

UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

Igor Kenji Kawai Ueno

**Estudo Comparativo de Métodos de Embeddings
de Grafo para Tarefa de Agrupamento com Foco
na Análise de Robustez ao Ruído**

São Carlos - SP

2026

Igor Kenji Kawai Ueno

**Estudo Comparativo de Métodos de Embeddings de Grafo
para Tarefa de Agrupamento com Foco na Análise de
Robustez ao Ruído**

Trabalho de Conclusão de Curso apresentado
ao curso de Engenharia de Computação da
Universidade Federal de São Carlos, como re-
quisito para a obtenção do título de Bacharel
em Engenharia de Computação.

Orientação Prof. Dr. Alan Demétrius Baria
Valejo

São Carlos - SP

2026

Dedico este trabalho à minha família e meus amigos por sempre me apoiarem

Agradecimentos

Gostaria de reservar este espaço para demonstrar minha gratidão às pessoas que estiveram ao meu lado durante minha jornada acadêmica e ao longo de minha vida.

Agradeço à minha mãe, muito obrigado, mama, por sempre estar do meu lado e me apoiar em todos os momentos da minha vida, mesmo quando não nos vemos todo dia, sinto como se nunca tivéssemos parado de nos encontrar.

Agradeço ao meu pai, muito obrigado, papa, por sempre se preocupar comigo, ainda que não demonstre tanto, sei que se importa bastante, obrigado pelas conversas e conselhos sobre assuntos diversos, mesmo tendo visões diferentes às vezes.

Agradeço à minha irmã, muito obrigado, Manami-chan, por sempre me apoiar e fazer companhia, mesmo que cada um fique no seu canto, ainda sinto que continuamos muito próximos e sou grato por sempre me aturar e dar suporte para as ideias que tenho.

Agradeço aos meus avós, muito obrigado Ueno jichan, Ueno bachan, Kawai jichan e Kawai bachan, sou imensamente grato por todo o esforço e dedicação para construir nossa família, tanto em Botucatu quanto em Atibaia, e por permitirem que tenhamos uma vida sem grandes preocupações com um teto e comida na mesa, ありがとうございます.

Agradeço à minha família, muito obrigado, desde sempre nos encontramos anualmente para conversar e atualizar sobre como estamos, quero continuar podendo compartilhar essas experiências com vocês.

Agradeço aos meus amigos do Elda, muito obrigado, mesmo que não nos encontremos faz um tempo, guardo os momentos juntos em um espaço especial, muito obrigado por tornar os tempos na escola mais leves e divertidos.

Agradeço aos meus amigos da Embraer, muito obrigado, seja durante o colégio ou depois dele, jogando frisbee ou vôlei, caminhando pelo Lageado ou jogando alguma coisa, esses momentos são especiais e sou grato por cada um deles.

Agradeço aos meus amigos da UFSCar, muito obrigado, pelo suporte dentro e fora da faculdade, juntos nos trabalhos das disciplinas e me acompanhando nos mais variados lugares, seja uma festa, uma chácara ou mesmo uma resenha no apartamento.

Agradeço ao meu orientador, muito obrigado, por me apoiar durante o desenvolvimento deste trabalho e pelo suporte dentro do PANDA, ainda que esteja em cima da hora, sou grato pela ajuda e mentoria durante esse tempo.

Agradeço também a todas as pessoas que não mencionei diretamente aqui, muito obrigado, por seu apoio e suporte em qualquer momento da minha vida.

*”O que realmente importa são as emoções subjacentes,
que você nunca as abandone.”*

(Hayao Miyazaki)

Resumo

Atualmente, a produção de informação é a maior da história, crescendo ano após ano, sem mostrar sinais de que irá desacelerar em um futuro próximo. Esse grande volume de dados apresenta diversas origens e formatos, desde a leitura de sensores de dispositivos de IoT, como *smartwatches*, até o compartilhamento de vídeos em redes sociais. Tudo isso compõe o conjunto massivo de dados e a área de estudo conhecida como *Big Data*, que pode ser descrita por meio dos 5 Vs: volume, velocidade, variedade, veracidade e valor, os quais resumem a riqueza dessa fonte quase ilimitada de dados. Mas não apenas sua produção foi amplificada; a qualidade e a complexidade desses dados também foram afetadas, com tabelas, imagens e redes cada vez maiores e mais detalhadas. Embora possuam alto potencial, sendo possível extrair informações e valor desses dados, sua forma bruta não é adequada para ser utilizada diretamente pela maioria dos algoritmos de Aprendizado de Máquina. É necessário um pré-processamento para refiná-los, otimizando assim seu desempenho. Dentre as tarefas dessa área, o agrupamento busca identificar padrões nos dados a fim de separá-los em grupos com base em sua similaridade; isso é feito sem a presença de um rótulo externo que ditaria essa distribuição. Quando considerado no contexto de grafos, o agrupamento pode ser traduzido para a tarefa de detecção de comunidades, na qual, além das possíveis *features* dos nós, também são consideradas as conexões entre eles. Outra maneira de abordar esse problema pode ser encontrada na aplicação do *embedding* de grafos, que transforma os nós em vetores numéricos em um espaço latente preservando noções de proximidade do espaço original; assim, ambas as tarefas (agrupamento e detecção de comunidades) tornam-se equivalentes. O processo de *embedding* pode ser aplicado sobre diversos tipos de dados e diferentes componentes a fim de unificar sua representação. Tendo isso em mente, este trabalho tem como objetivo utilizar diferentes modelos de *embedding* e de agrupamento sobre um conjunto de grafos, a fim de avaliar seus desempenhos na formação de grupos de nós, sendo a detecção de comunidades análoga a este caso. Além disso, foram realizados testes de robustez desses modelos em relação à presença de ruídos nos grafos, tendo em vista que dados reais frequentemente apresentam informações ruidosas que podem acabar não sendo tratadas e afetar a qualidade das tarefas realizadas sobre eles. Ao final do trabalho, foi possível observar que cada cenário de grafo e combinação de modelos apresenta resultados e comportamentos particulares, com diferentes hiperparâmetros a serem definidos e diferentes níveis de robustez ao ruído. Embora haja resultados promissores em alguns casos, a viabilidade dessa abordagem exige um estudo mais aprofundado para ser confirmada ou descartada.

Palavras-chave: aprendizado de máquina, agrupamento, embedding, grafo, rede, detecção de comunidade, ruído.

Abstract

The production of information is currently at its highest level in history, growing year after year with no signs of slowing down in the near future. This massive volume of data originates from a wide variety of sources and formats, ranging from sensor readings collected by Internet of Things (IoT) devices, such as smartwatches, to video sharing on social media platforms. Together, these data constitute the massive body of information and the field of study known as Big Data, which can be characterized by the 5 Vs: volume, velocity, variety, veracity, and value, summarizing the richness of this virtually unlimited source of data. However, not only has data production increased, but the quality and complexity of these data have also evolved, with tables, images, and networks becoming increasingly larger and more detailed. Although these data hold great potential, allowing valuable information and insights to be extracted, their raw form is not suitable for direct use by most Machine Learning algorithms. Therefore, preprocessing is required to refine the data, thereby improving the performance of these models. Among the tasks in this field, clustering aims to identify patterns in data in order to partition them into groups based on their similarity, without relying on external labels that define such a partition. When considered in the context of graphs, clustering can be translated into the task of community detection, in which, besides the possible features associated with the nodes, the connections between them are also taken into account. Another approach to this problem is graph embedding, which transforms graph nodes into numerical vectors in a latent space while preserving notions of proximity from the original space. As a result, clustering and community detection become equivalent tasks. The embedding process can be applied to different types of data and various data components in order to provide a unified vector representation. With this in mind, the objective of this work is to evaluate the performance of different graph embedding and clustering models on a collection of graphs for the task of grouping nodes, with community detection serving as the analogous problem in the graph domain. In addition, robustness analyses were conducted to evaluate the resilience of these models to the presence of noise in graphs, since real-world data frequently contain noisy information that may remain untreated and negatively affect the quality of downstream tasks. The experimental results demonstrate that each graph scenario and combination of embedding and clustering models exhibits distinct behaviors and performance characteristics, requiring different hyperparameter configurations and presenting different levels of robustness to noise. Although promising results were obtained in some cases, the practical viability of this approach requires further investigation before it can be conclusively confirmed or ruled out.

Keywords: machine learning, clustering, embedding, graph, network, community detection, noise.

Lista de ilustrações

Figura 1 – Um grafo direcionado e suas representações.	35
Figura 2 – Arquiteturas do Word2Vec.	43
Figura 3 – Exemplo de embedding de grafo.	44
Figura 4 – <i>Overview</i> do DeepWalk.	45
Figura 5 – Exemplo de grafo com destaque na vizinhança dos nós.	45
Figura 6 – Estratégias de busca em grafo.	46
Figura 7 – Comunidades definidas por node2vec com diferentes parâmetros.	47
Figura 8 – <i>Framework</i> do modelo de rede neural semi-supervisionado do SDNE.	48
Figura 9 – Exemplo de grafo com nós estruturalmente semelhantes.	48
Figura 10 – Exemplo de grafo do tipo barbell colorido de acordo com a semelhança estrutural dos nós.	49
Figura 11 – Grafos com diferentes níveis de ruído negativo aplicado.	60
Figura 12 – Grafos com ruído negativo nulo e máximo.	61
Figura 13 – Grafos com diferentes níveis de ruído positivo aplicado.	62
Figura 14 – Grafos com ruído positivo nulo e máximo.	62
Figura 15 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Zachary’s Karate Club usando os melhores parâmetros encontrados.	64
Figura 16 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Dolphins usando os melhores parâmetros encontrados.	64
Figura 17 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Football usando os melhores parâmetros encontrados.	65
Figura 18 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Political Blogs usando os melhores parâmetros encontrados.	66
Figura 19 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Zachary’s Karate Club com diferentes ruídos.	68
Figura 20 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Zachary’s Karate Club com diferentes ruídos.	69
Figura 21 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Zachary’s Karate Club com diferentes ruídos.	69
Figura 22 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Zachary’s Karate Club com diferentes ruídos.	91

Figura 23 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Zachary’s Karate Club com diferentes ruídos.	92
Figura 24 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Zachary’s Karate Club com diferentes ruídos.	92
Figura 25 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Zachary’s Karate Club com diferentes ruídos.	93
Figura 26 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Zachary’s Karate Club com diferentes ruídos.	93
Figura 27 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Zachary’s Karate Club com diferentes ruídos.	94
Figura 28 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Zachary’s Karate Club com diferentes ruídos.	94
Figura 29 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Zachary’s Karate Club com diferentes ruídos.	95
Figura 30 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Zachary’s Karate Club com diferentes ruídos.	95
Figura 31 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e K-Means sobre Zachary’s Karate Club com diferentes ruídos.	96
Figura 32 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e Affinity Propagation sobre Zachary’s Karate Club com diferentes ruídos.	96
Figura 33 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e HDBSCAN sobre Zachary’s Karate Club com diferentes ruídos.	97
Figura 34 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Dolphins com diferentes ruídos.	97
Figura 35 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Dolphins com diferentes ruídos.	98

Figura 36 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Dolphins com diferentes ruídos.	98
Figura 37 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Dolphins com diferentes ruídos.	99
Figura 38 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Dolphins com diferentes ruídos.	99
Figura 39 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Dolphins com diferentes ruídos.	100
Figura 40 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Dolphins com diferentes ruídos.	100
Figura 41 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Dolphins com diferentes ruídos.	101
Figura 42 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Dolphins com diferentes ruídos.	101
Figura 43 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Dolphins com diferentes ruídos.	102
Figura 44 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Dolphins com diferentes ruídos.	102
Figura 45 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Dolphins com diferentes ruídos.	103
Figura 46 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e K-Means sobre Dolphins com diferentes ruídos.	103
Figura 47 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e Affinity Propagation sobre Dolphins com diferentes ruídos.	104
Figura 48 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e HDBSCAN sobre Dolphins com diferentes ruídos.	104
Figura 49 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Football com diferentes ruídos.	105
Figura 50 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Football com diferentes ruídos.	105
Figura 51 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Football com diferentes ruídos.	106
Figura 52 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Football com diferentes ruídos.	106

Figura 53 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Football com diferentes ruídos.	107
Figura 54 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Football com diferentes ruídos. .	107
Figura 55 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Football com diferentes ruídos.	108
Figura 56 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Football com diferentes ruídos.	108
Figura 57 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Football com diferentes ruídos.	109
Figura 58 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Football com diferentes ruídos. . .	109
Figura 59 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Football com diferentes ruídos.	110
Figura 60 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Football com diferentes ruídos. .	110
Figura 61 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e K-Means sobre Football com diferentes ruídos.	111
Figura 62 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e Affinity Propagation sobre Football com diferentes ruídos.	111
Figura 63 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e HDBSCAN sobre Football com diferentes ruídos.	112
Figura 64 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Political Blogs com diferentes ruídos.	112
Figura 65 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Political Blogs com diferentes ruídos.	113
Figura 66 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Political Blogs com diferentes ruídos.	113
Figura 67 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Political Blogs com diferentes ruídos.	114

Figura 68 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Political Blogs com diferentes ruídos.	114
Figura 69 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Political Blogs com diferentes ruídos.	115
Figura 70 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Political Blogs com diferentes ruídos.	115
Figura 71 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Political Blogs com diferentes ruídos.	116
Figura 72 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Political Blogs com diferentes ruídos.	116
Figura 73 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Political Blogs com diferentes ruídos.	117
Figura 74 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Political Blogs com diferentes ruídos.	117
Figura 75 – <i>Heatmaps</i> de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Political Blogs com diferentes ruídos.	118

Lista de tabelas

Tabela 1	– Pontos mais relevantes tratados na literatura	53
Tabela 2	– Descrição das bases de dados.	56
Tabela 3	– Tamanho dos espaços de busca de cada par de modelos.	59
Tabela 4	– Melhores e piores pares de modelos para cada base de dados.	67
Tabela 5	– Valores considerados para cada hiperparâmetro dos modelos de <i>embedding</i>	89
Tabela 6	– Valores considerados para cada hiperparâmetro dos modelos de agrupamento.	90

Lista de abreviaturas e siglas

BFS	Busca em Largura
CBOW	Continuous Bag of Words
CNN	Rede Neural Convolucional
DFS	Busca em Profundidade
DL	Aprendizado Profundo
GCN	Graph Convolutional Network
IoT	Internet das Coisas
LINE	Large-scale Information Network Embedding
ML	Aprendizado de Máquina
NMI	Normalized Mutual Information
RGB	Red, Green, Blue
SDNE	Structural Deep Network Embedding
TF-IDF	Term Frequency–Inverse Document Frequency
VGAE	Variational Graph Auto-Encoder

Sumário

1	INTRODUÇÃO	27
1.1	Objetivo	28
1.2	Objetivo específico	28
1.3	Organização do trabalho	29
2	FUNDAMENTAÇÃO TEÓRICA	31
2.1	Grafos	31
2.1.1	Definições	31
2.1.2	Representação do grafo	33
2.1.2.1	Matriz de adjacência	33
2.1.2.2	Lista de adjacência	34
2.2	Inteligência Artificial	35
2.2.1	Aprendizado de Máquina	35
2.2.2	Aprendizado Profundo	36
2.3	Algoritmos de agrupamento	37
2.3.1	K-Means	37
2.3.2	Affinity Propagation	38
2.3.3	DBSCAN	38
2.3.4	HDBSCAN	39
2.3.5	Métricas para agrupamento	39
2.4	Otimização de hiperparâmetros	40
2.5	Embedding	40
2.6	Embedding de grafo	44
2.6.1	DeepWalk	44
2.6.2	LINE	45
2.6.3	node2vec	46
2.6.4	SDNE	47
2.6.5	struc2vec	48
3	REVISÃO DA LITERATURA	51
4	METODOLOGIA	55
4.1	Descrição geral	55
4.2	Ambiente de execução	55
4.3	Bases de dados	55
4.4	Otimização de hiperparâmetros	56

4.5	Teste sobre dados ruidosos	58
4.5.1	Ruído negativo	60
4.5.2	Ruído positivo	61
5	ANÁLISE E DISCUSSÃO DOS RESULTADOS	63
5.1	Busca de hiperparâmetros	63
5.1.1	Zachary's Karate Club	63
5.1.2	Dolphins	65
5.1.3	Football	65
5.1.4	Political Blogs	66
5.1.5	Observações	66
5.2	Teste de grafos ruidosos	67
5.2.1	Zachary's Karate Club	68
5.2.1.1	DeepWalk	68
5.2.1.2	LINE	70
5.2.1.3	node2vec	70
5.2.1.4	SDNE	70
5.2.1.5	struc2vec	70
5.2.2	Dolphins	70
5.2.2.1	DeepWalk	71
5.2.2.2	LINE	71
5.2.2.3	node2vec	71
5.2.2.4	SDNE	71
5.2.2.5	struc2vec	71
5.2.3	Football	72
5.2.3.1	DeepWalk	72
5.2.3.2	LINE	72
5.2.3.3	node2vec	72
5.2.3.4	SDNE	72
5.2.3.5	struc2vec	73
5.2.4	Political Blogs	73
5.2.4.1	DeepWalk	73
5.2.4.2	LINE	73
5.2.4.3	node2vec	74
5.2.4.4	SDNE	74
5.2.4.5	struc2vec	74
5.2.5	Resultados relevantes	74
6	CONCLUSÃO	77
6.1	Limitações	78

6.2	Trabalhos futuros	78
	REFERÊNCIAS	81
	APÊNDICES	87
	APÊNDICE A – ESPAÇOS DE BUSCA	89
	APÊNDICE B – MAPAS DE CALOR	91
B.1	Zachary's Karate Club	91
B.1.1	LINE	91
B.1.2	node2vec	93
B.1.3	SDNE	94
B.1.4	struc2vec	96
B.2	Dolphins	97
B.2.1	DeepWalk	97
B.2.2	LINE	99
B.2.3	node2vec	100
B.2.4	SDNE	102
B.2.5	struc2vec	103
B.3	Football	105
B.3.1	DeepWalk	105
B.3.2	LINE	106
B.3.3	node2vec	108
B.3.4	SDNE	109
B.3.5	struc2vec	111
B.4	Political Blogs	112
B.4.1	DeepWalk	112
B.4.2	LINE	114
B.4.3	node2vec	115
B.4.4	SDNE	117

1 Introdução

A informação pode ser armazenada em diversos formatos (GÉCZY, 2014), cada um com suas especificidades, vantagens e desvantagens. Um dos exemplos mais tradicionais de estruturação de dados são as tabelas, que organizam informação de forma clara e bem definida, permitindo validar os dados que serão armazenados e fornecendo previsibilidade sobre sua informação (SHWARTZ-ZIV; ARMON, 2022). Entretanto, são limitadas em relação à sua capacidade de representação, já que a mesma estrutura que se apresenta como seu ponto forte acaba limitando quais tipos de informação podem ser representados, visto que nem todo dado é capaz de ser organizado em linhas e colunas. Esse tipo de informação faz parte do conjunto de dados estruturados, devido à sua estruturação e organização rígidas.

Em contrapartida, existem diversas outras estruturas de dados mais complexas, as quais permitem representar tipos de dados específicos que não se enquadram no formato estabelecido por uma tabela, como em linguagem natural, sonora, visual ou, ainda, uma agregação de múltiplos tipos. Sendo considerados dados não estruturados (EBERENDU et al., 2016), já que não possuem um esquema pré-definido que dita seu modelo.

Nesses casos, o objetivo principal é guardar a informação de forma eficiente para que não desperdice espaço de armazenamento (SIDDIQA; KARIM; GANI, 2017), além de permitir sua decodificação em um formato que possa ser interpretado por programas que, por sua vez, fazem sua reprodução para que pessoas possam interpretá-los.

Entretanto, quando se começa a analisar esse cenário de dados, considerando possíveis situações para a aplicação de tarefas na área de Inteligência Artificial, um problema logo pode ser observado: a grande maioria dos dados gerados atualmente é não estruturada, e boa parte dos algoritmos dessa área foi desenvolvida pensando no trabalho sobre dados estruturados (SINGH; HOODA, 2023).

O armazenamento e a representação da informação não estavam sendo feitos tendo essa preocupação de interpretabilidade como um fator; os computadores precisavam apenas ser capazes de guardar, apresentar, e manter e verificar a coerência dos dados. Embora, a ideia de representar informações complexas, como textos, de forma numérica não seja nova (II; BRADLEY; HANDEL, 1996; SWANSON; KOBAYASHI; TEWFIK, 1998), seus conceitos ficaram restritos ao campo da matemática, sendo teorias que não apresentavam valor prático até recentemente, quando o advento de computadores cada vez mais potentes possibilitou que modelos de redes neurais também comessem a sair do campo teórico para serem implementados e utilizados na prática.

Assim, há duas abordagens para trabalhar com esses dados complexos: algoritmos

específicos para esse tipo de informação ou algoritmos que transformam essas informações no formato esperado pela maioria dos outros algoritmos, basicamente, transformando dados não estruturados em dados estruturados. Esse processo é conhecido como *embedding*, que realiza a transformação de informações complexas e de alta dimensionalidade em representações vetoriais em um espaço latente de menor dimensionalidade. Além de estender a área de aplicabilidade de diversos algoritmos, a geração de *embeddings* permite agregar informações nas representações e reduzir a dimensionalidade dos dados.

Dentre essas estruturas complexas está o grafo (ANGLES; GUTIERREZ, 2008), o qual permite a representação de informação e os diferentes relacionamentos e conexões entre eles. Embora, outras abordagens também possam representar relações, nos grafos isso é algo fundamental, sendo tão importante quanto os próprios dados ou ainda mais, dependendo da aplicação.

Por isso, grafos também precisam passar pelo processo de *embedding* (CAI; ZHENG; CHANG, 2018) para que possam ser utilizados pelos algoritmos clássicos dentro do ramo de Inteligência Artificial, como aqueles relacionados à tarefa de detecção de comunidades, os quais fazem parte do subcampo de Aprendizado de Máquina. Este trabalho busca analisar o desempenho dessa tarefa quando são utilizados *embeddings* para o agrupamento de nós de um grafo.

Ademais, também é realizado um estudo comparativo em relação à presença de ruído sobre os dados, visto que dados reais apresentam informações irrelevantes, corrompidas ou errôneas que prejudicam a análise das informações reais que estão sendo buscadas. Assim, no contexto de grafos, ruídos podem ser arestas extras que não refletem sua estrutura real ou arestas ausentes que não foram identificadas, mas deveriam existir.

1.1 Objetivo

O presente trabalho tem como objetivo geral analisar o comportamento de diferentes modelos de *embedding* e de agrupamento quando treinados sobre bases de dados ruidosos, levando em consideração um conjunto de grafos e modelos, além de diferentes níveis de ruído. Isto é, deseja-se avaliar os efeitos que o ruído em grafos pode ter sobre os agrupamentos definidos a partir dos *embeddings* gerados sobre esses dados.

1.2 Objetivo específico

Os objetivos específicos deste trabalho consistem em sub-metas dentro do objetivo geral que precisam ser satisfeitas para que este seja atingido. Dessa forma, podem-se definir dois grandes objetivos específicos: definição de hiperparâmetros para os modelos de *embedding* e de agrupamento; e teste de robustez em relação a ruído sobre grafos.

1. **Busca de hiperparâmetros:** Inicialmente, foi realizado um *Grid Search* para determinar os melhores parâmetros para os modelos que estão sendo avaliados. Esse processo foi repetido para cada combinação de base de dados, modelo de *embedding* e modelo de agrupamento considerado no trabalho.
2. **Teste de robustez sobre ruído:** Posteriormente, através de modelos configurados com os hiperparâmetros encontrados na etapa anterior, foram gerados *embeddings* e agrupamentos dos grafos considerando diferentes níveis de ruído sobre os dados.

