direcionados, impedindo a obtenção do número de componentes conexos deste grafo.

2.9. Nós mais Importantes

Após as análises numéricas, foram definidos os 10 nós mais importantes,. Para tal, foram sugeridos 3 métodos, dos quais apenas 2 foram utilizados. Os resultados de cada análise são:

2.9.1. Betweenness Centrality

O algoritmo que calcula Betweenness Centrality teve um tempo de execução extremamente alto e, portanto, não foi possível de calcular os principais nós a partir dele.

2.9.2. Graus de entrada

('Unix', 819.0)
('Usenet', 477.0)
('C', 443.0)
('IBM', 415.0)
('Internet', 413.0)
('operating system', 390.0)
('protocol', 373.0)
('MS-DOS', 295.0)
('standard', 255.0)
('ASCII', 254.0)

2.9.3. Graus de saída

('W2K', 289.0)
('ASCII', 195.0)
('486', 145.0)
('OSI', 98.0)
('WEB', 96.0)
('Emacs', 86.0)
('Abstract Syntax Notation 1', 72.0)
('URL', 72.0)
('object-oriented programming', 64.0)
('Windows sockets', 64.0)

2.9.4. Eigenvector Centrality

('Unix', 0.33609803832444873)
('operating system', 0.2212892922742617)
('Usenet', 0.1982098744942228)
('Internet', 0.17764976685571843)
('IBM', 0.17055632261368098)
('Intel 80386', 0.15740494744960148)
('clock rate', 0.1549475110579293)
('IBM PC', 0.15166315849826476)
('protocol', 0.1459247641357533)
('C', 0.14305606312953956)

2.10. Código de análise e caracterização:

Link para o Colab

```
import networkx as nx
from google.colab import files
graphfile = files.upload()

# Justificativa para ISO-8859-1:
# https://stackoverflow.com/questions/61668411/ansi-encoding-for-pandas-on-google-colab
G = nx.read_pajek("/content/foldoc.net", "ISO-8859-1")
# Alterar de um Multi Directed Graph para um Directed Graph (perda de 3 arestas)
G = nx.DiGraph(G)

print("Informações Gerais: ")
print(nx.info(G))
print()

print("(a) Tipo de rede (direcionada ou não, ponderada ou não)")
print("Gráfico direcionado: " + str(nx.is_directed(G)))
print("Gráfico ponderado: " + str(nx.is_weighted(G)))
print()

print("(b) Número de nós e arestas")
print("Número de nós: " + str(nx.number_of_nodes(G)))
print("Número de arestas: " + str(nx.number_of_edges(G)))
print()

print("(c) Grau máximo, mínimo e médio")
inDegreeDict = dict(G.in_degree(weight = "weight"))
inMaxDegree = max(inDegreeDict.values())
inMinDegree = min(inDegreeDict.values())
inAvgDegree = sum(inDegreeDict.values())/len(inDegreeDict.values())

outDegreeDict = dict(G.out_degree(weight = "weight"))
outMaxDegree = max(outDegreeDict.values())
outMinDegree = min(outDegreeDict.values())
outAvgDegree = sum(outDegreeDict.values())/len(outDegreeDict.values())

print("Grau de entrada máximo do Grafo: " + str(inMaxDegree))
print("Grau de entrada mínimo do Grafo: " + str(inMinDegree))
print("Grau de entrada médio do Grafo:  " + str(inAvgDegree))
print()
print("Grau de saída máximo do Grafo: " + str(outMaxDegree))
print("Grau de saída mínimo do Grafo: " + str(outMinDegree))
print("Grau de saída médio do Grafo:  " + str(outAvgDegree))
print()

print("(d) Densidade da rede")
print("Desidade da rede: " + str(nx.density(G)))
print()

print("(e) Número médio de triângulos")
trianglesDict = dict(nx.triangles(G.to_undirected()))
avgTriangles = sum(trianglesDict.values())/len(trianglesDict.values())
print("Número médio de triângulos: " + str(avgTriangles))
print()

print("(f) Média do coeficiente de agrupamento (clustering)")
print("Média do coeficiente de agrupamento: " + str(nx.average_clustering(G, weight = "weight")))
print()

print("(g) Diâmetro da rede")
#print("Diâmetro da rede: " + str(nx.diameter(G)))
print()

print("(h) Número de componentes conexos")
#print("Número de componentes conexos: " + str(nx.k_components(G)))
print()

#print("Betweennes Centrality: " + str(nx.betweenness_centrality(G)))
```

```
print()


print("10 nós mais importantes (pelo grau de entrada):")
sortedDegreeList = list(sorted(inDegreeDict.items(), key = lambda item: item[1], reverse = True))

for i in range(10):
    print(sortedDegreeList[i])
print()

print("10 nós mais importantes (pelo grau de saída):")
sortedDegreeList = list(sorted(outDegreeDict.items(), key = lambda item: item[1], reverse = True))

for i in range(10):
    print(sortedDegreeList[i])
print()

print("10 nós mais importantes (pela centralidade do autovetor): ")
engenCentrDict = dict(nx.eigenvector_centrality(G, weight = "weight"))
sortedEngenCentrList = list(sorted(engenCentrDict.items(), key = lambda item: item[1], reverse = True))

for i in range(10):
    print(sortedEngenCentrList[i])
print()
```