Vale destacar que foram utilizados grafos pertencentes a diferentes campos de estudo a fim de fazer uma análise mais geral sobre os grafos, sem que as especificidades de uma determinada aplicação enviesassem os resultados.

1.3 Organização do trabalho

O conteúdo do presente trabalho realiza uma fundamentação teórica, incluindo os modelos tanto de *embedding* quanto de agrupamento sendo avaliados, no Capítulo 2, dando uma ideia geral sobre a motivação por trás dos *embeddings*. Em seguida, discute-se alguns trabalhos relacionados ao tema no Capítulo 3, que também avaliam métodos de *embedding* de grafo. Também são descritas as bases de dados e a metodologia utilizada no Capítulo 4. Por fim, tem-se a apresentação dos resultados e sua análise no Capítulo 5, e a conclusão desse trabalho no Capítulo 6.

2 Fundamentação Teórica

Este capítulo inclui os principais conceitos utilizados neste trabalho, apresentando de forma resumida os conhecimentos sendo aplicados durante o desenvolvimento do presente estudo.

Inicialmente, descrevem-se as áreas de estudo das quais os modelos e algoritmos sendo avaliados fazem parte; em seguida, o conceito e funcionamento dessas estratégias são descritos, e, por fim, descrevem-se os conceitos usados para aplicar os métodos e avaliar seus resultados.

2.1 Grafos

Informação pode ser armazenada e representada das mais variadas formas, cada qual com objetivos específicos. Por exemplo, uma tabela permite organizar e estruturar dados de forma clara e definida através de linhas e colunas; essas características permitem que consultas sejam feitas de forma fácil e previsível. Por isso, são utilizadas em bases de dados estruturadas, que definem estruturas rígidas com regras para manter a consistência e coerência de seus dados.

Embora esse tipo de armazenamento de dados seja muito útil para informações estruturadas, ou seja, que possam ser organizadas no formato tabular, há diversas situações em que esse não é o caso. Na realidade, a grande maioria das informações geradas atualmente não se encaixa nessa definição. Imagem, áudio e vídeo também possuem regras que definem como sua informação deve ser organizada, mas são mais complexas que uma simples tabela, cada qual tendo características próprias para representar um tipo específico de informação. Por isso, geralmente, estão presentes em bases de dados não estruturadas, que são flexíveis para atender a qualquer tipo de dado e possuem capacidade de armazenar grandes quantidades de dados.

Um caso interessante de estrutura de dados que pode ser organizado tanto como uma tabela, quanto como uma estrutura mais complexa, é o grafo ([PFALTZ, 1972](#)), como é apresentado na Subseção [2.1.2](#).

2.1.1 Definições

A seguir, são apresentadas algumas definições da área de Teoria de Grafos ([BONDY; MURTY et al., 1976](#)), o que inclui algumas classificações atribuídas aos grafos de acordo com as características de seus componentes e de sua estrutura.

- **Grafo:** um grafo consiste de um conjunto de vértices (ou nós) V e um conjunto de arestas E , podendo ser representado através da Equação 2.1.

$$G = (V, E) \quad (2.1)$$

- Se uma aresta se conecta a um vértice, a aresta é **incidente** no vértice, e vice-versa; além disso, o vértice é uma **extremidade** da aresta.
- **Laço:** aresta que possui apenas uma extremidade.
- **Múltiplas arestas (ou arestas paralelas):** duas ou mais arestas que possuem as mesmas extremidades.
- **Vértices adjacentes:** dois vértices que estão conectados por uma aresta.
- **Passeio:** sequência alternada de vértices e arestas $v_1e_1v_2e_2\ldots v_ne_nv_{n+1}$ com $n \geq 0$. Se $v_1 = v_{n+1}$, então o passeio é **fechado**, caso contrário, é **aberto**. O **comprimento** do passeio corresponde ao número de arestas nele. Um caminho com comprimento zero é chamado de **passeio trivial**.
- **Trilha:** passeio sem repetição de arestas.
- **Caminho:** passeio sem repetição de vértices e, consequentemente, sem repetição de arestas.
- **Circuito:** trilha fechada.
- **Alcançabilidade:** capacidade de ir de um vértice a outro dentro de um grafo. Dado um par de vértices u e v , v é considerado alcançável a partir de u caso haja uma sequência de vértices (passeio) que comece em u e termine em v .
- **Grafo simples:** um grafo sem laços ou arestas paralelas. As arestas em um grafo simples podem ser representadas como conjuntos $\{v_i, v_j\}$ dos dois vértices que elas tornam adjacentes. Um grafo com mais de uma aresta entre um par de vértices é chamado de **multigrafo**, enquanto um grafo com laço é chamado de **pseudografo**.
- **Direcionado:** um grafo em que as arestas podem ser percorridas apenas em uma direção. Arestas em um grafo direcionado simples podem ser representadas por pares ordenados (v_i, v_j) dos dois vértices que elas conectam. Nesse caso, também, v_i é adjacente a v_j e vice-versa.
- **Grau:** o grau de um vértice corresponde ao número de arestas incidentes nele, sendo representado como $\deg(v)$ para um dado vértice $v \in V$.

- **Grafo conectado:** um grafo que possui um passeio entre todos os pares de vértices distintos.
- **Subgrafo:** um grafo H é um subgrafo de um grafo G caso todos os vértices e arestas de H também estejam em G , i.e., dado $H = (V_H, E_H)$ e $G = (V_G, E_G)$, se $V_H \subset V_G$ e $E_H \subset E_G$, então H é um subgrafo de G .
- **Componente conectado:** um componente conectado de G é um subgrafo conectado H de G de tal forma que nenhum outro subgrafo de G contenha H , i.e., H é um subgrafo apenas de G .
- **Grafo ponderado:** um grafo $G = (V, E)$ junto com uma função $w : E \rightarrow \mathbb{R}$ que associa um peso numérico a cada uma das arestas.
- **Grafo completo:** um grafo completo com n nós, representado como K_n , é o grafo simples com nós $\{1, \dots, n\}$ e uma aresta para todos os pares de nós distintos.
- **Grafo planar:** um grafo conectado simples que pode ser representado em um plano sem que nenhuma de suas arestas se cruzem. Esse desenho é denominado como o *embedding* do grafo em um plano.
- **Densidade:** medida que indica a proporção de arestas no grafo em relação à quantidade máxima de arestas possível. A densidade de um grafo é definida pela Equação 2.2 e varia entre 0 e 1 ($D \in [0, 1]$).

$$D = \frac{|E|}{E_{max}} \quad (2.2)$$

- **Grafo denso:** um grafo com densidade próxima de 1; i.e., sua quantidade de arestas é próxima do máximo possível.
- **Grafo esparso:** um grafo com densidade longe de 1; i.e., sua quantidade de arestas é significativamente menor que o máximo possível.

2.1.2 Representação do grafo

Um grafo pode ser representado de múltiplas formas (AHO; HOPCROFT, 1974), dentre elas estão a matriz de adjacência e a lista de adjacência, que são apresentadas a seguir.

2.1.2.1 Matriz de adjacência

Uma forma de representar um grafo é a matriz de adjacência. Essa matriz possui dimensões $|V| \times |V|$, tendo uma linha e uma coluna para cada um de seus vértices, e cada elemento $e_{u,v}$ representa uma aresta entre u e v no grafo.

Algumas definições para certos tipos de grafos são: em grafos não direcionados, a matriz é simétrica, ou seja, $e_{i,j} = e_{j,i}, \forall i, j \in V$; em grafos não ponderados, $e_{i,j} \in (0, 1), \forall i, j \in V$.

Essa representação é simples e fácil de entender e organizar, mas também tem seus pontos fracos. Como possui um elemento para cada possível aresta do vértice, para grafos esparsos, a maior parte da matriz será composta por zeros, e para grafos muito grandes, seu tamanho acaba se tornando um problema, já que é quadrático em relação ao seu número de vértices $|V|^2$.

2.1.2.2 Lista de adjacência

Outra forma de organizar um grafo é a partir da lista de adjacência, uma lista de listas que é indexada pelos vértices. Assim, cada vértice u possui uma lista associada a ele que indica para quais vértices possui uma conexão direta; ou seja, a lista de adjacência é formada por diversas listas de vizinhança dos nós.

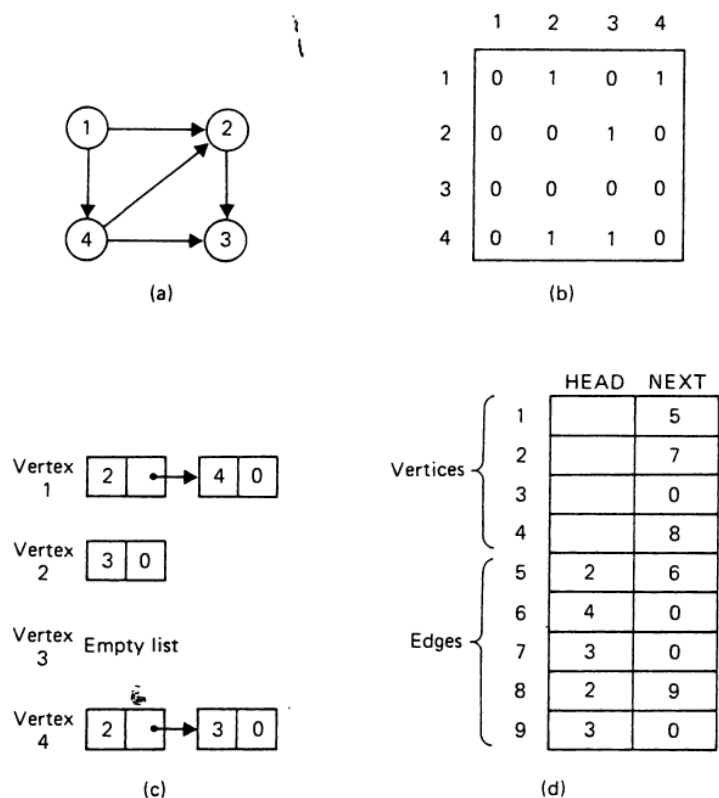
Assim, dado um par de vértices u e v , para descobrir se u é vizinho direto de v , é necessário percorrer a lista associada a u e verificar se v está presente nela.

Como armazena apenas as informações das arestas existentes no grafo, seu custo de memória é $|V| + |E|$, já que cada nó tem uma lista associada a ele com os nós de sua vizinhança, sendo muito eficiente para grafos muito grandes.

Entretanto, diferente da matriz de adjacência, em que o tempo para identificar a existência de uma conexão entre um par de vértices é $O(1)$, na lista de adjacência, passa a ser $O(d(u))$, sendo d o grau do vértice u , ou seja, a quantidade de vizinhos diretos de u . Isso se mostra um problema em situações em que se deseja saber se dois nós possuem um caminho que os conecte, já que seria necessário percorrer diversas listas para determinar isso. Outra situação que desfavorece essa busca, é em grafos densos, já que a lista de cada nó seria mais longa.

Na Figura 1, é apresentado um grafo direcionado em (a) com sua representação por meio de uma matriz de adjacência em (b) e através de uma lista de adjacência em (c) e (d), esta última organizada no formato tabular.

Figura 1 – Um grafo direcionado e suas representações.



Fonte: Adaptado de (AHO; HOPCROFT, 1974).

2.2 Inteligência Artificial

A inteligência artificial é o campo da computação voltado a estudar formas de reproduzir o comportamento humano através de sistemas que tentam simular como um ser humano resolveria certos problemas (KINGMA; BA, 2014). Há diversos caminhos para alcançar esse objetivo, sendo um deles o Aprendizado de Máquina.

2.2.1 Aprendizado de Máquina

O aprendizado de máquina (DOMINGOS, 2012; VAPNIK, 2013) busca alcançar o objetivo mencionado anteriormente através de sistemas que sejam capazes de detectar e aprender padrões sem a intervenção direta de uma pessoa, ou seja, não precisam receber instruções específicas sobre as tarefas designadas a eles.

Um exemplo de trabalho nesse contexto é o Aprendizado de Máquina Supervisionado (KOTSIANTIS et al., 2007), no qual um programa, comumente chamado de modelo, recebe uma quantidade significativa de dados rotulados e deve prever os rótulos de novos objetos que não possuem essa informação. Para deixar isso mais claro, pode-se considerar uma situação em que se deseja ter uma previsão dos ganhos de uma filial de uma rede

de supermercados. A partir de um relatório de vendas com informações diárias desse estabelecimento, é possível treinar um modelo com base em suas informações para prever os possíveis valores de ganho para dias futuros.

Outro tipo de tarefa é o Aprendizado de Máquina Não Supervisionado ([GHAHRAMANI, 2003](#)), no qual, a partir de um conjunto de dados, o modelo deve identificar padrões, estruturas e relacionamentos entre os dados. Nesse caso, não há a presença de um rótulo ou variável como objetivo, tendo apenas os próprios dados como entrada do modelo.

Além disso, diferente do aprendizado supervisionado, que possui um objetivo bem definido (um rótulo, por exemplo), no aprendizado não supervisionado, geralmente, não há uma única resposta que pode ser verificada e avaliada, o que exige uma análise posterior de seus resultados para entender seu significado.

Dentro do aprendizado não supervisionado, uma das principais tarefas é o agrupamento ([XU; WUNSCH, 2005](#)), no qual um modelo deve identificar grupos dentro de um conjunto de dados de entrada com base na semelhança ou proximidade entre eles. A saída resultante consiste em grupos de objetos que foram classificados como semelhantes entre si e diferentes dos demais.

2.2.2 Aprendizado Profundo

O aprendizado profundo ([LECUN; BENGIO; HINTON, 2015](#); [GOODFELLOW, 2016](#)) é um campo dentro do aprendizado de máquina, o qual busca simular o pensamento humano por meio de redes neurais artificiais, sistemas desenvolvidos inspirados no funcionamento e na estrutura do cérebro humano.

O componente básico de uma rede neural é o neurônio artificial; seu funcionamento segue: 1. recebimento de entradas e seus respectivos pesos; 2. soma ponderada das informações recebidas e agregação de viés (constante que flexibiliza sua ativação); e 3. função de ativação, o valor resultante anterior é passado por uma função não linear e a saída ativa o neurônio de acordo com seu valor. O esquema a seguir ilustra o funcionamento de um neurônio simples.

O resultado da agregação das entradas do neurônio pode ser calculado de acordo com a Equação 2.3.

$$z = \sum_i (x_i \times w_i) + b \quad (2.3)$$

Já a saída depende da função de ativação escolhida ([NWANKPA et al., 2018](#)); há diversas opções como Sigmoid, ReLu e Tangente hiperbólica. A Equação 2.4 mostra alguns exemplos de funções de ativação.

$$f(z) = \begin{cases} \frac{1}{1+\exp -x}, & \text{se } f = \text{Sigmoid} \\ \max(0, z), & \text{se } f = \text{ReLU} \\ \frac{\exp x - \exp -x}{\exp x + \exp -x}, & \text{se } f = \text{Tanh} \\ \dots & \end{cases} \quad (2.4)$$

Embora possa ter várias entradas, um neurônio é capaz de gerar apenas um único valor de saída, o que não é muito simples para a maioria das situações. Por isso, geralmente, são organizados em camadas, onde vários neurônios recebem a mesma entrada X_i e possuem peso W_i com valores distintos w_i para cada par entrada-neurônio. Com isso, é possível gerar uma saída de tamanho igual à quantidade de neurônios da camada.

Uma rede neural é composta por uma sequência de camadas de neurônios, entretanto, as especificidades de sua arquitetura variam conforme sua implementação, sendo flexíveis quanto à quantidade de camadas e de neurônios por camada. A última camada é responsável por calcular a previsão final da rede, por isso, a função de ativação associada a ela deve ser selecionada de acordo com a aplicação em que está inserida, por exemplo, para tarefas de classificação pode-se utilizar a função SoftMax (BRIDLE, 1989), que gera saídas entre 0 e 1 que quando somadas resultam em 1, podendo ser interpretadas como as probabilidades para cada rótulo.

2.3 Algoritmos de agrupamento

Há diversos modelos de agrupamento, cada qual se baseando em uma métrica, utilizando estratégias próprias e tendo hiperparâmetros específicos. Para este trabalho, foram escolhidos três modelos de agrupamento para serem aplicados sobre os *embeddings* gerados dos grafos: K-Means, Affinity Propagation e HDBSCAN, cada um com características próprias, vantagens e desvantagens.

2.3.1 K-Means

Modelo clássico para a tarefa de aprendizado não supervisionado (MACQUEEN, 1967; HARTIGAN; WONG, 1979), os dados são separados em grupos de acordo com sua proximidade a um centróide, um dado que representa o centro de um grupo. Seu algoritmo itera sobre todos os dados, atribuindo um grupo de acordo sua distância em relação aos centróides e atualizando os centróides para representar os novos centros dos grupos que tiveram novos dados inseridos. Essa abordagem exige a definição do parâmetro k antes de sua execução, o qual define a quantidade de grupos que serão gerados, mas que acaba sendo um limitante, visto que, na maioria dos casos, essa informação é desconhecida.

2.3.2 Affinity Propagation

Semelhante ao K-Means, o algoritmo Affinity Propagation ([FREY; DUECK, 2007](#)) define dados como representantes dos grupos que servem como referência para sua separação; entretanto, não exige a definição da quantidade de grupos que serão gerados. De forma simplificada, seu fluxo é: troca de mensagens entre os dados, definição de exemplares e atribuição dos grupos.

Nesse contexto, exemplares de *Affinity Propagation* são análogos aos centróides de *K-Means*, já que ambos são representantes de grupos.

Inicialmente, todos os dados trocam mensagens entre si, atualizando responsabilidade e disponibilidade entre os pares de dados até que se atinja uma condição de parada. Em seguida, todos os candidatos a serem exemplares são analisados e os exemplares são definidos; por fim, os dados restantes (que não são exemplares) são atribuídos ao grupo do exemplar com maior similaridade em relação a eles.

2.3.3 DBSCAN

Os modelos anteriores fazem parte do ramo de algoritmos paramétricos, ou seja, que assumem a existência de uma distribuição de probabilidade nos dados; por exemplo, K-Means assume que os dados possuem uma distribuição Gaussiana, sendo muito útil em casos onde esse tipo de informação está disponível. Entretanto, frequentemente, não se tem esse tipo de conhecimento, o que exige uma análise mais aprofundada dos dados; caso contrário, os resultados gerados podem ter uma menor qualidade.

O DBSCAN ([ESTER et al., 1996](#)) é um modelo não paramétrico, ou seja, que não assume qualquer distribuição sobre os dados, baseado em densidade. Assim, dados mais próximos uns aos outros são agrupados juntos; nesse caso, a definição de densidade é baseada em dois parâmetros: ϵ (epsilon) e número mínimo de amostras.

A construção dos grupos começa analisando todos os dados e definindo pontos centrais, pontos que têm pelo menos a quantidade mínima de amostras definidas até uma distância ϵ deles; o restante que não foi classificado como central é considerado um ponto de borda. Em seguida, começa o processo iterativo para atribuição dos grupos. Um ponto central é selecionado e atribuído a um grupo; então os pontos centrais a uma distância ϵ dele são adicionados a esse grupo, repetindo-se até não haja mais pontos centrais para serem adicionados. Depois disso, todos os pontos de borda a uma distância ϵ dos pontos centrais anteriores são adicionados ao grupo; assim, o primeiro grupo é definido. Esse processo é repetido com os dados restantes até que todos os pontos centrais façam parte de algum grupo. Os pontos de borda que não são atribuídos a nenhum grupo são considerados como *outliers*.

2.3.4 HDBSCAN

O HDBSCAN ([CAMPELLO et al., 2015](#)) é baseado no DBSCAN, adicionando a ideia de hierarquia em seus conceitos para atribuir grupos sem a necessidade de definir um valor para ϵ , que é um dos desafios na aplicação do DBSCAN, visto que valores muito baixos de ϵ levam a muitos grupos com poucos pontos, enquanto valores altos levam a poucos grupos com quase todos os dados.

HDBSCAN introduz o conceito de distância de alcançabilidade mútua, definida entre dois pontos como o máximo entre a própria distância entre os pontos e as distâncias até o k -ésimo ponto mais próximo de cada um, sendo k o número mínimo de amostras. Primeiramente, calculam-se as distâncias de alcançabilidade mútua entre todos os dados, que, por sua vez, formam um dendograma, uma árvore de densidade que separa os dados com base na distância mínima entre eles. Daquela árvore, pode-se obter agrupamentos definindo um valor de distância mínima e cortando a árvore nesse nível, tendo cada ramo cortado como um grupo e seus pontos como membros, que é equivalente à aplicação do DBSCAN com ϵ igual à distância escolhida; assim, pode-se obter os grupos resultantes do DBSCAN para qualquer valor de ϵ a partir dessa árvore.

O agrupamento final é definido com base na longevidade dos ramos no dendograma e na quantidade de pontos neles; assim, é escolhido o nível (ϵ ou distância mínima) com ramos que não se dividiram ao longo de vários níveis e que possuam uma grande quantidade de pontos.

2.3.5 Métricas para agrupamento

A fim de avaliar a qualidade dos grupos gerados a partir dos dados passados aos modelos de agrupamento, diversas métricas foram desenvolvidas. Neste trabalho é utilizado o Coeficiente de Silhueta para isso.

Coeficiente de Silhueta

O Coeficiente de Silhueta ([ROUSSEEUW, 1987](#)) é uma das métricas utilizadas para mensurar a qualidade da separação gerada pelos modelos. Ele determina quão semelhante um objeto é em relação a seu grupo em comparação com os demais grupos. A Equação 2.5 apresenta o cálculo do índice para um dado objeto.

$$s = \frac{b - a}{\max(a, b)} \quad (2.5)$$

Sendo a a distância média da instância para todos os membros de seu grupo (mede compactação) e b a distância média da instância para todos os membros do grupo mais

próximo. Seu valor está entre -1 e 1 ($-1 \leq s \leq 1$), s próximo a -1 indica que o objeto está muito próximo a um outro grupo, s próximo a 1 indica que o grupo foi bem atribuído e s próximo a 0 indica que está na fronteira de dois grupos.

Para calcular o Coeficiente de Silhueta do agrupamento como um todo, são calculados os coeficientes individuais de cada objeto; então, toma-se sua média.

2.4 Otimização de hiperparâmetros

Os modelos de aprendizado de máquina possuem parâmetros internos que são ajustados durante seu treinamento, sendo atualizados automaticamente enquanto iteram sobre os dados de treino de acordo com a lógica definida para cada modelo. Como dito anteriormente, é essa característica que define esses tipos de algoritmos.

Entretanto, há outro tipo de parâmetro nesses modelos, os quais são definidos antes de sua aplicação e são utilizados dentro de sua lógica de atualização, sendo chamados de hiperparâmetros. Seus valores permitem um ajuste fino do modelo, sendo possível parametrizar seus cálculos e melhorar seu desempenho.

Os hiperparâmetros são específicos para cada modelo, refletindo características internas de sua implementação. Além disso, seus valores ótimos não são constantes, assim, dependem da aplicação em que são utilizados. Por isso, há diversas estratégias para obter seus valores ([LIASHCHYNSKYI; LIASHCHYNSKYI, 2019](#)).

Grid Search

Um desses algoritmos de busca de hiperparâmetros é o Grid Search, que realiza uma busca exaustiva sobre um conjunto de hiperparâmetros. Assim, é necessário definir o espaço de busca antes da execução do algoritmo, o qual consiste em um conjunto de listas, contendo uma série de valores para cada hiperparâmetro que se deseja otimizar.

Embora cubra uma vasta gama de opções, garantindo que a melhor opção dentre elas seja escolhida, devido a esse grande volume de execuções necessárias para isso, o tempo de processamento dessa abordagem é mais longo. Além disso, muitas das iterações do algoritmo podem considerar valores ruins, o que desperdiçaria tempo.

2.5 Embedding

Boa parte dos modelos de aprendizado de máquina é voltada para trabalhar com dados tabulares, i.e., organizados em forma de linhas e colunas; por isso, caso se deseje utilizar informações que não se enquadram nesse formato, é preciso manipulá-las antes que possam ser aplicadas aos modelos.

Uma das formas de se fazer isso é através da transformação dos dados em vetores numéricos. Como exemplo, tem-se a transformação de uma imagem composta por uma matriz de *pixels* em um único vetor numérico, que poderia ser feita diretamente, considerando cada pixel como uma *feature* distinta; assim, uma imagem seria convertida em um vetor de $h \times w$ *features*, sendo h a altura da imagem e w sua largura, ambos em *pixels*. Isso vale para imagens com apenas um canal de cor (como fotos em escala de cinza), mas grande parte delas possui múltiplos desses canais, um dos mais comuns sendo imagens RGB, que possuem 3 canais: vermelho, verde e azul; assim, neste caso, a representação vetorial corresponderia à concatenação de 3 vetores com $h \times w$ *features* cada. Atribuindo números a esse exemplo, uma imagem RGB com resolução de 720p (1280×720 *pixels*) seria convertida em um vetor com 2.764.800 *features*.

Em vez de manipular apenas o formato dos dados, outras abordagens realizam uma transformação sobre seus valores, utilizando-se de regras definidas para isso, como, por exemplo, contagens, frequências, estatísticas, atributos conhecidos ou funções matemáticas fixas. Um desses processos é o TF-IDF (AIZAWA, 2003), que transforma um documento composto por uma sequência ordenada de palavras em um vetor numérico. Em resumo, os valores dos vetores correspondem aos pesos das palavras de acordo com sua importância no documento e no *corpus*, i.e., cada *feature* do vetor corresponde a um termo do vocabulário e é calculada como a combinação de 2 medidas: TF mede a frequência do termo no documento, assim, quanto mais frequente, maior seu peso; e IDF mede a raridade do termo no *corpus*, assim, quanto menos frequente, maior seu peso; a ideia por trás desse cálculo é de que palavras comuns em um documento são importantes no contexto específico, mas palavras raras no *corpus* que aparecem em um documento são importantes no contexto geral.

Entretanto, é possível perceber limitações nesses tipos de abordagens apresentadas anteriormente. No exemplo da imagem, ainda que o número de *features* seja muito grande, isso não agrega informação relevante sobre ela, visto que cada valor possui apenas informação local sobre a imagem. Além disso, o uso direto dos valores dos *pixels* torna o número de *features* dependente da resolução das imagens, então imagens com diferentes resoluções acabariam tendo quantidades diferentes de *features*, sendo necessário desconsiderar parte da imagem ou preenchê-la com *pixels* extras para manter as mesmas dimensões.

Por sua vez, no TF-IDF, também há essa dependência em relação ao tamanho dos dados, nesse caso sobre o vocabulário do *corpus*; assim, aplicações com vocabulários extensos levariam a vetores muito longos. Ademais, como cada termo corresponde a uma dimensão independente do vetor, situações com sinônimos ou polissemia não são reconhecidas, assim como acontece com o contexto das palavras.

Como uma alternativa para essa manipulação de dados, o processo de *embedding*

(FROME et al., 2013) permite gerar uma representação vetorial em um espaço latente denso com menor dimensionalidade, tendo no máximo algumas centenas de valores, além de uniformizar seu tamanho. Nesse caso, essas representações são aprendidas, sendo o resultado do treinamento de algoritmos que otimizam vetores a fim de minimizar uma determinada função de perda; com isso, busca-se preservar relações existentes no espaço original dos dados, transformando objetos semelhantes em vetores próximos.

Um *embedding* pode ser representado como uma função $f : X \rightarrow \mathbb{R}^d$ que tem como objetivo que noções de proximidade ou similaridade em X sejam preservadas no espaço latente vetorial, sendo d o tamanho dos vetores resultantes, que pode ser parametrizado antes da geração dos *embeddings*. Destaca-se que as *features* neste caso não são interpretáveis individualmente, já que correspondem à combinação de diversas características dos dados; a informação acaba sendo distribuída pelo vetor.

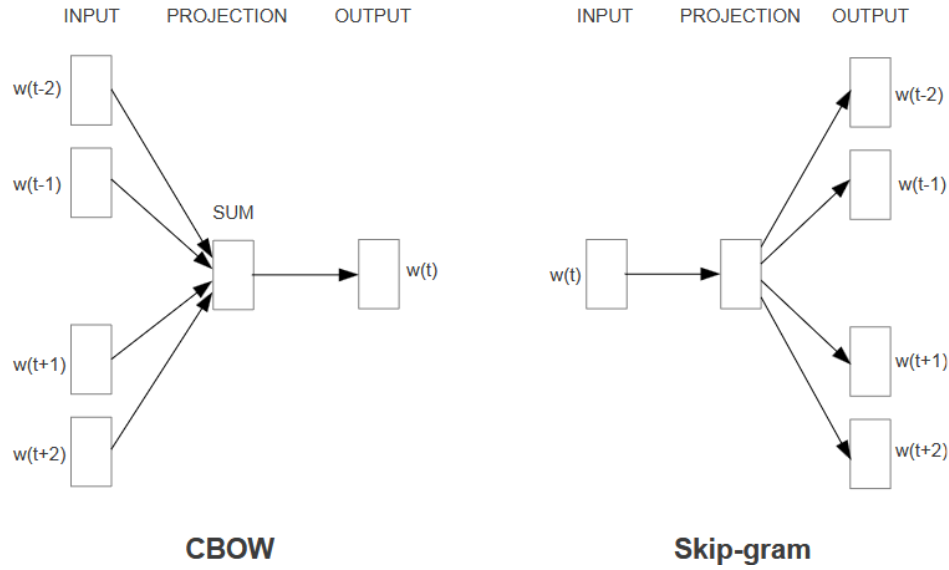
Word2Vec

O modelo Word2Vec (MIKOLOV et al., 2013a; MIKOLOV et al., 2013b) é uma rede neural rasa, i.e., possui poucas camadas ocultas, nesse caso, apenas uma, que busca gerar representações de palavras em um espaço latente. Assim, sua estrutura pode ser definida por 3 camadas: entrada e saída, com V nós cada, sendo V o tamanho do vocabulário sendo considerado, além da camada oculta de propagação com d nós, sendo d a dimensão do vetor de *embedding*. Além disso, o modelo usa *stochastic gradient descent* e *backpropagation* para o cálculo e atualização de seus parâmetros.

São propostas 2 arquiteturas para o modelo, ambas buscando adicionar contexto a fim de enriquecer as informações para seu treinamento, sendo elas *Continuous Bag-of-Words* (CBOW) e *Skip-gram*. A estrutura proposta por cada arquitetura pode ser visualizada na Figura 2, $w(t)$ corresponde à palavra atual, enquanto $w(t - 2)$, $w(t - 1)$, $w(t + 1)$ e $w(t + 2)$ são as palavras de seu contexto. No caso do CBOW, o contexto é a entrada e busca-se prever o termo atual, como se estivesse tentando adivinhar qual a palavra faltante em uma frase; por sua vez, no Skip-gram, a entrada é a palavra atual, enquanto o contexto é aquilo que se busca prever, assim, é como se tentasse descobrir em quais contextos uma dada palavra poderia estar inserida.

- **CBOW:** busca prever uma palavra (atual) com base no seu contexto, i.e., nas palavras antes e depois dela em uma sentença.
- **Skip-gram:** utiliza uma palavra (atual) para prever seu contexto, i.e., as palavras antes e depois dela em uma sentença.

Em ambos os casos, há duas matrizes envolvidas nos cálculos dos valores, W e W' , W é uma matriz de $V \times d$ que agrega as entradas enquanto W' , de $d \times V$, calcula o score

Figura 2 – Arquiteturas do Word2Vec.

Fonte: Adaptado de (PEROZZI; AL-RFOU; SKIENA, 2014).

de cada palavra.

Ao final, originalmente, havia uma camada de SoftMax (BRIDLE, 1989) que calculava as probabilidades finais de cada palavra, de acordo com a Equação 2.6, entretanto, isso é muito custoso, sendo inviável para situações com vocabulários mais extensos. Por isso, são utilizadas estratégias de otimização para aproximar seu comportamento. Por exemplo, considerando um vocabulário com 30.000 palavras e uma representação latente de 128 dimensões, o modelo teria ao todo 3.840.000 parâmetros para serem otimizados, o que pode deixar seu treinamento consideravelmente mais lento.

$$p(w_O|w_I) = \frac{\exp(v'_{w_O} \top v_{w_I})}{\sum_{w=1}^W \exp(v'_w \top v_{w_I})} \quad (2.6)$$

A otimização da última etapa do modelo consiste na aproximação do SoftMax completo por meio de uma dessas estratégias: Hierarchical SoftMax e Negative Sampling.

- **Hierarchical SoftMax:** ao invés de uma camada de SoftMax com um nó para cada saída, a camada é estruturada como uma árvore binária, assim, cada nó na camada de saída mapeia um nó não folha na árvore, enquanto os nós de saída originais correspondem às suas folhas. Essa abordagem reduz o tamanho da saída de W para $\log_2(W)$ e ainda mantém as propriedades do SoftMax, i.e., a soma das probabilidades de cada saída é 1.
- **Negative Sampling:** durante o treinamento do modelo, certos nós são desativados para reduzir a quantidade de parâmetros calculados durante sua otimização, assim,

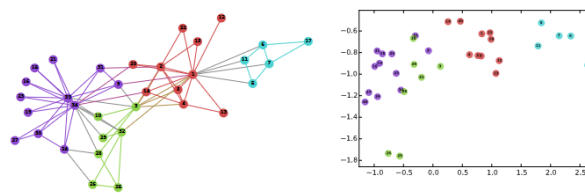
de acordo com um parâmetro *negative*, apenas essa quantidade de nós de saída é considerada, servindo como exemplos negativos junto ao nó da saída esperada.

Embora tenha sido desenvolvida para aplicações que trabalham com informações em linguagem natural, diversos métodos para *embedding* de grafo fazem uso de modelos como Word2Vec, fazendo uma analogia entre palavras em sentenças e nós em um passeio.

2.6 Embedding de grafo

Há diversas estratégias de *embedding* de grafo (CAI; ZHENG; CHANG, 2018), cada uma explorando diferentes formas de extrair informações dessas estruturas de dados. Como neste trabalho utilizamos grafos mais simples, o *embedding* é gerado baseando-se exclusivamente em suas informações estruturais, entretanto, é possível construir sua representação vetorial agregando valores como atributos de nós e pesos de arestas por exemplo, em casos mais complexos. Na Figura 3, é apresentado o grafo Zachary's Karate Club com nós coloridos de acordo com comunidades detectadas (à esquerda) e o embedding de seus nós como vetores de 2 dimensões, podendo ser representados por meio de um gráfico de dispersão (à direita). Pode-se observar que os nós pertencentes às mesmas comunidades também estão próximos no gráfico, demonstrando que as similaridades identificadas no grafo foram trazidas para sua representação vetorial.

Figura 3 – Exemplo de embedding de grafo.



Fonte: Adaptado de (PEROZZI; AL-RFOU; SKIENA, 2014).

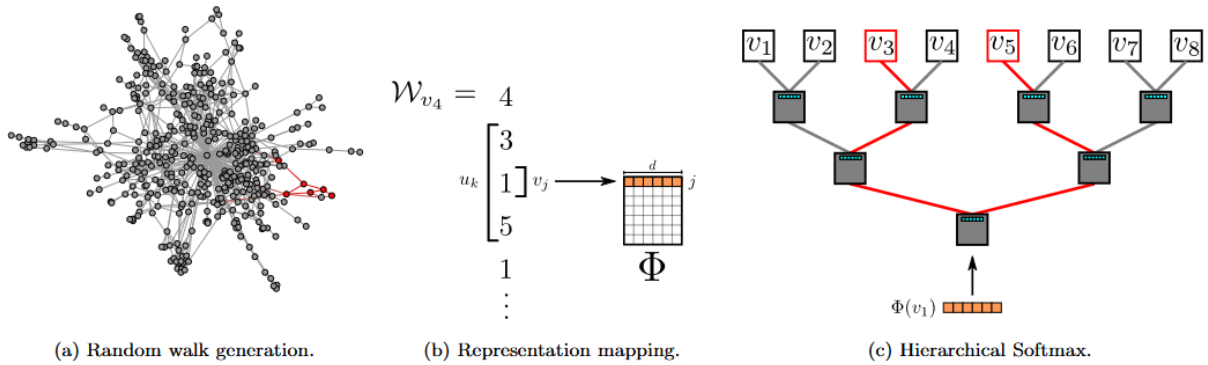
As estratégias sendo avaliadas neste trabalho incluem: DeepWalk, LINE, Node2Vec, SDNE e Struc2Vec.

2.6.1 DeepWalk

Um dos primeiros trabalhos a associar *embeddings* de grafos aos *embeddings* de texto (PEROZZI; AL-RFOU; SKIENA, 2014), fazendo uma analogia entre palavras em sentenças e nós em passeios, ele propõe o uso do mesmo modelo que mapeia sentenças textuais no contexto dos grafos. Nesse caso, é utilizado o modelo Word2Vec com a arquitetura Skip-gram para agregar informação e SoftMax Hierárquico para otimização.

Na Figura 4, é apresentada uma visão geral do processo de *embedding* usando o modelo DeepWalk. Inicialmente, são gerados n *random walks* (passeios aleatórios) de tamanho l a partir de cada um dos nós presentes no grafo em (a), então, cada passeio é processado e atualiza os *embeddings* dos nós através da arquitetura *Skip-gram* em (b). Por fim, a saída correspondente ao nó v_1 é calculada por meio do *Hierarchical SoftMax* em (c).

Figura 4 – *Overview* do DeepWalk.

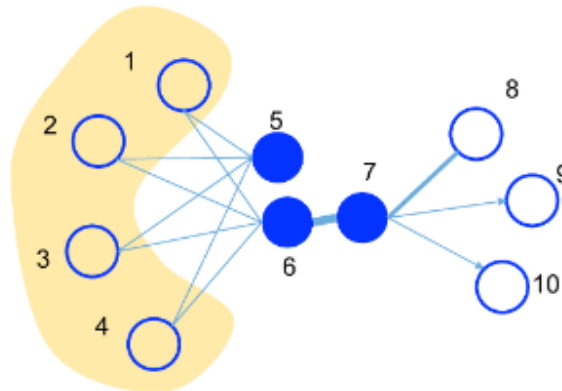


Fonte: Adaptado de (PEROZZI; AL-RFOU; SKIENA, 2014).

2.6.2 LINE

Esse trabalho (TANG et al., 2015) usa outra abordagem para realizar a transformação dos grafos, baseando-se na vizinhança de diferentes graus dos nós, mais especificamente, de 1º e 2º graus. Dessa forma, o modelo busca gerar vetores próximos para nós próximos no grafo considerando sua proximidade de 1º e 2º graus.

Figura 5 – Exemplo de grafo com destaque na vizinhança dos nós.



Fonte: Adaptado de (TANG et al., 2015).

O modelo é composto por dois componentes: LINE-1, que extrai informações da vizinhança de 1º grau, e LINE-2, que faz isso com a vizinhança de 2º grau. Com base no grafo na Figura 5, os nós 6 e 7 estariam próximos nos embeddings de LINE-1 já que, por serem terem uma aresta ligando-os diretamente, são vizinhos de 1º grau, enquanto isso, os

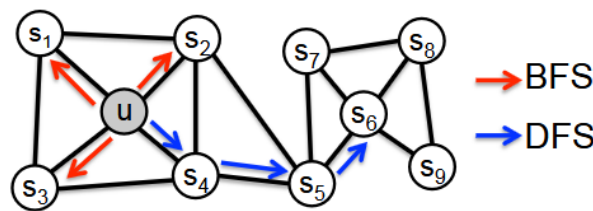
nós 5 e 6 teriam embeddings próximos em LINE-2 por compartilharem os mesmos nós na vizinhança de 2º grau. A saída final do modelo consiste na combinação dos *embeddings* de LINE-1 e LINE-2, considerando vizinhanças de 1º e 2º graus respectivamente.

2.6.3 node2vec

Partindo da ideia desenvolvida por DeepWalk, esse trabalho (GROVER; LESKOVEC, 2016) flexibiliza o conceito de vizinhança de um nó, considerando seu papel estrutural dentro do grafo e a comunidade onde está inserido. A construção dos passeios aleatórios é tratada como um problema de busca em grafo, com isso, esse processo passa a ser pensado como uma busca dentro de um espectro entre BFS e DFS, que caracterizam nós com papéis estruturais análogos e descobrem comunidades homófilas respectivamente. Na Figura 6, é possível visualizar quais nós seriam visitados na terceira iteração de cada método de busca. BFS visita todos os nós a uma distância k do nó inicial u antes de buscar qualquer nó mais distante, por isso, teria visitado os nós s_1 , s_2 e s_3 (indicados pelas setas vermelhas) que estão a uma distância 1 de u . Por sua vez, DFS teria visitado os nós s_4 , s_5 e s_6 , já que escolhe o próximo nó com base nos vizinhos do último nó, retornando para nós anteriores apenas quando o atual não possui vizinhos ou todos eles já tiverem sido visitados.

O ajuste sobre quão próximas dessas estratégias é a construção dos passeios é definido por 2 parâmetros, p e q , respectivamente. p é o parâmetro de retorno, indicando a probabilidade de visitar um nó que já foi visitado anteriormente, enquanto q é o parâmetro de entrada-saída, indicando a probabilidade de escolher nós "para fora" ou "para dentro" em relação ao nó imediatamente anterior ao atual em um *random walk*.

Figura 6 – Estratégias de busca em grafo.

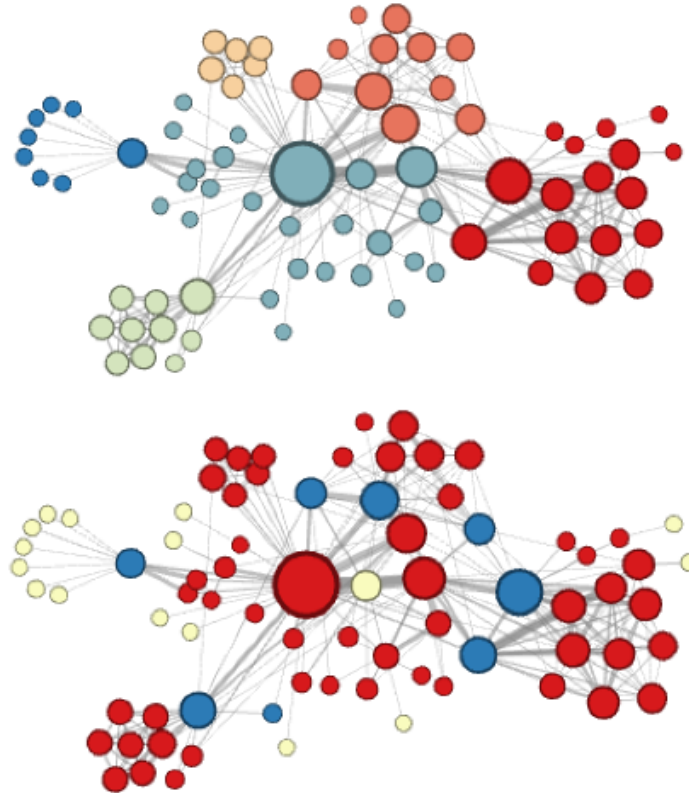


Fonte: Adaptado de (GROVER; LESKOVEC, 2016).

A Figura 7 apresenta duas representações do grafo Les Misérables (KNUTH; KNUTH, 1993), em que os personagens do romance aparecem como nós, sendo conectados caso apareçam juntos em algum momento da história. Ambos os grafos foram coloridos com base nos grupos definidos por K-Means sobre os *embeddings* dos nós gerados pelo node2vec, diferenciando-se em relação à configuração utilizada pelo modelo de *embedding*. Os grupos do grafo de cima foram gerados com o modelo configurado com $p = 1$ e $q = 0.5$,

refletindo a homofilia dos nós do grafo, enquanto que para os de baixo foram usados $p = 1$ e $q = 2$, a fim de refletir a equivalência estrutural dos nós.

Figura 7 – Comunidades definidas por node2vec com diferentes parâmetros.

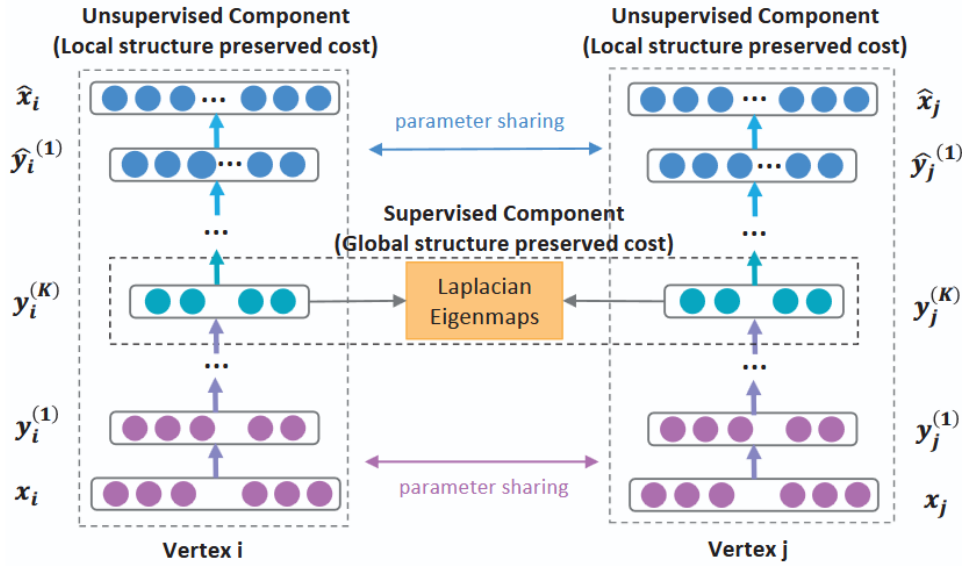


Fonte: Adaptado de (GROVER; LESKOVEC, 2016).

2.6.4 SDNE

Nesse trabalho (WANG; CUI; ZHU, 2016), é utilizado um modelo de Deep Learning composto por camadas não lineares para realizar a transformação do grafo. Como entrada são utilizadas as informações de proximidade dos nós, sendo extraídos dados sobre a vizinhança de 1º e 2º graus dos nós para recuperar informações locais e globais do grafo. A utilização desse tipo de modelo busca gerar um *embeddings* em um espaço altamente não-linear semelhante à própria estrutura do grafo. A função de perda leva em conta as proximidades de primeiro e segundo graus dos nós, sendo parametrizável sobre a parte de primeiro grau, podendo aumentar ou diminuir sua relevância e, consequentemente, afetando a relevância da parte de segundo grau.

Na Figura 8, é apresentado o *framework* do modelo semi-supervisionado de rede profunda que está por trás do SDNE.

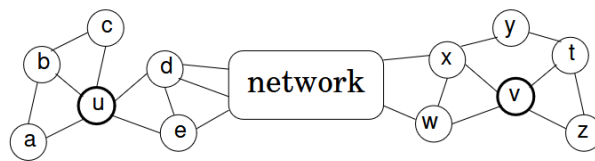
Figura 8 – *Framework* do modelo de rede neural semi-supervisionado do SDNE.

Fonte: Adaptado de (WANG; CUI; ZHU, 2016).

2.6.5 struc2vec

Diferente dos trabalhos mencionados anteriormente, esse modelo (RIBEIRO; SAVERESE; FIGUEIREDO, 2017) foca na identidade estrutural do grafo quando gera sua representação no espaço latente, i.e., os *embeddings* dos nós são gerados com base no seu papel estrutural dentro do grafo em vez de sua vizinhança. O objetivo, nesse caso, é gerar representações semelhantes para nós estruturalmente semelhantes no grafo, então, mesmo que não compartilhem a mesma vizinhança, contanto que a estrutura em volta dos nós apresente características parecidas, seus *embeddings* também serão próximos.