3. Considerações Finais

O grafo analisado é ponderado e direcionado, onde o nó de saída é definido no dicionário pelos nós de entrada. Levando em conta os graus de entrada e saída, e possível concluir que uma palavra é definida no dicionário FOLDOC por pelo menos três outras, observando o grau mínimo de saída; mas existem palavras que não definem nenhuma outra, observando o grau mínimo de entrada. Analisando a densidade, é possível de observar que esta rede não é densa, ou seja, possui, em comparação com o número máximo de arestas possíveis, uma baixa conectividade entre seus nós. Ao calcular, é possível adquirir uma média de 9 arestas por nó (grau médio), em comparação a um máximo de 13354 por nó (onde todos os nós estão conectados).

Para analisar o número médio de triângulos, sabendo que a função não calculava para grafos direcionados, foi utilizada uma transformação do grafo direcionado para um não-direcionado, levando em conta que os tipos de triângulos do grafo direcionado não importam. Com isso, mesmo com uma perda significativa no número de arestas, é possível ter uma ideia geral da média dos triângulos, que se firmou em torno de 23. Analisando o coeficiente de agrupamento, é possível definir que o grafo não tem uma grande quantidade de grupos separados formados.

Sobre o diâmetro e o número de componentes conexos, são funções que, ou o tempo de execução é altíssimo, impedindo o acesso a estes dados, ou a função não é implementada para o tipo do grafo analisado. A transformação do grafo para um mais simples pode não entregar um resultado próximo o bastante para valer a aproximação, e portanto não foi utilizada nestes métodos.

Para a analise dos nós mais importantes, não foi possível utilizar o algoritmo de Betweennes Centrality por conta de um tempo alto de execução. Porém, analisando os outros dois métodos, é possível observar que aqueles com um maior grau de entrada são aqueles que explicam o maior número de palavras, e aqueles com um maior grau de saída são aqueles que são explicados por um maior número de palavras. Além disso, utilizando o algoritmo Eigenvector Centrality, que calcula a centralidade de um nó baseada na centralidade de seus vizinhos, não se importando com arestas de entrada e saída, é possível perceber que,

como é mais propenso uma palavra explicar várias, do que uma palavra ser explicada por muitas, a lista se aproxima da lista formada pela análise dos graus de entrada.

Referências

Links de cada referência e link para o Colab:

Link para o Colab

Network Science.

foldoc.net Reference.

Network analysis of dictionaries.

NetworkX Documentation.

4. Contribuição Individual

O autor Gustavo Castro ficou no encargo de escrever, organizar e formatar e os documentos do LaTex e Colab, além de escrever os códigos e auxiliar na pesquisa. Já o autor Mateus Miquelino ficou de selecionar as funções, pesquisar a biblioteca e separar os conhecimentos gerais de cada função e algoritmo, além de revisar o documento.