Na Figura 9, os nós u e v são estruturalmente semelhantes, já que possuem graus parecidos (5 e 4), estão conectados a uma quantidade semelhante de triângulos (3 e 2), além disso, estão conectados ao restante do grafo por meio de 2 nós, mesmo encontrando-se distantes dentro do grafo.

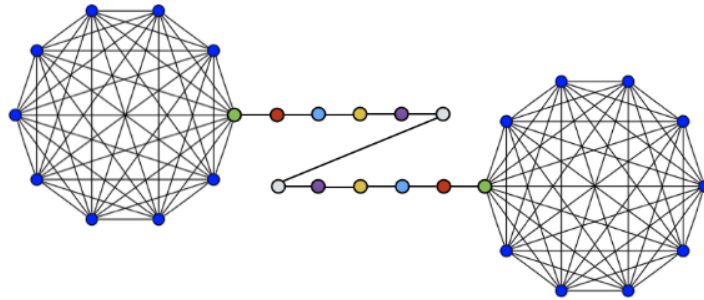
Figura 9 – Exemplo de grafo com nós estruturalmente semelhantes.

Fonte: Adaptado de (WANG; CUI; ZHU, 2016).

A Figura 10 mostra um grafo *barbell* (halteres), que consiste de dois subgrafos totalmente conectados, ligados por um subgrafo caminho linear. Quando analisado do ponto

de vista estrutural, pode-se observar a semelhança estrutural entre os nós dos subgrafos completos (em azul), além disso, os demais nós também apresentam uma semelhança estrutural, visto os nós no subgrafo caminho possuem pares com forte similaridade (demais cores).

Figura 10 – Exemplo de grafo do tipo barbell colorido de acordo com a semelhança estrutural dos nós.



Fonte: Adaptado de ([WANG; CUI; ZHU, 2016](#)).

3 Revisão da Literatura

O *embedding* de grafos é um tema bastante abordado na literatura, seja como foco do trabalho, quando um novo modelo é proposto ou analisam-se métodos já existentes, ou como uma ferramenta para auxiliar algum outro objetivo. Há diversas publicações que abordam essa temática.

Em (CAI; ZHENG; CHANG, 2018), não tem-se a proposta ou a utilização de algum método de *embedding*, ainda assim, é muito relevante para o estudo do *embedding* de grafos, sendo feito um amplo levantamento e análise de diversos algoritmos de *embeddings*, incluindo quase 100 itens, que foram classificados em relação à sua configuração, i.e., sobre o que esperam como dados de entrada e saída, e técnicas utilizadas para a geração dos vetores representativos. Ainda que não entrem a fundo em seus detalhes técnicos, a segmentação dos diferentes métodos fornece uma visão das diferentes subáreas e problemas tratados por cada um deles.

Embora não seja novo, ainda é capaz de fornecer uma boa visão geral sobre o tema da representação de grafos em espaço latente por meio de *embeddings*, além de se aprofundar um pouco nos temas que tangem cada um dos métodos e técnicas utilizadas. De acordo com as classificações descritas nele, pode-se definir que, neste trabalho, são gerados vetores em espaço latente que representam os nós de grafos homogêneos (nós e arestas de mesmo tipo em toda a estrutura), utilizando técnicas de aprendizado profundo (DeepWalk, Node2Vec, SDNE e Struc2Vec) e reconstrução de arestas baseada em otimização (LINE).

Em (TANDON et al., 2021), há um objetivo mais próximo ao do presente trabalho, já que são analisados diferentes métodos de *embedding* de grafo, seguido do agrupamento de seus resultados, entretanto, nesse caso, deseja-se comparar os grupos gerados a partir dos *embeddings* com os obtidos por meio de algoritmos que atuam diretamente sobre os grafos. Os grafos são obtidos por meio do benchmark LFR, no qual pode-se parametrizar algumas das características dos dados sendo gerados, como número de nós e grau médio dos nós. Além disso, é possível pré-definir grupos nos dados gerados, então, há um *ground truth* que possibilita o cálculo de índices de validação externa como o NMI nesse caso.

Ao final, o trabalho conclui que os resultados utilizando os métodos de *embedding* parametrizados com valores otimizados em outras tarefas apresentaram resultados muito próximos dos métodos tradicionais de detecção de comunidades em grafos. A utilização de parâmetros otimizados para essa tarefa demonstrou melhores resultados que os algoritmos padrão, entretanto, observou-se que os resultados são muito sensíveis a variações nos parâmetros considerados, além de não conseguirem definir um conjunto de parâmetros otimizados para a maioria dos casos.

Em (PANKRATZ; KAMIŃSKI; PRAŁAT, 2023), o processo de *embedding* é utilizado como uma ferramenta para otimizar algoritmos de agrupamento de nós que atuam diretamente sobre os próprios grafos, como Louvain e Leiden, a fim de reduzir a volatilidade apresentada quando o mesmo algoritmo é aplicado sobre o mesmo conjunto de dados. Isso acontece porque eles são algoritmos gulosos, i.e., funcionam escolhendo a melhor opção local, o que pode levar a decisões subótimas quando consideradas em relação ao todo, que se propagam até o resultado final e degradam sua qualidade.

Com isso, é proposta uma abordagem que gera partições iniciais dos nós a partir de agrupamentos realizados sobre seus *embeddings*, esses grupos são pequenos e servem como ponto de partida para os algoritmos mencionados anteriormente. Assim, possíveis ruídos e más escolhas iniciais são mitigados. Ao final, os resultados obtidos demonstraram que a abordagem proposta melhorou os resultados de ambos os algoritmos, com pouco impacto sobre Louvain, mas melhorias significativas e redução da volatilidade para Leiden.

Em (DEHGHAN-KOOSHKHAZI et al., 2022), é feita uma análise sobre o framework de *Divergence Score* (KAMIŃSKI; PRAŁAT; THÉBERGE, 2019), uma abordagem não supervisionada que atribui pontuação aos *embeddings* de grafos e poderia indicar a qualidade das representações geradas. O trabalho propõe o uso do *Divergence Score* como métrica para comparar diferentes estratégias de *embedding* de grafo.

Para cada método de *embedding* selecionado, foram utilizados seus parâmetros padrões para todas as suas aplicações, com exceção da dimensão dos vetores gerados. Durante os testes, os métodos foram aplicados sobre cada grafo selecionado, variando as dimensões de saída, cada teste foi repetido 50 vezes e usou a média e o desvio padrão dos scores das execuções como base para análise. No caso dos grafos artificiais, também variaram-se alguns parâmetros sobre a geração dos dados, a fim de verificar como mudanças em suas características afetariam o score resultante. Por fim, testou-se o desempenho dos métodos de *embedding* sobre tarefas de aprendizado de máquina, nesse caso, classificação de nós, identificação de comunidades e predição de *links*. Assim, pode-se observar que os vetores gerados pelos métodos que tiveram maior *Divergence Score* durante a fase de testes também tiveram melhor desempenho nessas tarefas.

Em (GUO; DAI, 2022), realiza-se a combinação das tarefas de *embedding* e de agrupamento a fim de obter vetores que sejam representativos em relação tanto aos atributos dos nós quanto à topologia de suas comunidades, assim, nós com propriedades similares seriam agrupados juntos. Para isso, a função objetivo desse modelo também leva em consideração os centros dos grupos sendo gerados, além do próprio *embedding* do grafo durante sua otimização.

Também é desenvolvido um algoritmo de autoaprendizado para otimizar a identificação de padrões durante as fases não supervisionadas. Além disso, é adequado em cenários em que grupos são grandes, tratando esses casos para que o *embedding* gerado

não seja distorcido.

Em (BHATTACHARYA; NAGWANI; TRIPATHI, 2023), é proposta uma abordagem que utiliza node2vec e GNC, assim, os processos de *embedding* e de passagem de mensagem por FFN seriam combinados por meio de GNCs voltados para a tarefa de agrupamento. Dessa forma, busca-se obter uma matriz de similaridades unificada que combina as similaridades dos nós e estruturais do grafo, o que aumentaria a acurácia e a efetividade dos algoritmos de detecção de comunidade.

O modelo proposto mostrou-se eficiente em cenários com *outliers* e em grafos esparsos, já que incorpora as médias das características das vizinhanças dos nós em conjunto com suas próprias propriedades. Também teve menor custo computacional e conseguiu detectar comunidades únicas.

Tabela 1 – Pontos mais relevantes tratados na literatura

Trabalho	Contribuição
Community detection in networks using graph embeddings (TANDON et al., 2021)	Comparação de diferentes modelos de <i>embedding</i> em dados artificiais com base no <i>benchmark</i> LFR, usando os grupos pré-definidos em sua geração como <i>ground truth</i> para avaliação das representações.
Community detection supported by node embedding (PANKRATZ; KAMIŃSKI; PRAŁAT, 2023)	<i>Embedding</i> e agrupamento dos nós são uma etapa auxiliar dos algoritmos de detecção de comunidades que atuam diretamente sobre grafos, assim, os grupos gerados são traduzidos em comunidades e servem como ponto de partida para esses algoritmos.
Evaluating node embeddings of complex networks (DEHGHAN-KOOSHKHAZI et al., 2022)	Comparação de métodos de <i>embedding</i> com base no <i>Divergence Score</i> , métrica desenvolvida pelos mesmos autores, que indica a qualidade do <i>embedding</i> .
Graph Clustering via Variational Graph Embedding (GUO; DAI, 2022)	<i>Framework</i> em que o aprendizado de <i>embedding</i> e agrupamento é feito de forma simultânea sobre dados não euclidianos, utilizando uma versão melhorada de VGAE.
A community detection model using node embedding approach and graph convolutional network with clustering technique (BHATTACHARYA; NAGWANI; TRIPATHI, 2023)	Propõem a combinação de node2vec com GCN para a geração dos <i>embeddings</i> buscando combinar as similaridades encontradas nos atributos dos nós e na topologia do grafo.

4 Metodologia

Este capítulo detalha a metodologia utilizada, descrevendo as bases de dados, algoritmos e métodos que fizeram parte do desenvolvimento do trabalho. A estrutura apresentada foi elaborada com o intuito de se adequar às especificidades das técnicas e tarefas sendo avaliadas, garantindo resultados mais consistentes e representativos.

4.1 Descrição geral

Inicialmente, foram selecionados grafos menores que tinham presença considerável na literatura. Em seguida, os métodos de *embeddings* de grafo selecionados foram adaptados para que pudessem ser aplicados como estimadores da biblioteca scikit-learn ¹. Com isso, em conjunto aos métodos de agrupamento, foi realizado um *Grid Search* em busca de bons parâmetros para os modelos em relação a cada base de dados.

Por fim, utilizando os melhores parâmetros encontrados para cada par de modelos, eles foram testados sobre os mesmos grafos com diferentes graus de ruído adicionado.

4.2 Ambiente de execução

O desenvolvimento do trabalho foi realizado por meio da plataforma Google Colab ², utilizando a linguagem de programação Python ³.

Foi escolhido o Google Colab porque ele fornece um ambiente de desenvolvimento em nuvem já configurado com bibliotecas e módulos típicos para trabalhos que envolvem ciência de dados, sendo bastante utilizado por pessoas que atuam nessa área. Além disso, as execuções são desacopladas do computador pessoal, no qual poderiam competir com outros processos rodando em paralelo.

4.3 Bases de dados

As bases de dados selecionadas são grafos que frequentemente aparecem na literatura. Embora sejam pequenos, permitem avaliar os modelos sobre diversos dados distintos.

- **Zachary's Karate Club:** um dos grafos mais tradicionais da literatura (ZACHARY, 1976), costuma ser um dos primeiros grafos vistos quando se estuda sobre o tema.

¹ <<https://scikit-learn.org/>>

² <<https://colab.google/>>

³ <<https://www.python.org/>>

Consiste em uma rede social observada por Wayne Zachary em um clube universitário de karatê nos EUA na década de 70.

- **Dolphins:** rede social de golfinhos-nariz-de-garrafa (LUSSEAU et al., 2003), cada aresta no grafo representa associações frequentes entre os golfinhos.
- **Football:** rede de jogos de futebol americano entre faculdades da Divisão IA durante a temporada regular de outono de 2000 (GIRVAN; NEWMAN, 2002), os nós apresentam valores indicando a qual conferência eles pertencem.
- **Political Blogs:** rede de blogs políticos dos Estados Unidos (ADAMIC; GLANCE, 2005), construída com base nos hiperlinks entre eles.

Na Tabela 2 pode-se visualizar algumas métricas relacionadas a cada grafo. Observa-se que todos os grafos possuem menos de mil vértices e têm densidade próxima de 10%.

Tabela 2 – Descrição das bases de dados.

Grafo	$ V $	$ E $	Maior grau	Menor grau	Grau médio	Densidade
Zachary's Karate Club	34	72	17	1	4.58824	0.13904
Dolphins	62	159	12	1	5.12903	0.08408
Football	115	613	12	7	10.66087	0.09352
Political Blogs	643	2.280	165	1	7.09176	0.01105

Com exceção do grafo Karate Club, que pôde ser recuperado pela própria biblioteca NetworkX ⁴ do Python, os demais grafos foram obtidos por meio do The Network Data Repository ⁵ (ROSSI; AHMED, 2015), um repositório online de grafos que disponibiliza diversos tipos de redes. Vale destacar que esses grafos são simples, conexos, não direcionados e nem ponderados.

4.4 Otimização de hiperparâmetros

O processo de agrupamento de nós dos grafos consiste na geração dos *embeddings* dos grafos no espaço latente, seguido do agrupamento dos nós de acordo com essas representações. Assim, neste trabalho, os hiperparâmetros dos modelos consistem nos hiperparâmetros de dois componentes: modelos de *embedding* de grafo e modelos de agrupamento.

Devido à natureza intrinsecamente aleatória dos algoritmos de *embedding*, foi necessário criar um *Grid Search* personalizado específico para este trabalho a fim de

⁴ <<https://networkx.org/en/>>

⁵ <<https://networkrepository.com/>>

selecionar bons hiperparâmetros com base em scores confiáveis. A seguir, é descrito como se deu esse processo.

Inicialmente, a fim de facilitar esse fluxo, essas etapas foram combinadas por meio de uma instância da classe Pipeline ⁶ do scikit-learn, tendo o modelo de *embedding* como um transformador de dados anterior ao modelo de agrupamento. Para isso, foi criada uma classe que empacota cada modelo de *embedding* e herda das classes BaseEstimator e TransformerMixin do scikit-learn para que possa ser utilizada em um Pipeline.

Na primeira tentativa, foi utilizada a classe GridSearchCV ⁷ (AHMAD et al., 2022) do scikit-learn para busca dos hiperparâmetros. Como o objetivo do trabalho é avaliar o agrupamento de todos os nós do grafo, foi feita a parametrização para que seja utilizado todo o conjunto de dados em vez do comportamento de k-fold padrão.

Entretanto, observou-se que múltiplas execuções do mesmo Pipeline sobre o mesmo grafo resultavam em scores com certa variação do Coeficiente de Silhueta, o que acontece devido à presença de aleatoriedade nos algoritmos por trás dos modelos de *embedding*. Por isso, fez-se necessário um *Grid Search* que seja capaz de executar múltiplas vezes para cada conjunto de parâmetros.

Assim, foi definido que o score de um dado conjunto de parâmetros é representado pela média e desvio padrão de 10 execuções dessa configuração. Sabe-se que essa quantidade não é ideal e ainda é suscetível a *outliers* nos scores, mas valores maiores aumentariam ainda mais o tempo de processamento para os maiores grafos. Então, foi estabelecido esse valor fixo para manter a consistência dos testes sobre todos os grafos.

Nesta versão do *Grid Search*, cada Pipeline foi configurado com cada conjunto de parâmetros e testado iterativamente sobre cada grafo, o que levou a $G \times E \times EP \times C \times CP \times \text{num_iters}$ execuções, sendo G a quantidade de grafos, E a quantidade de modelos de *embedding*, EP a quantidade de conjuntos de parâmetros para *embedding*, C a quantidade de modelos de agrupamento, CP a quantidade de conjuntos de parâmetros para agrupamento e num_iters a quantidade de iterações para cada configuração. A equação 4.1 organiza as variáveis descritas anteriormente.

$$\begin{aligned}
 &\text{Número de execuções} = \text{Grafos} \\
 &\quad \times \text{Modelos de embedding} \times \text{Grade de parâmetros de embedding} \\
 &\quad \times \text{Modelos de agrupamento} \times \text{Grade de parâmetros de agrupamento} \\
 &\quad \times 10
 \end{aligned} \tag{4.1}$$

A fim de otimizar essa busca, abandonaram-se os Pipelines, desacoplando os processos de *embedding* e agrupamento. Assim, partiu-se para uma abordagem em que os

⁶ <<https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>>

⁷ <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>

embeddings gerados são compartilhados entre os modelos de agrupamento, i.e., dado um modelo de *embedding* configurado com um conjunto de parâmetros, gera-se o *embedding* de um grafo, então cada modelo de agrupamento configurado com cada conjunto de parâmetros é aplicado sobre essa representação, e os scores são salvos em uma lista. Isso é repetido `num_iters` vezes, o que reduz a quantidade de *embeddings* gerados e agrega mais consistência nos scores, já que eles são calculados com base nas mesmas representações vetoriais.

Um próximo passo foi paralelizar a geração dos *embeddings*, utilizando *threads* para calcular `num_iters` representações simultaneamente. Assim, para cada modelo de *embedding* e conjunto de parâmetros, primeiro gera-se `num_iters` representações, então itera-se sobre cada uma da mesma forma descrita anteriormente.

Além disso, destaca-se que os espaços de busca dos hiperparâmetros foram reduzidos para diminuir o tempo de processamento necessário, visto que mesmo com grafos pequenos, as buscas estavam despendendo muito tempo para serem concluídas. Isso se deu principalmente devido ao uso exclusivo de CPU para execução dos processos. Como a GPU tem uso limitado no Google Colab, não foi possível utilizá-la para a maioria dos testes. Ainda assim, tentou-se manter pelo menos 2 valores para os principais parâmetros de cada modelo.

Espaços de busca

Como mencionado anteriormente, a maioria dos hiperparâmetros teve apenas dois valores considerados. No Apêndice A é detalhado sobre os espaços de busca considerados, na Tabela 5 são listados os valores que compõem o espaço de busca dos modelos de *embedding*, enquanto que na Tabela 6 são listados os dos modelos de agrupamento.

O espaço de busca de cada par de modelos corresponde à união dos conjuntos de parâmetros de cada um. Assim, a quantidade de combinações de hiperparâmetros corresponde à multiplicação das quantidades de cada um dos modelos. Na Tabela 3 são apresentados os tamanhos desses espaços para os modelos sendo utilizados.

4.5 Teste sobre dados ruidosos

Após determinar os melhores parâmetros dentro dos escopos definidos para cada um dos modelos, foi testada sua robustez em relação à presença de ruído nas bases de dados, nesse caso, foram considerados dois tipos de ruídos.

- Ruído negativo: remoção aleatória de arestas dos grafos.
- Ruído positivo: adição aleatória de arestas aos grafos.

Tabela 3 – Tamanho dos espaços de busca de cada par de modelos.

Modelo de <i>embedding</i>	Modelo de agrupamentos	# de conjuntos de hiperparâmetros
DeepWalk	K-Means	384
DeepWalk	Affinity Propagation	144
DeepWalk	HDBSCAN	288
LINE	K-Means	384
LINE	Affinity Propagation	144
LINE	HDBSCAN	288
Node2Vec	K-Means	576
Node2Vec	Affinity Propagation	216
Node2Vec	HDBSCAN	432
SDNE	K-Means	288
SDNE	Affinity Propagation	108
SDNE	HDBSCAN	216
Struc2Vec	K-Means	256
Struc2Vec	Affinity Propagation	96
Struc2Vec	HDBSCAN	192

Fonte: Autor.

O nível de ruído é definido por uma porcentagem em relação à quantidade de arestas do grafo original, por exemplo, um ruído negativo de 10% indica a remoção de $0.1|E_o|$ arestas do grafo em questão, enquanto um ruído positivo de 5% indica a adição de $0.05|E_o|$ arestas, sendo $|E_o|$ a quantidade de arestas no grafo original.

Foram considerados diferentes níveis para cada tipo de ruído, para o ruído negativo definiu-se valores partindo de 0% incrementando 10% até o máximo de 50%. Por sua vez, para o ruído positivo, os valores partiram de 0% e foram incrementados em 5% até o máximo de 25%. A avaliação foi realizada aplicando cada par de valores de ruído negativo e positivo sobre cada grafo sendo considerado, o que soma 30 cenários.

Para cada um desses casos, os ruídos associados foram aplicados sobre os grafos e os melhores parâmetros encontrados para cada modelo foram utilizados para realizar *embeddings* e agrupamento dos dados. Por fim, de forma semelhante ao que foi feito anteriormente, a pontuação de cada processo foi definida por meio da média e do desvio padrão de 10 execuções dos modelos sobre os mesmos grafos ruidosos.

Vale destacar que essas 10 execuções foram realizadas sobre o mesmo grafo com ruído para cada um dos 30 cenários considerados, i.e., dado um grafo, foram aplicados ruído negativo e positivo sobre ele, então gerou-se seu *embedding* e determinou-se seus grupos por 10 vezes usando esse mesmo grafo.

Como a aplicação do ruído é feita de forma aleatória uma única vez, o grafo resultante pode apresentar uma topografia que favorece a separação de grupos em comparação com sua estrutura original, por isso, grafos ruidosos podem acabar levando a maiores Coeficientes de Silhueta em comparação com o grafo original.

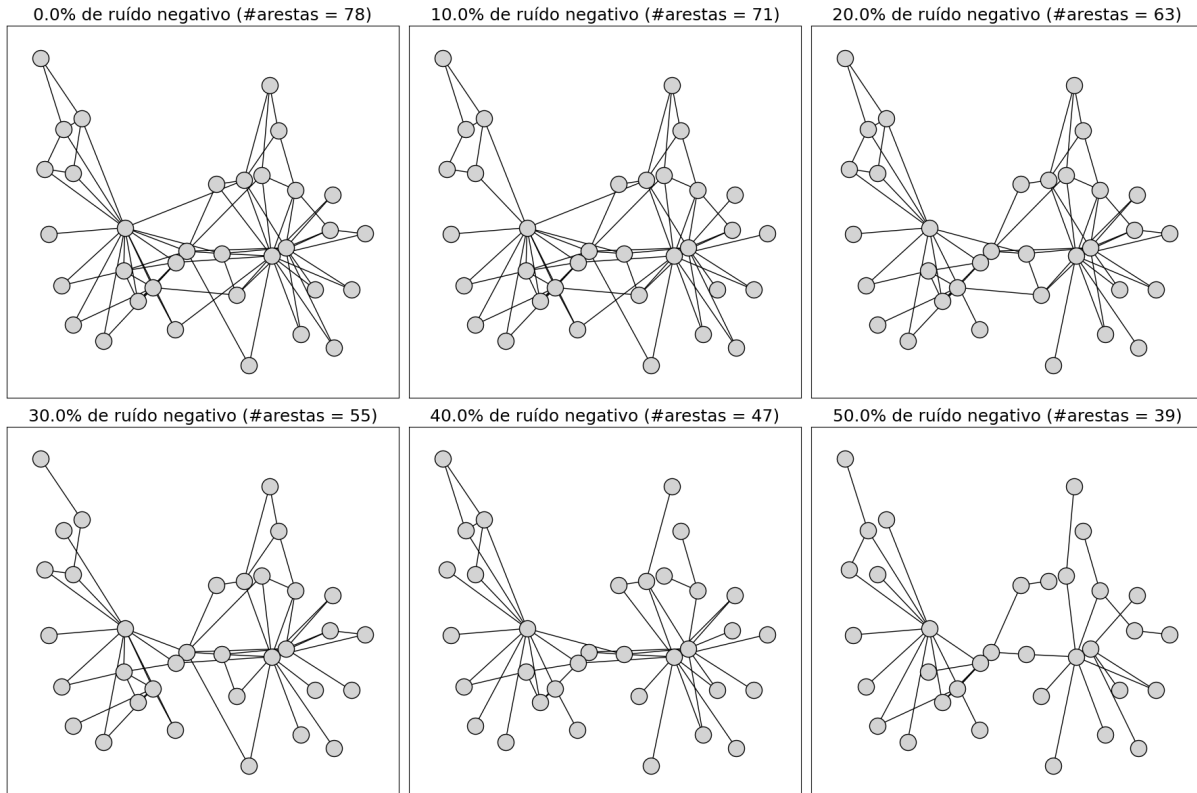
Essa questão não foi tratada no trabalho devido a certas limitações em relação ao tempo de processamento, que já estava alto considerando apenas um grafo e acabaria ficando inviável caso fosse repetido para múltiplos grafos.

A seguir, são apresentados grafos após a aplicação de ruído, a fim de permitir a visualização de como os diferentes ruídos afetam a estrutura de um grafo.

4.5.1 Ruído negativo

Na Figura 11, é apresentado o grafo Zachary's Karate Club com diferentes níveis de ruído negativo, o grafo com 0% é o original e os demais grafos foram gerados removendo $x\%$ de suas arestas. A aplicação do ruído foi feita de forma independente, e.g., o grafo com 20% de ruído foi gerado a partir do original, não do grafo com 10% de ruído.

Figura 11 – Grafos com diferentes níveis de ruído negativo aplicado.

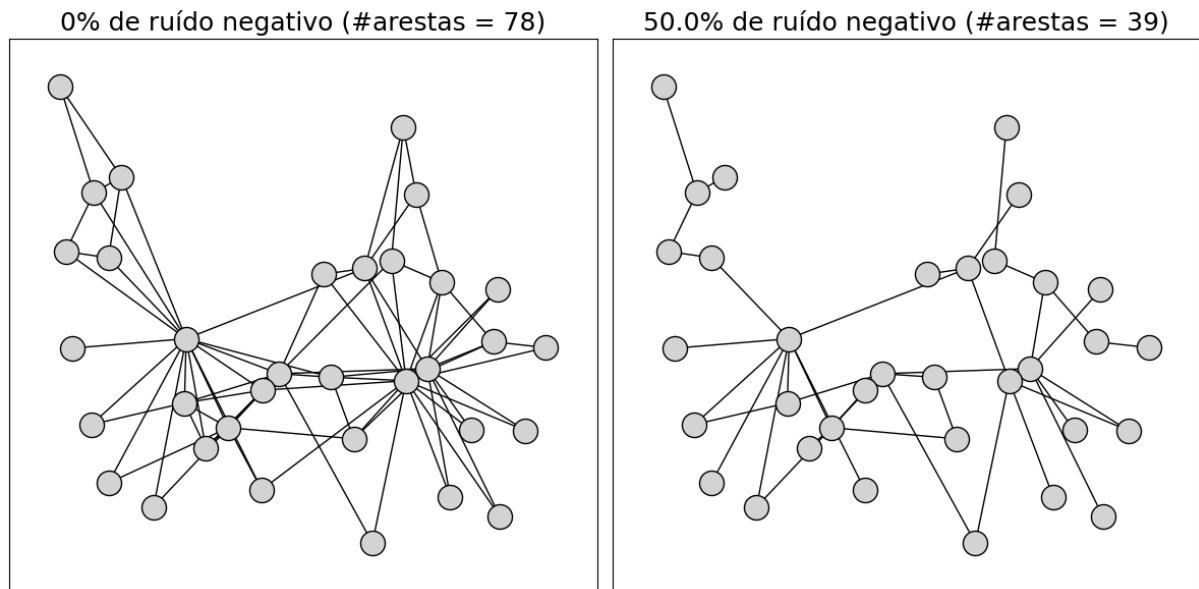


Fonte: Autor.

A seguir, na Figura 12, pode-se observar melhor a diferença entre o grafo original e o grafo com o ruído negativo máximo considerado, tendo apenas metade das arestas do

original. Como a remoção de arestas é feita de forma aleatória, ao comparar os grafos com ruído de 50% na Figura 11 e na Figura 12, nota-se que suas estruturas são consideravelmente diferentes.

Figura 12 – Grafos com ruído negativo nulo e máximo.

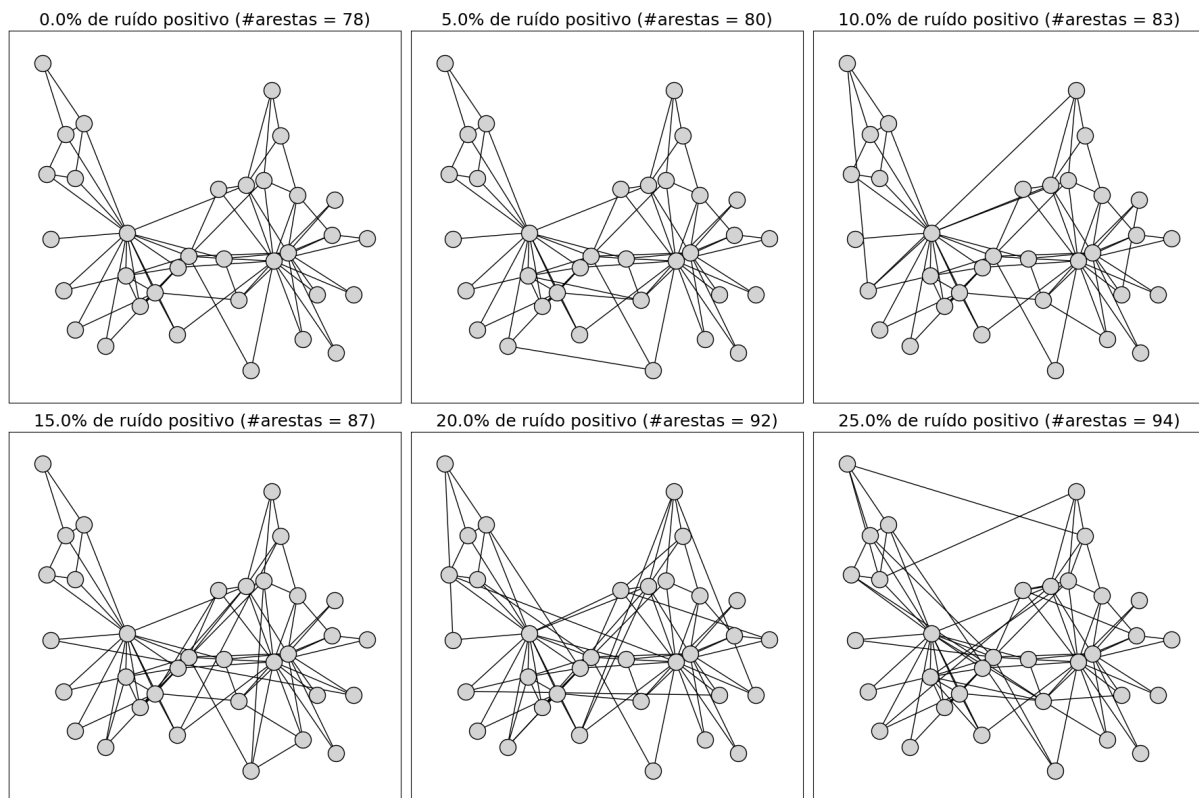


Fonte: Autor.

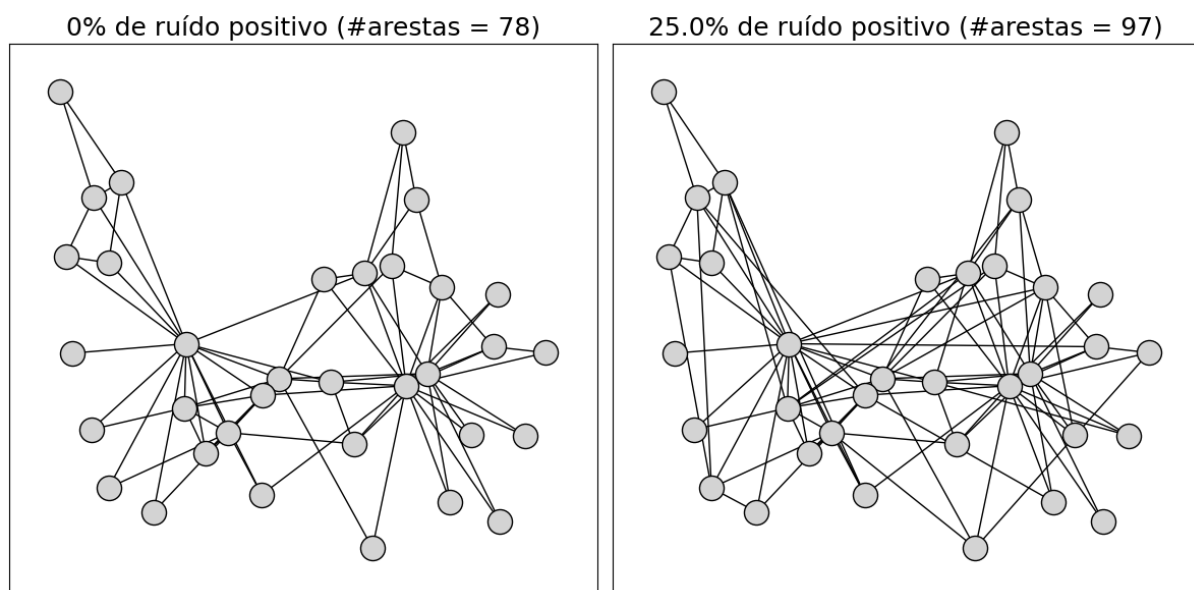
4.5.2 Ruído positivo

A aplicação do ruído positivo é análoga à do negativo, entretanto, dessa vez, são adicionadas novas arestas em vez de remover as existentes no grafo original. Na Figura 13, são apresentados diferentes níveis de ruído positivo aplicados ao grafo Zachary's Karate Club. De forma semelhante ao ruído negativo, a aplicação do ruído positivo foi feita independentemente em cada caso, sendo feitos em relação ao grafo original.

A seguir, são apresentados o grafo original e o grafo com o ruído positivo máximo considerado na Figura 14, sendo possível perceber um grafo bem mais denso após a aplicação do ruído. Também é possível perceber a diferença entre os grafos com ruído de 50% na Figura 13 e na Figura 14.

Figura 13 – Grafos com diferentes níveis de ruído positivo aplicado.

Fonte: Autor.

Figura 14 – Grafos com ruído positivo nulo e máximo.

Fonte: Autor.

5 Análise e discussão dos resultados

Este capítulo apresenta os resultados obtidos na busca por bons parâmetros para os modelos de *embedding* e agrupamento, demonstrados através dos scores de Silhueta obtidos dos agrupamentos gerados quando configurados com tais valores. Ademais, são apresentados os scores resultantes dos testes com grafos ruidosos para cada cenário sendo avaliado.

5.1 Busca de hiperparâmetros

A busca por bons hiperparâmetros para os modelos foi realizada par a par entre os modelos de *embedding* e agrupamento, sendo definidos de forma independente para cada conjunto de dados. Assim, com 5 modelos de *embedding*, 3 modelos de agrupamento e 4 bases de dados, foram definidos 60 conjuntos de melhores parâmetros.

Os gráficos apresentam os melhores scores do Coeficiente de Silhueta obtidos em cada cenário, utilizando os melhores parâmetros encontrados na busca realizada anteriormente.

Foram criados 4 gráficos, um para cada conjunto de dados, que possuem suas barras divididas por modelo de agrupamento. Assim, cada um é composto pelo conjunto dos scores obtidos por cada modelo de *embedding* junto com ele. Assim, há 3 grupos de barras, um para cada modelo em: K-Means, Affinity Propagation e HDBSCAN. As cores das barras representam o modelo de *embedding* utilizado para calcular o score.

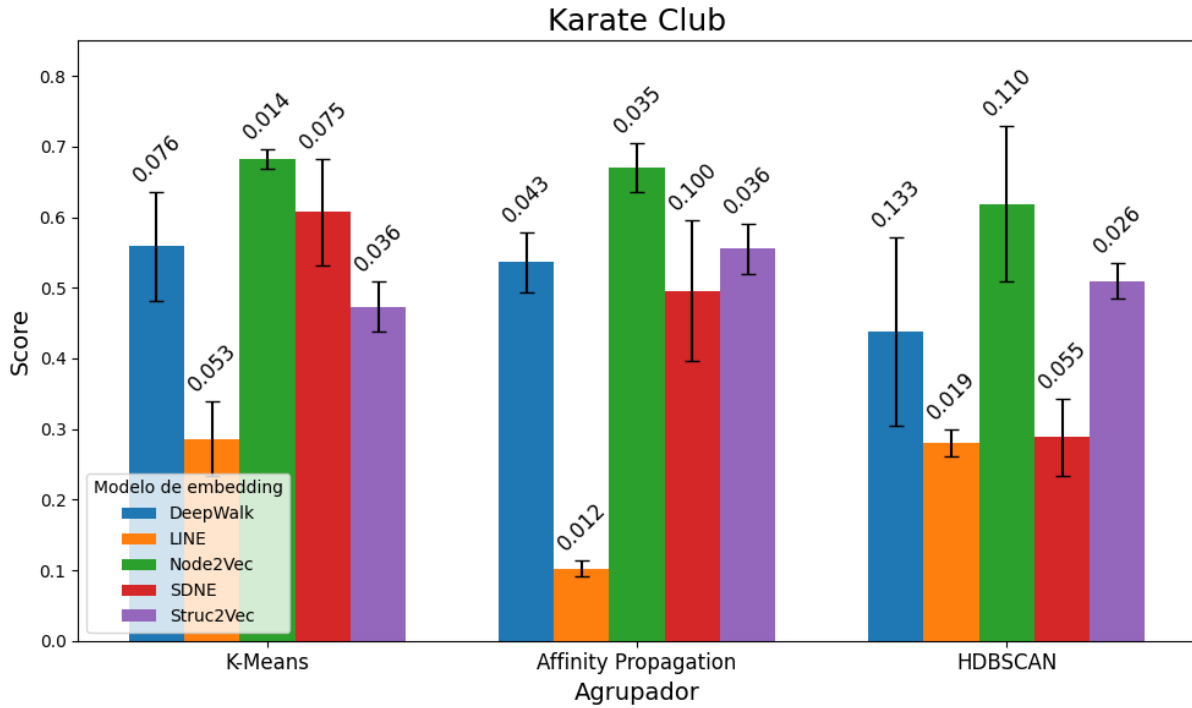
Além disso, os desvios padrão associados às médias de cada score são apresentados como barras de erro centralizadas em cada barra de score, os valores dos desvios padrão são incluídos acima dessas barras e sua altura é relacionada ao seu valor.

5.1.1 Zachary's Karate Club

Na Figura 15, com resultados em relação ao grafo Zachary's Karate Club, o modelo node2vec apresentou melhores resultados para todos os agrupadores; DeepWalk, SDNE e struc2vec tiveram resultados medianos, com SDNE apresentando uma queda quando usado junto com HDBSCAN; e LINE teve scores baixos em todos os casos.

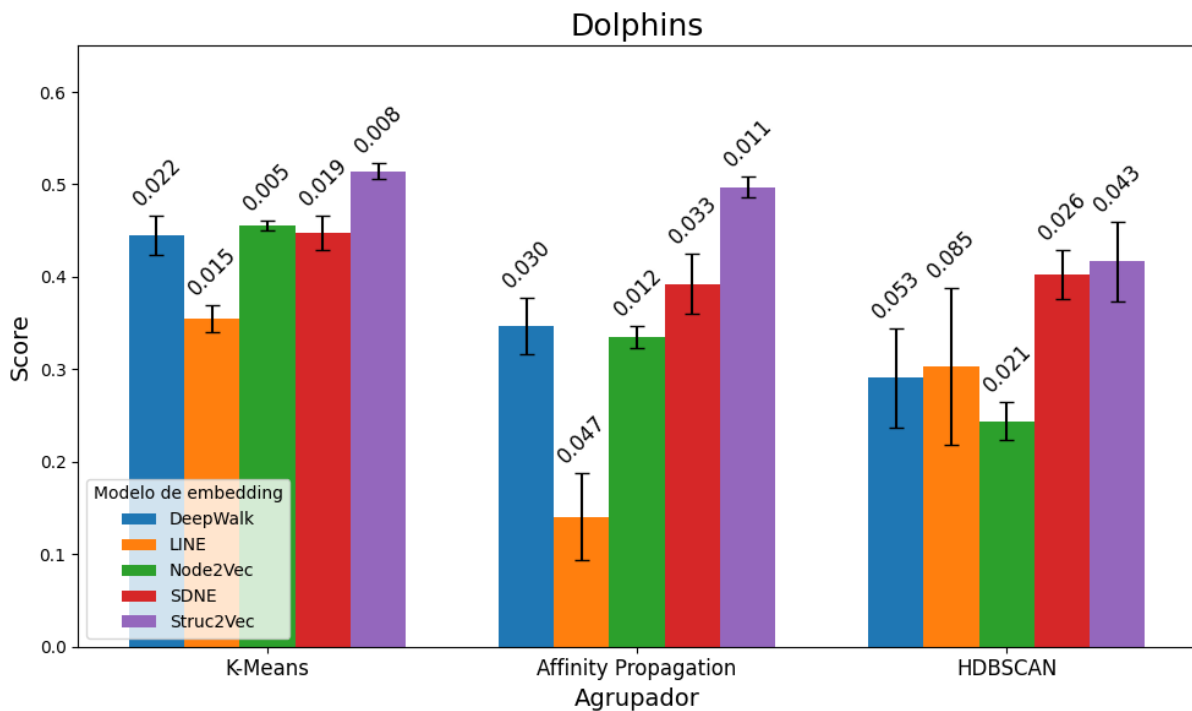
Nota-se que os desvios padrão, com exceção de struc2vec que teve valores mais consistentes, apresentaram maior variação entre diferentes cenários para os demais.

Figura 15 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Zachary’s Karate Club usando os melhores parâmetros encontrados.



Fonte: Autor.

Figura 16 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Dolphins usando os melhores parâmetros encontrados.



Fonte: Autor.

5.1.2 Dolphins

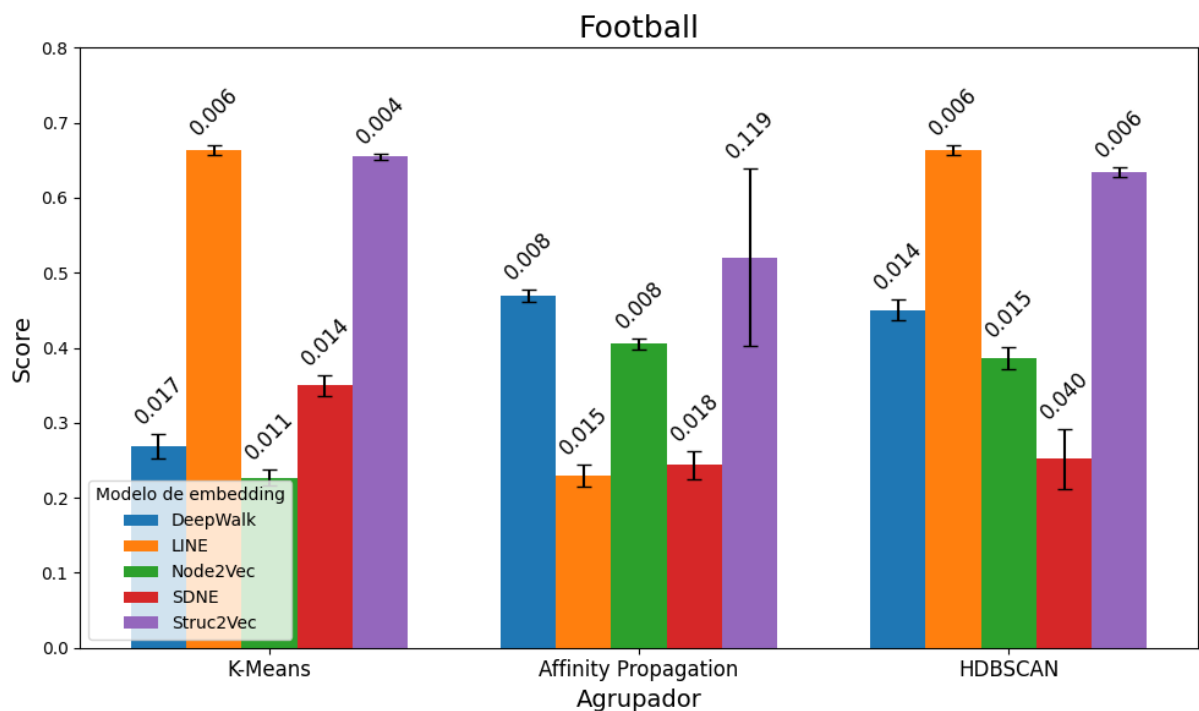
Na Figura 16, com resultados em relação ao grafo Dolphins, dessa vez, struc2vec apresentou melhores scores em todos os cenários, seguido por SDNE que teve resultados um pouco melhores que os outros quando usado com Affinity Propagation e muito próximo a struc2vec quando utilizado com HDBSCAN; DeepWalk, node2vec e SDNE obtiveram resultados um pouco piores para o restante dos casos; LINE teve pontuações baixas no geral, salvo quando usado com HDBSCAN.

Vale destacar que os scores foram medianos no geral, além de não apresentarem diferenças significativas entre as melhores pontuações e as demais.

Os desvios padrão apresentaram maior variação para todos os modelos de embedding, sendo um pouco mais estáveis em node2vec e SDNE.

5.1.3 Football

Figura 17 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Football usando os melhores parâmetros encontrados.



Fonte: Autor.

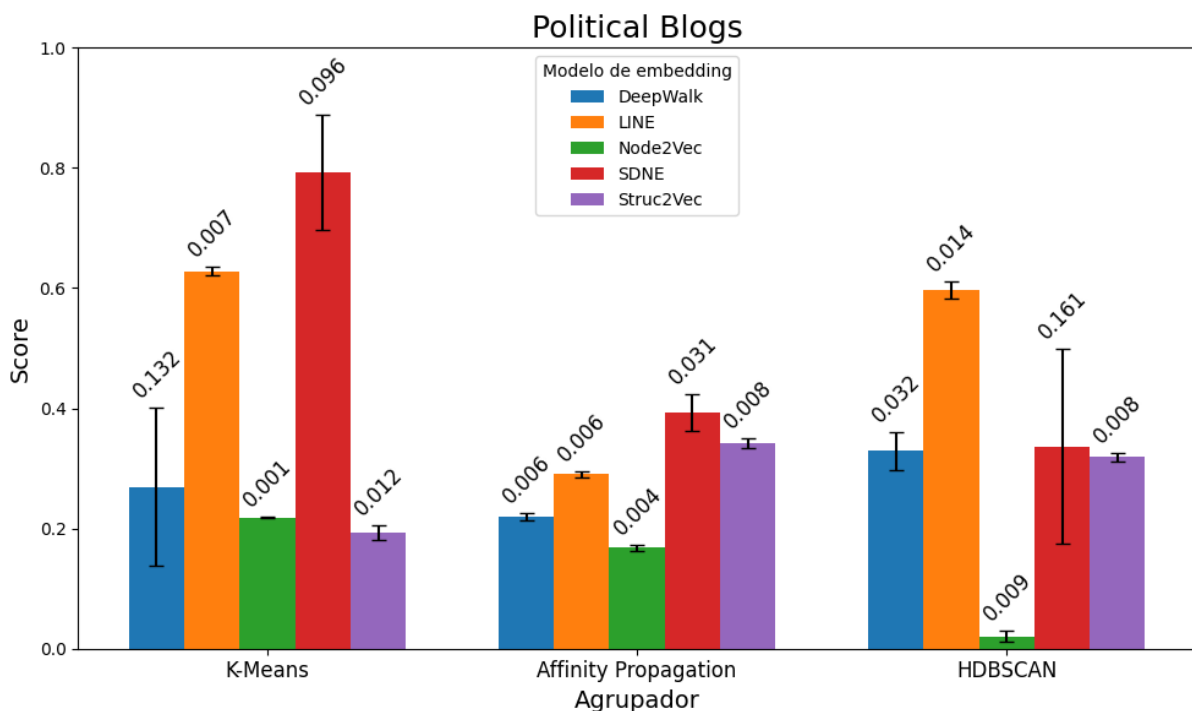
Na Figura 17, com resultados em relação ao grafo Football, LINE e struc2vec apresentaram as maiores pontuações em quase todos os casos, exceto que quando usados juntos com Affinity Propagation, em que ambos os scores foram menores, mas struc2vec manteve com o maior valor, enquanto LINE teve o pior desempenho nesse caso; DeepWalk e node2vec obtiveram resultados semelhantes em todos os casos, sendo piores junto com

K-Means; SDNE teve resultados piores no geral, mas junto a K-Means ficou um pouco acima dos piores.

Os desvios padrão mantiveram-se estáveis para a maioria dos casos, apresentando um salto maior para struc2vec e Affinity Propagation, e menor para SDNE e HDBSCAN.

5.1.4 Political Blogs

Figura 18 – Gráfico de barras de médias e desvios padrão dos scores de agrupamentos gerados sobre Political Blogs usando os melhores parâmetros encontrados.



Fonte: Autor.

Na Figura 18, com resultados em relação ao grafo Political Blogs, LINE e SDNE apresentaram pontuações mais altas junto a K-Means, e LINE teve uma pontuação maior junto a HDBSCAN; os demais métodos tiveram baixas pontuações nos outros casos; node2vec apresentou uma pontuação muito mais baixa junto a HDBSCAN.

Os desvios padrão mantiveram certa estabilidade no geral, tendo valores mais altos em DeepWalk e K-Means, e SDNE junto a K-Means e HDBSCAN.

5.1.5 Observações

Com base nos scores obtidos, pode-se observar que as pontuações variam bastante dependendo do grafo, do modelo de embedding e do modelo de agrupamento sendo utilizado. A melhor escolha é altamente dependente de cada situação, não sendo possível definir com certeza se um desses componentes será adequado ou não à aplicação apenas com base em

experimentos realizados em outras configurações. Assim, mostra-se necessária uma análise caso a caso para definir a melhor escolha para cada parte desse processo.

Na Tabela 4 é possível observar uma tendência de K-Means aparecer nos pares de melhores modelos, entretanto, também aparece uma vez como um dos piores. De forma semelhante, outros modelos têm esse mesmo comportamento, aparecendo tanto na coluna de melhores modelos quanto na de piores.

Tabela 4 – Melhores e piores pares de modelos para cada base de dados.

Base	Melhores modelos	Piores modelos
Zachary's Karate Club	node2vec + K-Means	LINE + Affinity Propagation
Dolphins	struc2vec + K-Means	LINE + Affinity Propagation
Football	LINE + K-Means/HDBSCAN	node2vec + K-Means
Political Blogs	SDNE + K-Means	node2vec + HDBSCAN

5.2 Teste de grafos ruidosos

A seguir, são apresentados gráficos com o Coeficiente de Silhueta médio junto ao desvio padrão de 10 execuções para cada uma das situações descritas anteriormente em 4.5. Há casos em que foi gerado apenas um único grupo, o que não permite o cálculo do coeficiente, por isso, foram considerados apenas os coeficientes de execuções com pelo menos 2 grupos gerados.

A princípio, espera-se que o desempenho dos agrupamentos fosse degradado com o aumento do ruído sobre os grafos, o que se traduziria em valores menores para quadrados mais à direita e para cima nos mapas de calor. Entretanto, como é possível observar nas figuras a seguir, esse comportamento não é uma regra, havendo casos que fogem dessa ideia.

Assim, são apresentados os resultados obtidos em relação a cada um dos grafos considerados: Karate Club, Dolphins, Football e Political Blogs. Os mapas de calor do cenário usando DeepWalk sobre o grafo Zachary's Karate Club são apresentados na Subseção 5.2.1.1 a seguir, mas os gráficos dos demais casos estão listados no Apêndice B para não estender demais este capítulo.

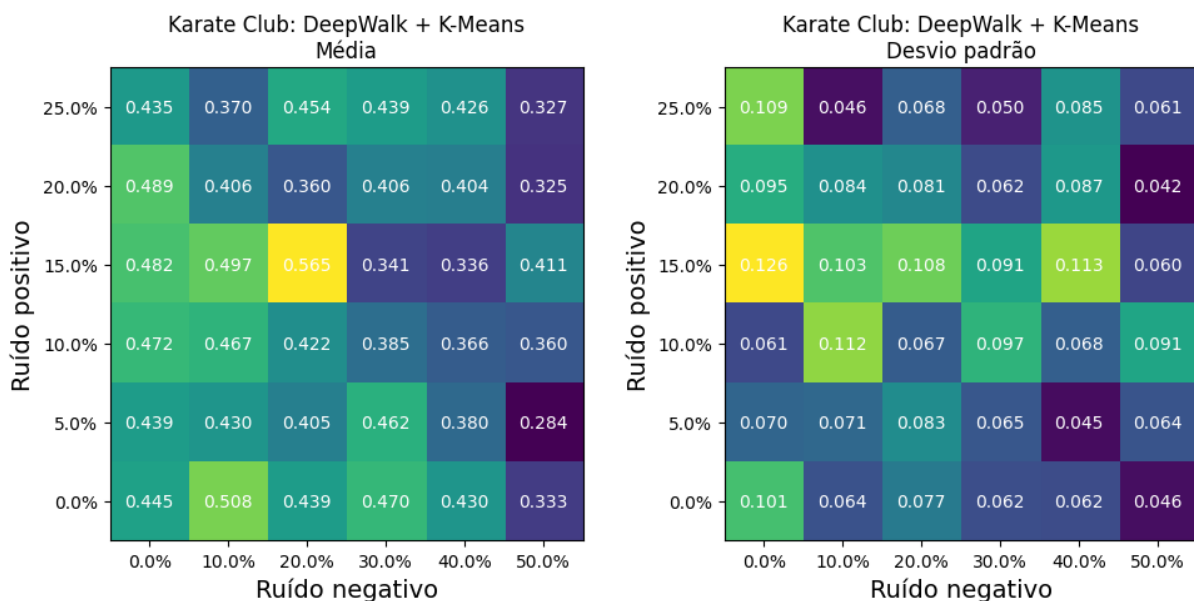
5.2.1 Zachary's Karate Club

Os *heatmaps* em relação ao DeepWalk são apresentados a seguir na Subseção 5.2.1.1, os demais gráficos estão na Seção B.1 do Apêndice B, tendo LINE na Subseção B.1.1, node2vec na Subseção B.1.2, SDNE na Subseção B.1.3 e struc2vec na Subseção B.1.4.

5.2.1.1 DeepWalk

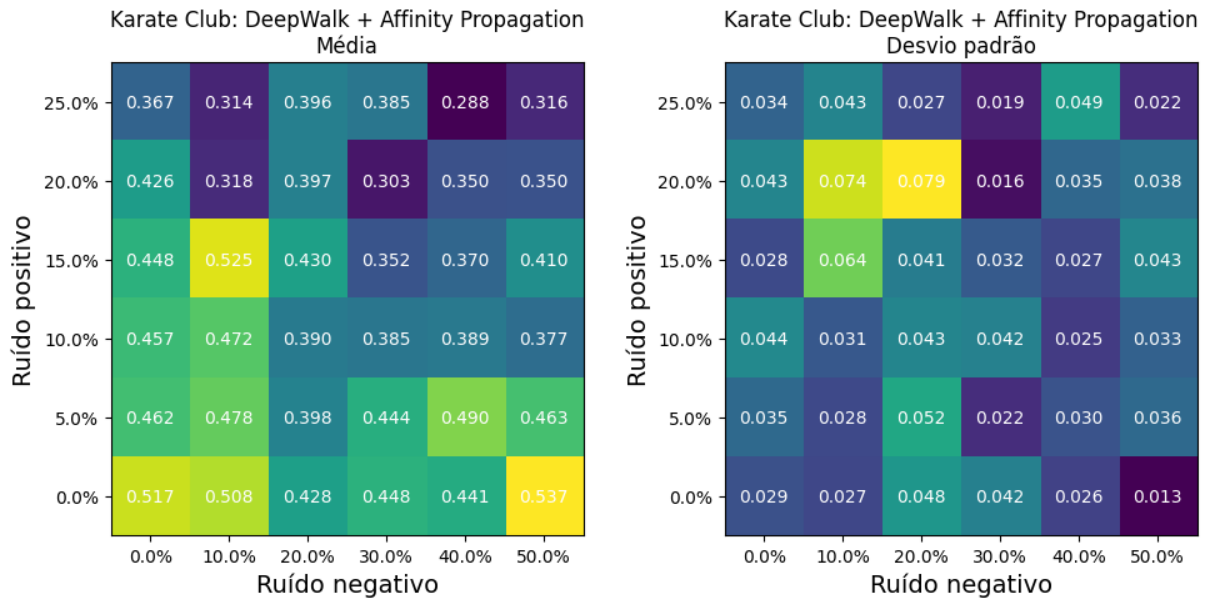
É possível observar médias maiores na região do triângulo inferior, seguindo o comportamento esperado na Figura 19, Figura 20 e Figura 21, mas há scores mais altos em certas combinações de ruído, o que pode ser interpretado como casos em que o ruído acabou favorecendo a criação de grupos artificiais. Já sobre o desvio padrão, não é possível identificar uma relação entre seu valor e os ruídos aplicados.

Figura 19 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Zachary's Karate Club com diferentes ruídos.



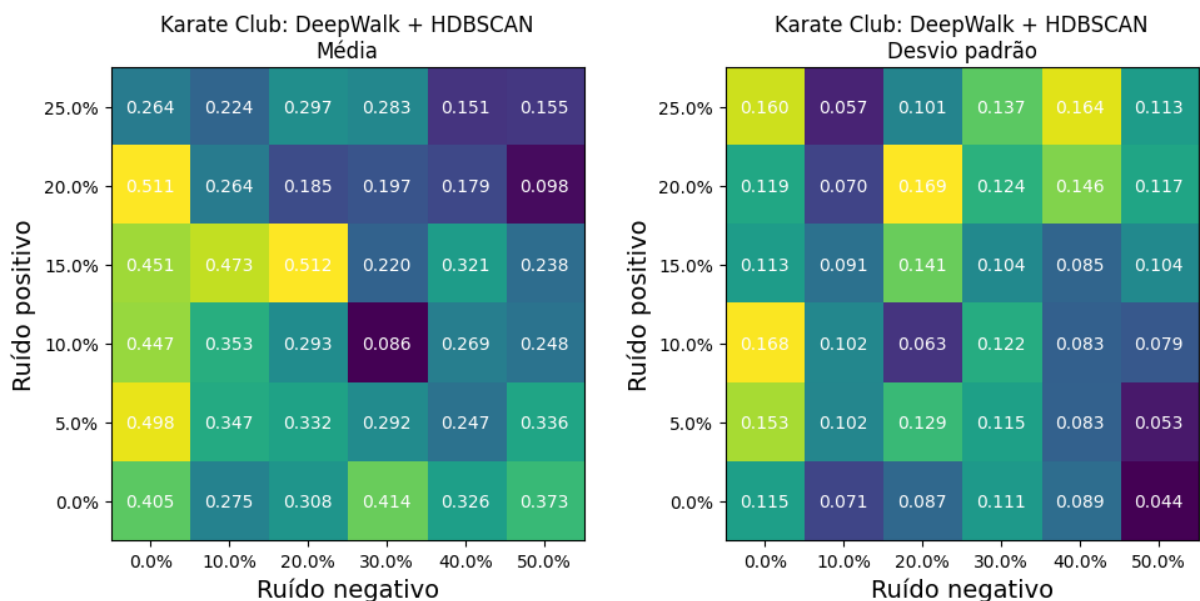
Fonte: Autor.

Figura 20 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

Figura 21 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

5.2.1.2 LINE

Dessa vez, as maiores médias estão na porção esquerda da matriz na Figura 22, Figura 23 e Figura 24, apresentando até uma tendência para valores mais altos de ruído positivo, a qual se repete até certo ponto no desvio padrão. Nota-se quadrados vazios na Figura 24, nesses casos, todas as 10 execuções acabaram gerando um único grupo, o que não permitiu o cálculo do Índice de Silhueta.

5.2.1.3 node2vec

Também segue o comportamento esperado, com o triângulo inferior tendo as maiores médias, além de quadrados um pouco acima dele também apresentando scores altos, no K-Means (Figura 25) e no HDBSCAN (Figura 27), o que pode indicar maior robustez do modelo. Já o desvio padrão apresentou características diversas para cada modelo de agrupamento, sendo maior com ruídos negativos maiores em K-Means, maior com ruídos negativos menores em Affinity Propagation (Figura 26) e mais homogêneo com HDBSCAN.

5.2.1.4 SDNE

Quando junto com Affinity Propagation (Figura 29) e HDBSCAN (Figura 30), as maiores médias estavam próximas do canto inferior direito (ruído negativo máximo e positivo nulo), com desvios padrão distribuídos de forma uniforme. Por sua vez, SDNE com K-Means (Figura 28) manteve-se com médias consistentes em diversos níveis de ruído, assim como seus desvios padrão, vale notar alguns pontos isolados em que os resultados foram melhores ou piores.

5.2.1.5 struc2vec

Para struc2vec, as médias se mantiveram consistentes para cada modelo de agrupamento (Figura 31, Figura 32 e Figura 33), apresentando apenas alguns pontos com valores destoantes do restante, mas sem um padrão que possa indicar suas causas. Os desvios padrão seguem esse comportamento, tendo valores similares durante os testes.

5.2.2 Dolphins

Os *heatmaps* estão na seção B.2 do Apêndice B, tendo DeepWalk na Subseção B.2.1, LINE na Subseção B.2.2, node2vec na Subseção B.2.3, SDNE na Subseção B.2.4 e struc2vec na Subseção B.2.5.

5.2.2.1 DeepWalk

As médias de K-Means (Figura 34) e HDBSCAN (Figura 36) foram mais altas no triângulo inferior, enquanto Affinity Propagation (Figura 35) teve melhores resultados na região inferior, com menores ruídos positivos. Os desvios padrões demonstraram-se razoavelmente uniformes em todos os casos.

5.2.2.2 LINE

Semelhante ao que foi visto em DeepWalk, as médias foram maiores quando o ruído positivo foi mais alto e ruído negativo, mais baixo, na Figura 37, Figura 38 e Figura 39, mas é possível observar nos mapas de calor que há uma "linha" a partir da qual os valores aumentam, fato que se repete para todos os casos. Uma análise mais aprofundada seria necessária para descobrir sua causa. Os desvios padrão parecem bem uniformes novamente.

5.2.2.3 node2vec

As maiores médias aparecem na parte inferior do mapa de calor, onde o ruído positivo é menor, junto ao K-Means (Figura 40), apresentam valores mais próximos ao esperado, tendo médias maiores e desvios padrão menores quando mais próximos do quadrado sem ruído. Por sua vez, junto com Affinity Propagation (Figura 41) e HDBSCAN (Figura 42), as maiores as médias são maiores quando mais próximas do quadrado com ruído negativo máximo e ruído positivo nulo, o que pode indicar que foram detectadas comunidades artificiais provenientes da remoção de arestas, nesses casos, os desvios padrão não parecem seguir nenhum padrão.

5.2.2.4 SDNE

Semelhante ao comportamento visto em node2vec, as maiores médias tendem a aparecer na região inferior (Figura 43), e SDNE junto a Affinity Propagation (Figura 44) e HDBSCAN (Figura 45) tem médias mais altas próximas do canto inferior direito (ruído negativo máximo e positivo nulo). Os desvios padrão têm uma leve tendência a serem maiores próximos desse mesmo quadrado.

5.2.2.5 struc2vec

Diferente dos demais modelos, no struc2vec as maiores médias apareceram na região direita do mapa de calor, onde o ruído negativo é maior, o que pode indicar que as arestas removidas aumentaram a representatividade estrutural dos nós do grafo, possivelmente a redução de arestas tornou nós mais semelhantes do ponto de vista estrutural.

Os desvios padrão junto a Affinity Propagation (Figura 47) são baixos em sua maioria; junto a K-Means (Figura 46), parecem maiores próximos ao canto inferior direito; e junto a HDBSCAN (Figura 48) parecem menores próximos a essa região.

5.2.3 Football

Os *heatmaps* estão na seção B.3 do Apêndice B, com DeepWalk na Subseção B.3.1, LINE na Subseção B.3.2, node2vec na Subseção B.3.3, SDNE na Subseção B.3.4 e struc2vec na Subseção B.3.5.

5.2.3.1 DeepWalk

As médias dos scores seguem o comportamento esperado nas Figura 49, Figura 50 e Figura 51, com valores maiores próximos do quadrado com ruídos nulos. Os desvios padrão não demonstram ter uma distribuição definida.

5.2.3.2 LINE

Exceto pela combinação com Affinity Propagation (Figura 53), que teve valores mais próximos do esperado, com médias e desvios padrão maiores, próximos do quadrado sem ruído, o modelo LINE apresentou médias altas quando utilizado junto ao K-Means (Figura 52) e HDBSCAN (Figura 54), que se mantiveram consistentes para praticamente todos os cenários ruidosos considerados. Além disso, os desvios padrão também ficaram consistentemente baixos para cada situação.

5.2.3.3 node2vec

Também apresentou valores próximos ao esperado de se observar, com médias maiores próximas no cenário com ruídos nulos nas Figura 55, Figura 56 e Figura 57. Junto ao K-Means (Figura 55), o desvio padrão também mostrou tender a apresentar valores mais altos próximos desse quadrado, enquanto que com os outros modelos não parece seguir um padrão.

5.2.3.4 SDNE

SDNE apresentou resultados diferentes para cada modelo de agrupamento: junto ao K-Means (Figura 58), teve resultados mais consistentes, com médias semelhantes para diferentes níveis de ruído e desvios padrão um pouco maiores para ruídos médios; junto ao Affinity Propagation (Figura 59), teve médias mais altas e menores desvios padrão do lado esquerdo (menor ruído negativo) enquanto teve médias menores e desvios padrão maior do lado direito (maior ruído negativo); junto ao HDBSCAN (Figura 60) a região à

direita (maior ruído negativo) teve médias mais altas, enquanto os desvios padrão não apresentaram distribuição aparente.

5.2.3.5 struc2vec

As médias ficaram constantes para diferentes níveis de ruído e diferentes modelos de agrupamento, sendo menores para ruídos negativos mais altos. Com Affinity Propagation (Figura 62) e K-Means (Figura 61), os desvios padrão foram baixos durante os testes, mas com HDBSCAN (Figura 63), eles foram mais baixos com ruídos menores. Vale destacar que com K-Means, a média do cenário sem ruído foi consideravelmente mais alta comparada aos demais, em contrapartida, com Affinity Propagation esse comportamento se inverteu, com os cenários com ruído tendo médias maiores que aquele sem.

5.2.4 Political Blogs

Os *heatmaps* estão na seção B.4 do Apêndice B, com DeepWalk na Subseção B.4.1, LINE na Subseção B.4.2, node2vec na Subseção B.4.3 e SDNE na Subseção B.4.4.

5.2.4.1 DeepWalk

As médias mostraram seguir o esperado, com valores mais altos nos cenários de baixo ruído e diminuindo conforme os níveis de ruído crescem. Os desvios padrão tiveram valores variados para cada teste.

Diferente dos scores vistos obtidos em relação aos grafos anteriores, DeepWalk apresentou médias que fogem do esperado, mas ainda possui valores mais altos na região inferior (menor ruído negativo). Junto ao K-Means (Figura 64), as maiores médias são vistas quando o ruído negativo é médio e o positivo é nulo, junto ao Affinity Propagation (Figura 65), isso acontece quando o ruído negativo é máximo e o positivo é nulo, já junto ao HDBSCAN (Figura 66), apresenta médias mais altas quando não há ruído, enquanto o restante das médias (quando há ruído) são muito mais baixas.

5.2.4.2 LINE

Junto ao HDBSCAN (Figura 69), o modelo apresentou médias e desvios padrão consistentes durante os testes, apenas com certos pontos destoantes. Com Affinity Propagation (Figura 68), isso também aconteceu na maior parte dos casos, mas próximo do canto inferior direito (maior ruído negativo e menor ruído positivo), as médias foram menores e os desvios padrão, maiores, indicando que a redução de arestas do grafo prejudicou mais essa combinação de modelos. Com K-Means (Figura 67), as médias foram maiores na região inferior (menor ruído positivo), enquanto que os desvios padrão foram menores na região esquerda (menor ruído negativo).

5.2.4.3 node2vec

Quando utilizado junto com K-Means (Figura 70), os resultados seguiram o comportamento esperado, com médias maiores nos testes com ruídos menores, no entanto, percebe-se que o aumento do ruído positivo teve um efeito mais intenso do que o ruído negativo, visto que as médias horizontais são semelhantes enquanto que verticalmente essa diferença é maior. Os desvios padrão mantiveram-se mais baixos durante os testes. Por sua vez junto ao Affinity Propagation (Figura 71) e HDBSCAN (Figura 72), as médias seguiram um padrão semelhante ao esperado, mas com referência ao canto inferior direito, o que pode indicar que a remoção de arestas aproximou os nós em relação a seus papéis estruturais no grafo. Os desvios padrão tiveram valores constantes no decorrer dos testes.

5.2.4.4 SDNE

Junto ao K-Means (Figura 73) e HDBSCAN (Figura 75), as médias apresentaram valores mais constantes nos testes, apresentando alguns valores menores nos cantos. Para K-Means isso ocorreu quando ambos os ruídos estavam altos, enquanto que para HDBSCAN isso foi com ruídos mais baixos. Seus desvios padrão seguiram esse comportamento, com valores menores onde sua média era maior. Por sua vez, com Affinity Propagation (Figura 74), as médias foram maiores no canto inferior direito (ruído negativo máximo e positivo nulo), e seus desvios padrão foram menores nessa região também.

5.2.4.5 struc2vec

Os testes da aplicação de struc2vec sobre o grafo Political Blogs não foram realizados devido a limites em relação ao tempo de processamento, que acabou se mostrando muito alto nesse caso.

5.2.5 Resultados relevantes

Considerando todos os cenários descritos em 5.2, foram observados os seguintes comportamentos:

- **DeepWalk e comportamento esperado:** usando o modelo DeepWalk em cada um dos diferentes cenários, aplicado junto aos modelos de agrupamento para cada grafo, é possível perceber que, consistentemente, as médias são maiores nos cenários em que ambos os ruídos são mais baixos, com exceção apenas dos cenários do grafo Political Blogs, em que todos os resultados fogem do padrão visto nos demais casos. Assim, pode-se perceber que DeepWalk é suscetível à presença de ruído nos dados.
- **LINE e variabilidade:** o modelo LINE apresentou resultados com distribuições distintas para cada cenário em que foi aplicado. Em determinadas situações, as médias

foram consistentes durante os testes (como visto na Subseção 5.2.3.2 e Subseção 5.2.4.2), enquanto que em outros, os resultados são mais próximos do esperado ou apresentam uma distribuição com causas pouco claras.

- **node2vec e melhores resultados:** o node2vec apresentou tanto comportamentos mais próximos do esperado quanto cenários que fogem dele, visto que há testes em que as maiores médias foram obtidas quando o ruído negativo era máximo e positivo era nulo, podendo indicar que os grafos ruidosos podem favorecer a identificação de grupos, sejam eles equivalentes ao que é visto nos dados originais ou gerados pelas alterações da estrutura dos grafos.
- **SDNE e ausência de padrões:** observando os diferentes cenários de teste, pode-se notar que a distribuição das médias utilizando SDNE varia bastante caso a caso, tendo médias consistentes, médias maiores próximas a um dos cantos ou médias mais altas em certas regiões, dependendo do grafo e do modelo de agrupamento que o acompanham.
- **struc2vec, constância e variação:** semelhante a alguns casos vistos para outros modelos, tiveram testes em que as médias foram consistentes para diferentes níveis de ruído, mas, no caso do struc2vec, houveram mais pontos de variação quando comparado a cenários parecidos de LINE e SDNE. Também houve testes em que as médias mais altas estavam associadas ao aumento do ruído negativo, podendo indicar que a remoção de arestas dos grafos uniformizou a representação estrutural de seus nós, favorecendo a identificação e separação de grupos.

6 Conclusão

Tendo em vista os resultados apresentados e descritos no capítulo 5, pode-se observar que os scores do Coeficiente de Silhueta são altamente dependentes da aplicação sendo considerada, como é demonstrado pelos gráficos em 5.1. Definindo um cenário específico, como a construção dos *embeddings* dos nós de um grafo por meio de um modelo de *embedding*, seguido da definição de grupos através de um modelo de agrupamento com base nesses vetores numéricos, destacam-se os três componentes desse processo: grafo, modelo de *embedding* e modelo de agrupamento, os quais definem em conjunto o desempenho da tarefa de agrupamento que está sendo realizada. Assim, a troca de uma única dessas partes já impacta drasticamente os resultados produzidos e, consequentemente, os scores dos grupos definidos.

Isso também é refletido nos testes realizados sobre grafos ruidosos, como descrito com mais detalhes na Seção 5.2 do Capítulo 5. Cada par de modelos de *embedding* e agrupamento apresentou resultados e comportamentos distintos quando aplicado sobre cada um dos conjuntos de dados. As médias dos scores dos grupos gerados nos testes apresentaram distribuições diversas, dependendo do cenário sendo avaliado, um mesmo par de modelos pode apresentar médias consistentes em relação a diferentes níveis de ruído sobre um grafo, mas ser suscetível à degradação quando o ruído é aplicado a outro grafo.

Com isso, foi possível constatar que a adaptação da tarefa de detecção de comunidades de nós de um grafo, quando transformada no agrupamento dos *embeddings* de seus nós, introduziu novas variáveis no processo, sendo elas os modelos de *embedding* e de agrupamento, que exigem testagem e validação a fim de definir candidatos adequados para a aplicação. Entretanto, a grande quantidade de métodos, tanto para *embedding* de grafo quanto para agrupamento de dados, cada um com seu conjunto de parâmetros específicos, acaba tornando essa abordagem menos atraente, visto a grande quantidade de hiperparâmetros que precisam ser otimizados, além do tempo e poder de processamento exigidos para isso, o que pode tornar sua aplicabilidade muito complexa ou ser necessário reduzir o conjunto de opções sendo consideradas, podendo descartar bons candidatos caso não haja uma boa análise para isso.

Portanto, pôde-se concluir que a adaptação da tarefa de detecção de comunidades de grafos nas tarefas de geração de *embeddings* dos nós e de agrupamento dessas representações acaba resultando em um processo complexo, tendo em vista as diversas opções de modelos disponíveis para cada tarefa, os hiperparâmetros específicos que cada um deles traz consigo, além das próprias bases de dados sendo trabalhadas, que podem favorecer modelos diferentes, dependendo de suas características e estruturas. Por isso, é necessário um estudo

mais aprofundado e detalhado sobre essa abordagem, a fim de identificar se há cenários em que sua aplicação é adequada ou se os benefícios trazidos por ela recompensam as dificuldades que a acompanham.

6.1 Limitações

Os experimentos foram realizados no Google Colab, que disponibiliza o uso de GPUs de forma limitada em seus ambientes, quando usados nos planos gratuitos, o que levou a tempos de processamento maiores durante os testes e acabou limitando o que foi avaliado no trabalho.

As bases de dados consideradas consistem em um conjunto reduzido de grafos pequenos, ponderou-se sobre adicionar grafos maiores, mas o tempo de processamento se mostrou muito alto nesses casos, o que inviabilizou sua inclusão no trabalho.

Como os testes foram feitos considerando pares de modelos de *embedding* e agrupamento, durante a busca por seus hiperparâmetros, cada conjunto de parâmetros dos modelos de *embedding* foi combinado com os parâmetros dos de agrupamento para serem então avaliados. Com isso, foi necessário reduzir o campo de busca considerado, tendo apenas dois valores para cada um dos principais parâmetros de cada modelo.

Além disso, cada um deles foi aplicado 10 vezes, que não é um valor muito alto e ainda torna média e desvio padrão suscetíveis a scores muito altos ou muito baixos em uma de suas execuções.

Durante os testes de ruído, cada caso de ruído é aplicado uma única vez e o grafo ruidoso resultante é o mesmo nas 10 iterações realizadas. Isso acaba tornando os resultados suscetíveis à forma como as arestas no grafo são removidas e adicionadas durante a aplicação do ruído. Idealmente, seriam gerados múltiplos grafos para cada cenário ruidoso, mas isso deixaria essa etapa inviável na configuração utilizada neste trabalho.

Outro ponto de atenção é a implementação dos modelos de *embedding* utilizada, que é baseada nos códigos de (SHENWEICHEN, 2013), a busca por outras implementações mais eficientes poderia reduzir o tempo de processamento dos experimentos.

6.2 Trabalhos futuros

Levando em consideração as limitações mencionadas anteriormente, um trabalho que expanda as opções consideradas aqui poderia dar uma visão mais ampla sobre os conceitos e ideias analisados.

Também seria interessante realizar a comparação da abordagem utilizada (*embedding* e agrupamento) com algoritmos de detecção de comunidades que atuam diretamente

sobre grafos, considerando a qualidade dos grupos/comunidades encontrados, o tempo e poder de processamento exigidos, além de repetir os testes de robustez sobre grafos ruidosos.

Além disso, há outras tarefas de aprendizado de máquina que poderiam ser substituir o agrupamento neste trabalho, como classificação e previsão de *links*, a fim de analisar seu comportamento em relação à presença de ruídos nos grafos.

Referências

ADAMIC, L. A.; GLANCE, N. The political blogosphere and the 2004 us election: divided they blog. In: ACM. *Proceedings of the 3rd international workshop on Link discovery*. [S.l.], 2005. p. 36–43. Citado na página 56.

AHMAD, G. N. et al. Efficient medical diagnosis of human heart diseases using machine learning techniques with and without gridsearchcv. *IEEE Access*, v. 10, p. 80151–80173, 2022. Citado na página 57.

AHO, A. V.; HOPCROFT, J. E. *The design and analysis of computer algorithms*. [S.l.]: Pearson Education India, 1974. Citado 2 vezes nas páginas 33 e 35.

AIZAWA, A. An information-theoretic perspective of tf-idf measures. *Information Processing & Management*, Elsevier, v. 39, n. 1, p. 45–65, 2003. Citado na página 41.

ANGLES, R.; GUTIERREZ, C. Survey of graph database models. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 40, n. 1, feb 2008. ISSN 0360-0300. Disponível em: <<https://doi.org/10.1145/1322432.1322433>>. Citado na página 28.

BHATTACHARYA, R.; NAGWANI, N. K.; TRIPATHI, S. A community detection model using node embedding approach and graph convolutional network with clustering technique. *Decision Analytics Journal*, Elsevier, v. 9, p. 100362, 2023. Citado na página 53.

BONDY, J. A.; MURTY, U. S. R. et al. *Graph theory with applications*. [S.l.]: Macmillan London, 1976. v. 290. Citado na página 31.

BRIDLE, J. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In: TOURETZKY, D. (Ed.). *Advances in Neural Information Processing Systems*. Morgan-Kaufmann, 1989. v. 2. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/1989/file/0336dcbab05b9d5ad24f4333c7658a0e-Paper.pdf>. Citado 2 vezes nas páginas 37 e 43.

CAI, H.; ZHENG, V. W.; CHANG, K. C.-C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. on Knowl. and Data Eng.*, IEEE Educational Activities Department, USA, v. 30, n. 9, p. 1616–1637, set. 2018. ISSN 1041-4347. Disponível em: <<https://doi.org/10.1109/TKDE.2018.2807452>>. Citado 3 vezes nas páginas 28, 44 e 51.

CAMPELLO, R. J. G. B. et al. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, Association for Computing Machinery, New York, NY, USA, v. 10, n. 1, jul 2015. ISSN 1556-4681. Disponível em: <<https://doi.org/10.1145/2733381>>. Citado na página 39.

DEHGHAN-KOOSHKHAZI, A. et al. Evaluating node embeddings of complex networks. *Journal of Complex Networks*, v. 10, n. 4, p. cnac030, 08 2022. ISSN 2051-1329. Disponível em: <<https://doi.org/10.1093/comnet/cnac030>>. Citado 2 vezes nas páginas 52 e 53.

DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 10, p. 78–87, 2012. Citado na página 35.

EBERENDU, A. C. et al. Unstructured data: an overview of the data of big data. *International Journal of Computer Trends and Technology*, v. 38, n. 1, p. 46–50, 2016. Citado na página 27.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. [S.l.]: AAAI Press, 1996. (KDD'96), p. 226–231. Citado na página 38.

FREY, B. J.; DUECK, D. Clustering by passing messages between data points. *science*, American Association for the Advancement of Science, v. 315, n. 5814, p. 972–976, 2007. Citado na página 38.

FROME, A. et al. Devise: A deep visual-semantic embedding model. *Advances in neural information processing systems*, v. 26, 2013. Citado na página 42.

GÉCZY, P. Big data characteristics. *The Macrotheme Review*, v. 3, n. 6, p. 94–104, 2014. Citado na página 27.

GHAHRAMANI, Z. Unsupervised learning. In: *Summer school on machine learning*. [S.l.]: Springer, 2003. p. 72–112. Citado na página 36.

GIRVAN, M.; NEWMAN, M. E. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 99, n. 12, p. 7821–7826, 2002. Citado na página 56.

GOODFELLOW, I. *Deep learning*. [S.l.]: MIT press, 2016. Citado na página 36.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 855–864. ISBN 9781450342322. Disponível em: <<https://doi.org/10.1145/2939672.2939754>>. Citado 2 vezes nas páginas 46 e 47.

GUO, L.; DAI, Q. Graph clustering via variational graph embedding. *Pattern Recognition*, Elsevier, v. 122, p. 108334, 2022. Citado 2 vezes nas páginas 52 e 53.

HARTIGAN, J. A.; WONG, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, [Royal Statistical Society, Oxford University Press], v. 28, n. 1, p. 100–108, 1979. ISSN 00359254, 14679876. Disponível em: <<http://www.jstor.org/stable/2346830>>. Citado na página 37.

II, M. T. S.; BRADLEY, J. N.; HANDEL, T. G. Data embedding method. In: *SPIE. Integration Issues in Large Commercial Media Delivery Systems*. [S.l.], 1996. v. 2615, p. 226–259. Citado na página 27.

KAMIŃSKI, B.; PRAŁAT, P.; THÉBERGE, F. An unsupervised framework for comparing graph embeddings. *Journal of Complex Networks*, Oxford University Press (OUP), v. 8, n. 5, Nov 2019. ISSN 2051-1329. Disponível em: <<http://dx.doi.org/10.1093/comnet/cnz043>>. Citado na página 52.

- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 35.
- KNUTH, D. E.; KNUTH, D. *The Stanford GraphBase: a platform for combinatorial computing*. [S.l.]: AcM Press New York, 1993. v. 1. Citado na página 46.
- KOTSIANTIS, S. B. et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, Amsterdam, v. 160, n. 1, p. 3–24, 2007. Citado na página 35.
- LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015. Citado na página 36.
- LIASHCHYNSKYI, P.; LIASHCHYNSKYI, P. *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*. 2019. Disponível em: <<https://arxiv.org/abs/1912.06059>>. Citado na página 40.
- LUSSEAU, D. et al. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, Springer, v. 54, n. 4, p. 396–405, 2003. Citado na página 56.
- MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: . [s.n.], 1967. Disponível em: <<https://api.semanticscholar.org/CorpusID:6278891>>. Citado na página 37.
- MIKOLOV, T. et al. Efficient estimation of word representations in vector space. In: *International Conference on Learning Representations*. [s.n.], 2013. Disponível em: <<https://api.semanticscholar.org/CorpusID:5959482>>. Citado na página 42.
- MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Red Hook, NY, USA: Curran Associates Inc., 2013. (NIPS'13), p. 3111–3119. Citado na página 42.
- NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018. Citado na página 36.
- PANKRATZ, B.; KAMIŃSKI, B.; PRAŁAT, P. Community detection supported by node embeddings (searching for a suitable method). In: CHERIFI, H. et al. (Ed.). *Complex Networks and Their Applications XI*. Cham: Springer International Publishing, 2023. p. 221–232. ISBN 978-3-031-21131-7. Citado 2 vezes nas páginas 52 e 53.
- PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2014. (KDD '14), p. 701–710. ISBN 9781450329569. Disponível em: <<https://doi.org/10.1145/2623330.2623732>>. Citado 3 vezes nas páginas 43, 44 e 45.
- PFALTZ, J. L. Graph structures. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 19, n. 3, p. 411–422, 1972. Citado na página 31.

RIBEIRO, L. F.; SAVERESE, P. H.; FIGUEIREDO, D. R. struc2vec: Learning node representations from structural identity. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2017. (KDD '17), p. 385–394. ISBN 9781450348874. Disponível em: <<https://doi.org/10.1145/3097983.3098061>>. Citado na página 48.

ROSSI, R. A.; AHMED, N. K. The network data repository with interactive graph analytics and visualization. In: *AAAI*. [s.n.], 2015. Disponível em: <<https://networkrepository.com>>. Citado na página 56.

ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, v. 20, p. 53–65, 1987. ISSN 0377-0427. Disponível em: <<https://www.sciencedirect.com/science/article/pii/0377042787901257>>. Citado na página 39.

SHENWEICHEN. *GraphEmbedding*. [S.l.]: GitHub, 2013. <<https://github.com/shenweichen/GraphEmbedding>>. Citado na página 78.

SHWARTZ-ZIV, R.; ARMON, A. Tabular data: Deep learning is not all you need. *Information Fusion*, v. 81, p. 84–90, 2022. ISSN 1566-2535. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1566253521002360>>. Citado na página 27.

SIDDIQA, A.; KARIM, A.; GANI, A. Big data storage technologies: a survey. *Frontiers of Information Technology & Electronic Engineering*, ZUP, v. 18, n. 8, p. 1040–1070, 2017. Citado na página 27.

SINGH, S.; HOODA, S. A study of challenges and limitations to applying machine learning to highly unstructured data. In: IEEE. *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*. [S.l.], 2023. p. 1–6. Citado na página 27.

SWANSON, M. D.; KOBAYASHI, M.; TEWFIK, A. H. Multimedia data-embedding and watermarking technologies. *Proceedings of the IEEE*, IEEE, v. 86, n. 6, p. 1064–1087, 1998. Citado na página 27.

TANDON, A. et al. Community detection in networks using graph embeddings. *Physical Review E*, American Physical Society (APS), v. 103, n. 2, Feb 2021. ISSN 2470-0053. Disponível em: <<http://dx.doi.org/10.1103/PhysRevE.103.022316>>. Citado 2 vezes nas páginas 51 e 53.

TANG, J. et al. Line: Large-scale information network embedding. In: *Proceedings of the 24th International Conference on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2015. (WWW '15), p. 1067–1077. ISBN 9781450334693. Disponível em: <<https://doi.org/10.1145/2736277.2741093>>. Citado na página 45.

VAPNIK, V. *The nature of statistical learning theory*. [S.l.]: Springer science & business media, 2013. Citado na página 35.

WANG, D.; CUI, P.; ZHU, W. Structural deep network embedding. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 1225–1234.

ISBN 9781450342322. Disponível em: <<https://doi.org/10.1145/2939672.2939753>>. Citado 3 vezes nas páginas 47, 48 e 49.

XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on neural networks*, Ieee, v. 16, n. 3, p. 645–678, 2005. Citado na página 36.

ZACHARY, W. An information flow model for conflict and fission in small groups1. *Journal of anthropological research*, v. 33, 11 1976. Citado na página 55.

Apêndices

APÊNDICE A – Espaços de busca

Neste apêndice são apresentados os espaços de busca considerados durante o processo de otimização de hiperparâmetros dos modelos de embedding e de agrupamento descrito na subseção 4.4 do Capítulo 4.

Tabela 5 – Valores considerados para cada hiperparâmetro dos modelos de *embedding*.

Modelo	Hiperparâmetro	Valores
DeepWalk	walk_lenght	5, 10, 20
DeepWalk	num_walks	80, 120
DeepWalk	embed_size	64, 128
DeepWalk	window_size	5, 10
DeepWalk	iter	5, 10
LINE	embedding_size	64, 128
LINE	negative_ratio	5, 10
LINE	order	first, second, all
LINE	batch_size	128, 256
LINE	epochs	5, 10
node2vec	walk_length	10, 20
node2vec	num_walks	80, 120
node2vec	p	0.5, 1, 2
node2vec	q	0.5, 1, 2
node2vec	embed_size	64, 128
SDNE	hidden_size	[512, 256, 128], [256, 128, 64]
SDNE	alpha	1e-06, 0.1, 0.2
SDNE	beta	5, 10, 20
SDNE	epochs	5, 10
struc2vec	walk_length	20, 30
struc2vec	num_walks	50, 100
struc2vec	embed_size	64, 128
struc2vec	window_size	10, 20
struc2vec	iter	5, 10

Fonte: Autor.

Tabela 6 – Valores considerados para cada hiperparâmetro dos modelos de agrupamento.

Modelo	Hiperparâmetro	Valores
K-Means	<code>n_clusters</code>	5, 10, 20
K-Means	<code>init</code>	k-means++
K-Means	<code>algorithm</code>	lloyd, elkan
Affinity Propagation	<code>damping</code>	0.5, 0.75, 0.9
Affinity Propagation	<code>preference</code>	None
HDBSCAN	<code>min_cluster_size</code>	3, 5, 10
HDBSCAN	<code>leaf_size</code>	30, 50

Fonte: Autor.

APÊNDICE B – Mapas de Calor

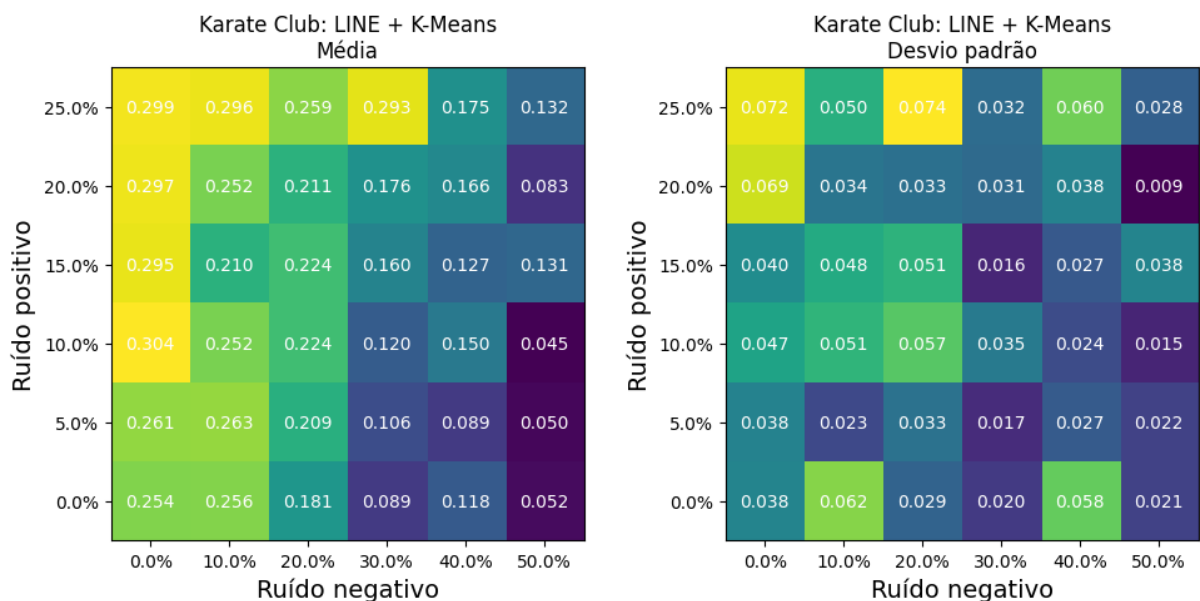
Neste apêndice, são apresentados os mapas de calor para cada caso de teste com ruído descrito na Subseção 4.5, com exceção dos testes usando DeepWalk sobre o grafo Zachary’s Karate Club.

Os *heatmaps* a seguir são referenciados na análise dos resultados feita no Capítulo 5.

B.1 Zachary’s Karate Club

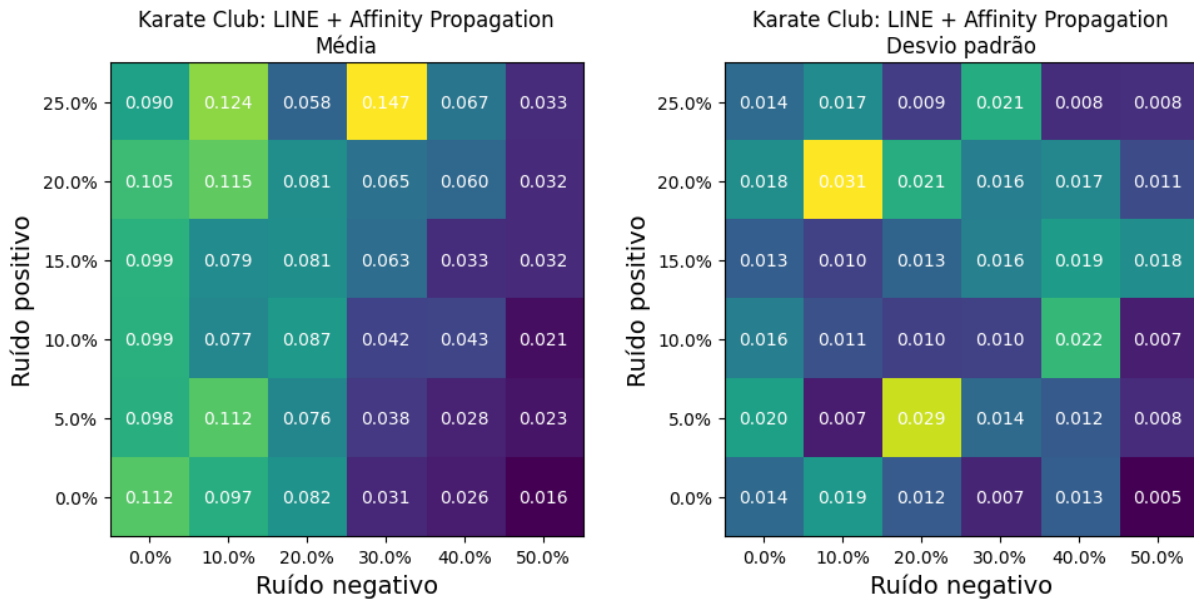
B.1.1 LINE

Figura 22 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Zachary’s Karate Club com diferentes ruídos.



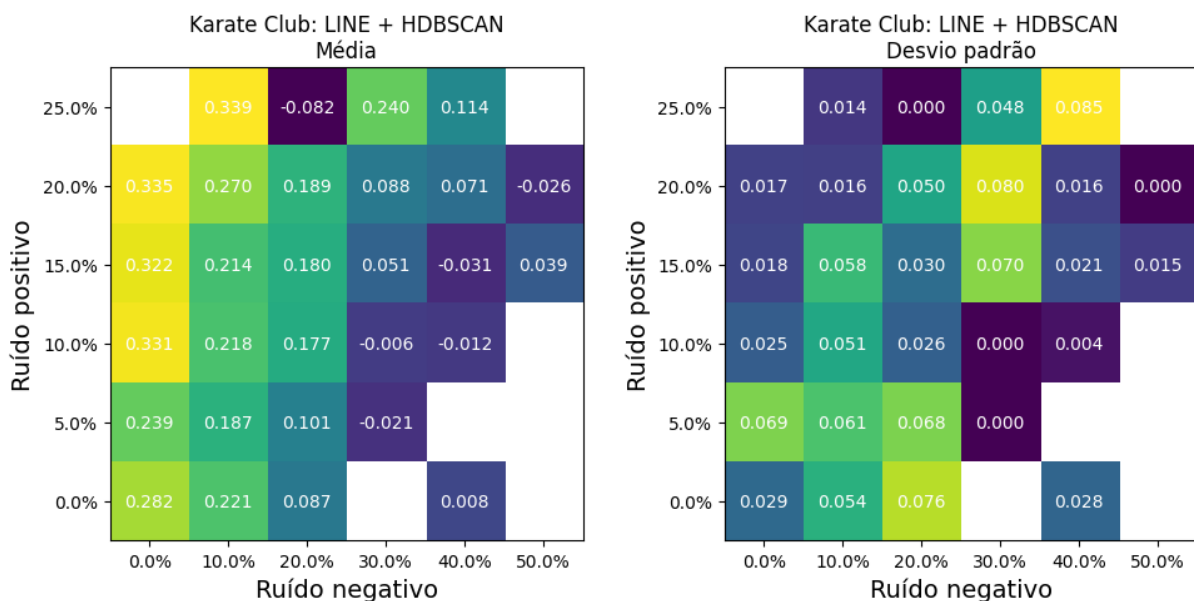
Fonte: Autor.

Figura 23 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

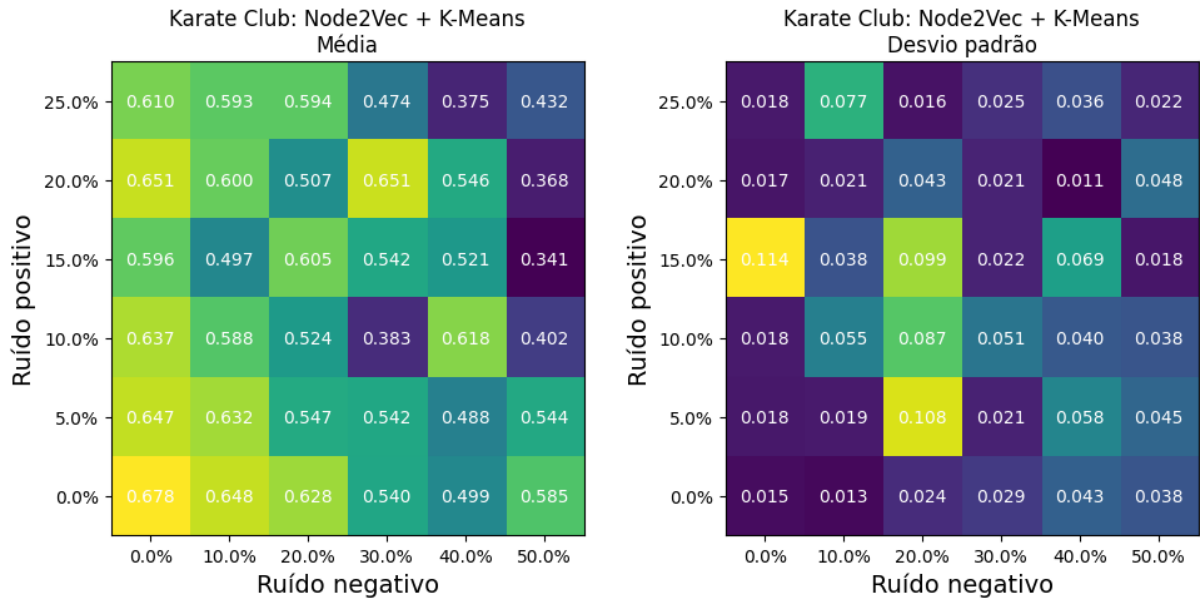
Figura 24 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

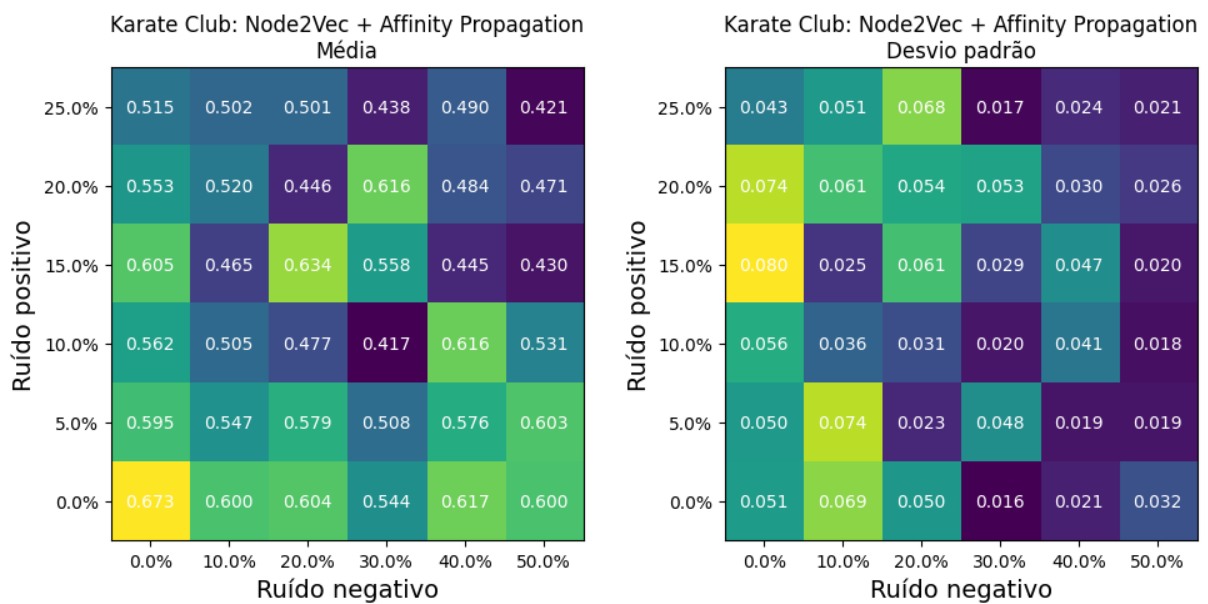
B.1.2 node2vec

Figura 25 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Zachary's Karate Club com diferentes ruídos.



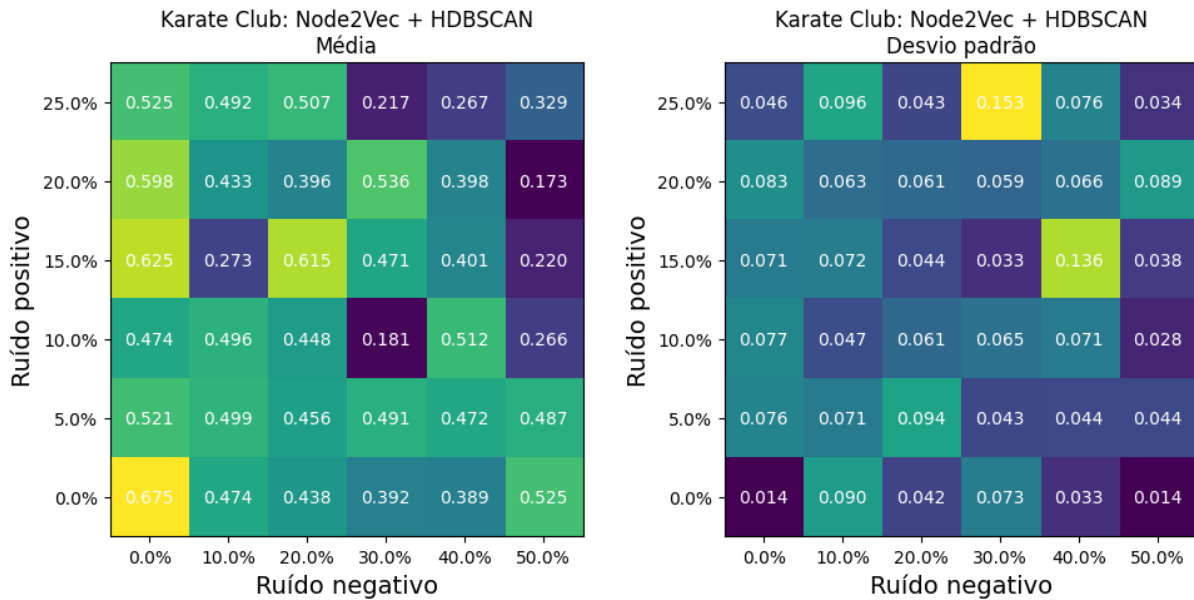
Fonte: Autor.

Figura 26 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

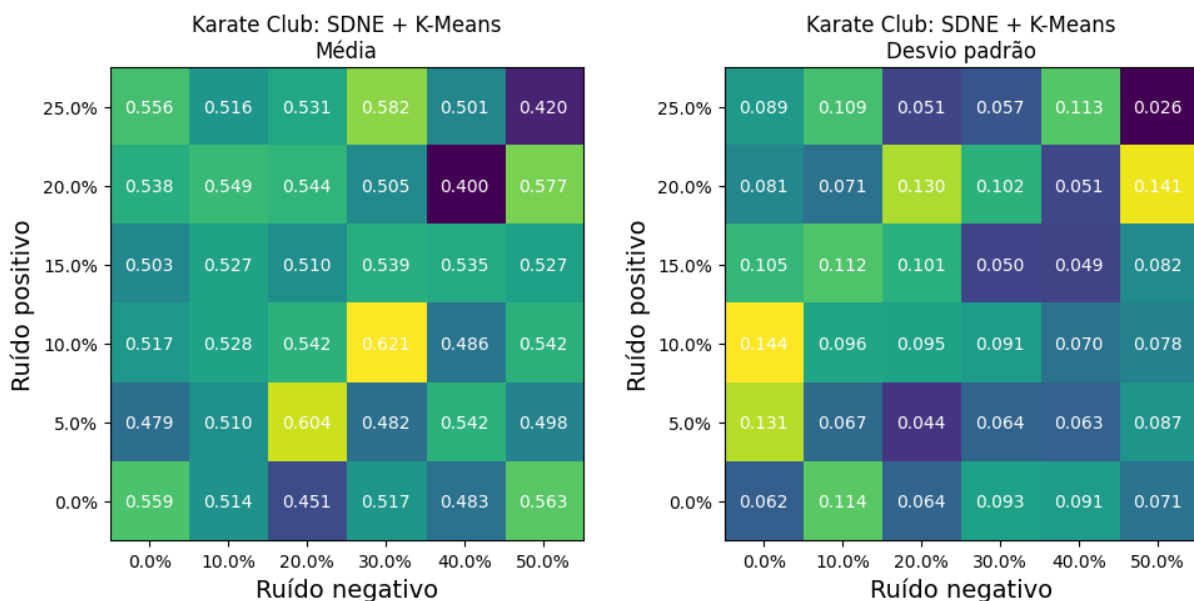
Figura 27 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

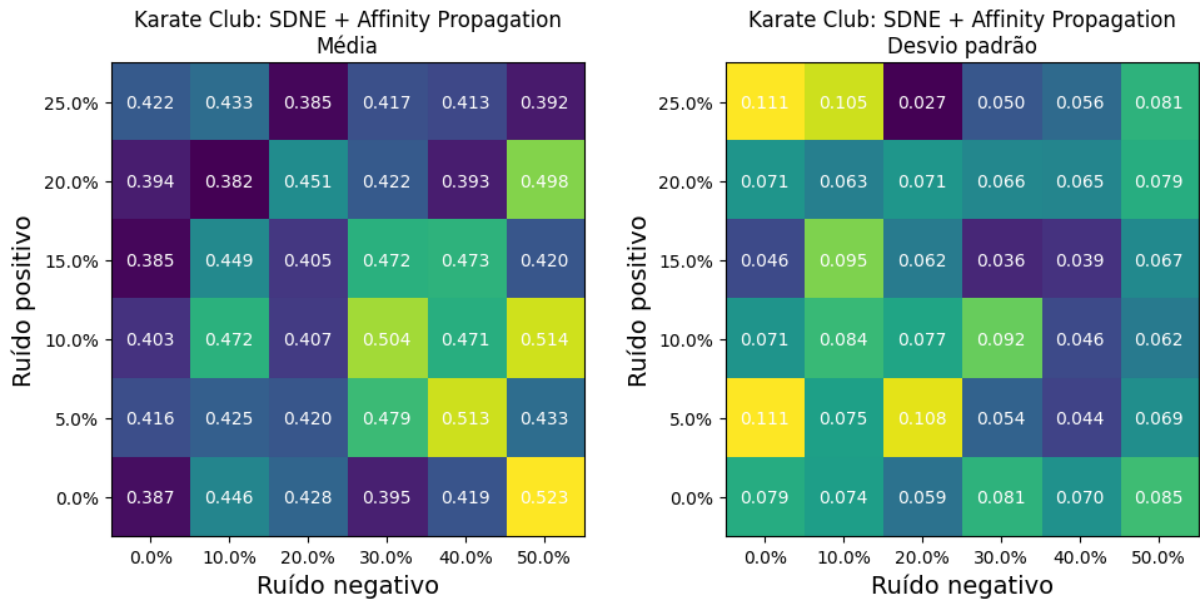
B.1.3 SDNE

Figura 28 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Zachary's Karate Club com diferentes ruídos.



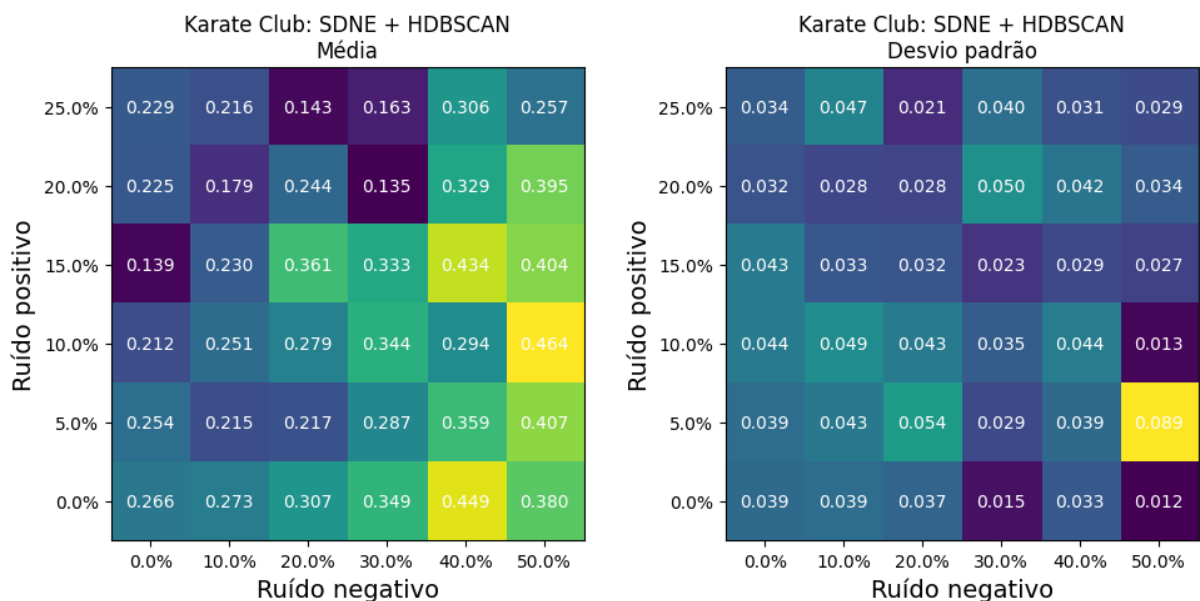
Fonte: Autor.

Figura 29 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

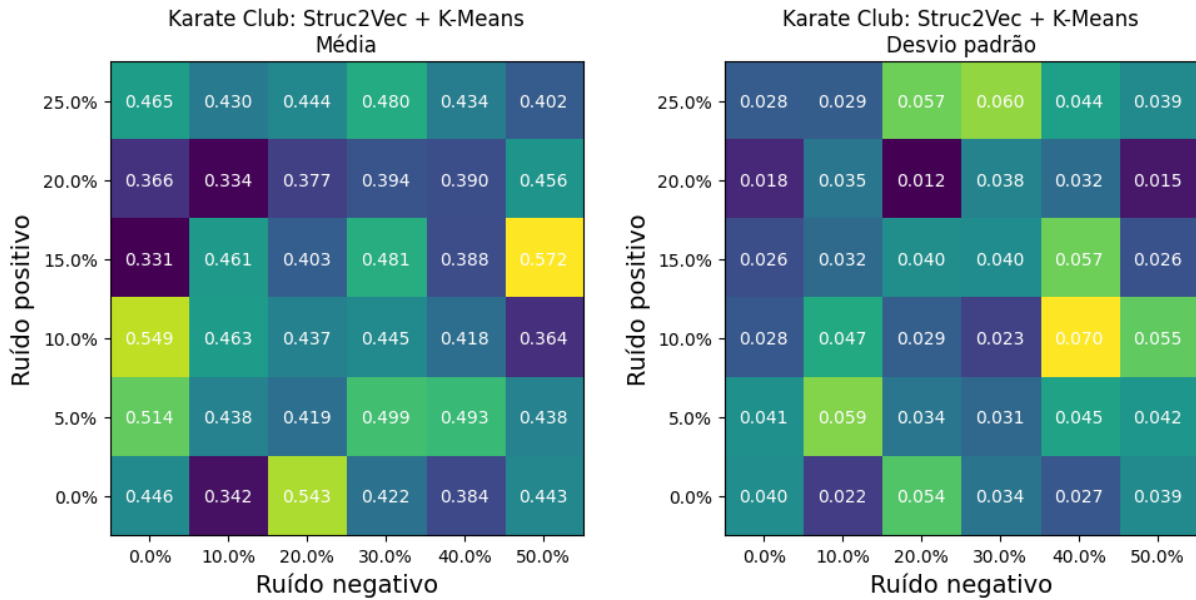
Figura 30 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

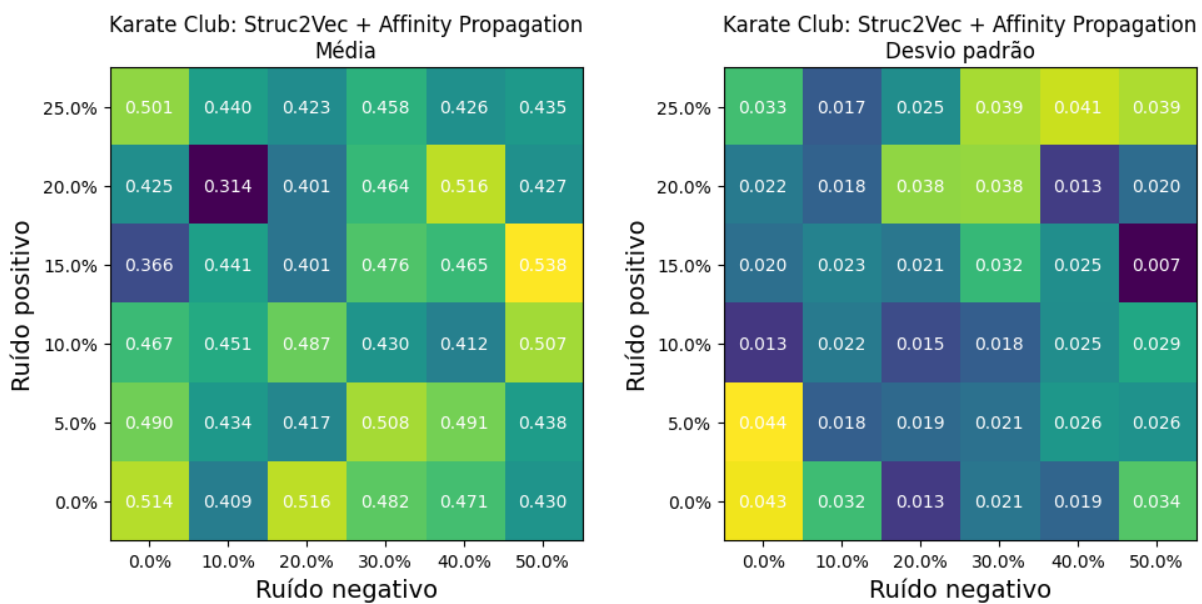
B.1.4 struc2vec

Figura 31 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e K-Means sobre Zachary's Karate Club com diferentes ruídos.



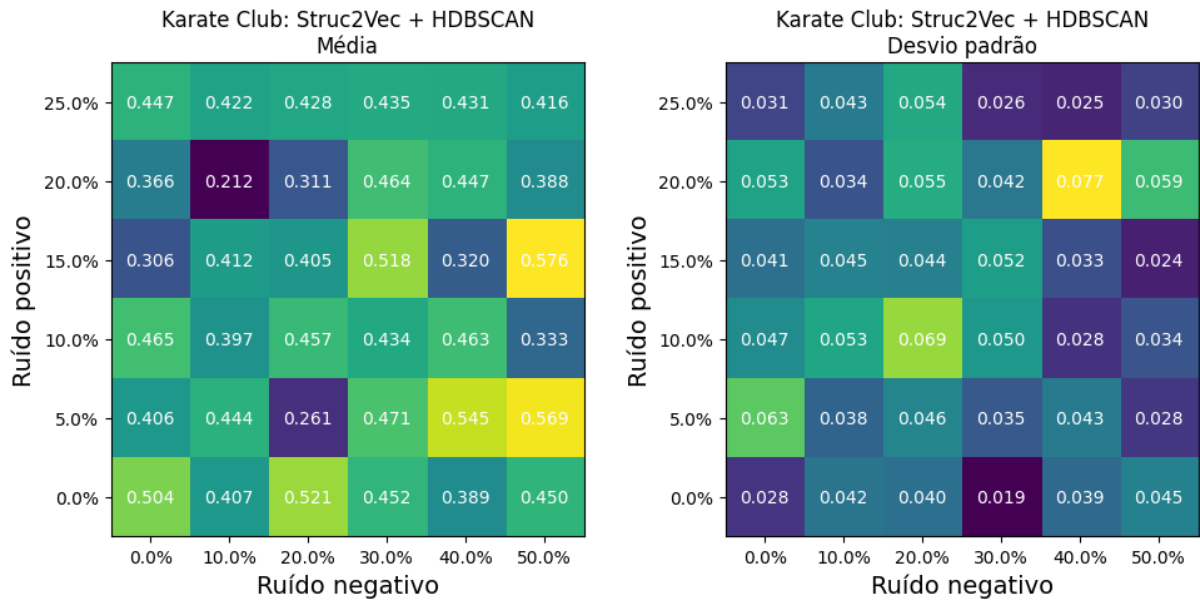
Fonte: Autor.

Figura 32 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e Affinity Propagation sobre Zachary's Karate Club com diferentes ruídos.



Fonte: Autor.

Figura 33 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e HDBSCAN sobre Zachary's Karate Club com diferentes ruídos.

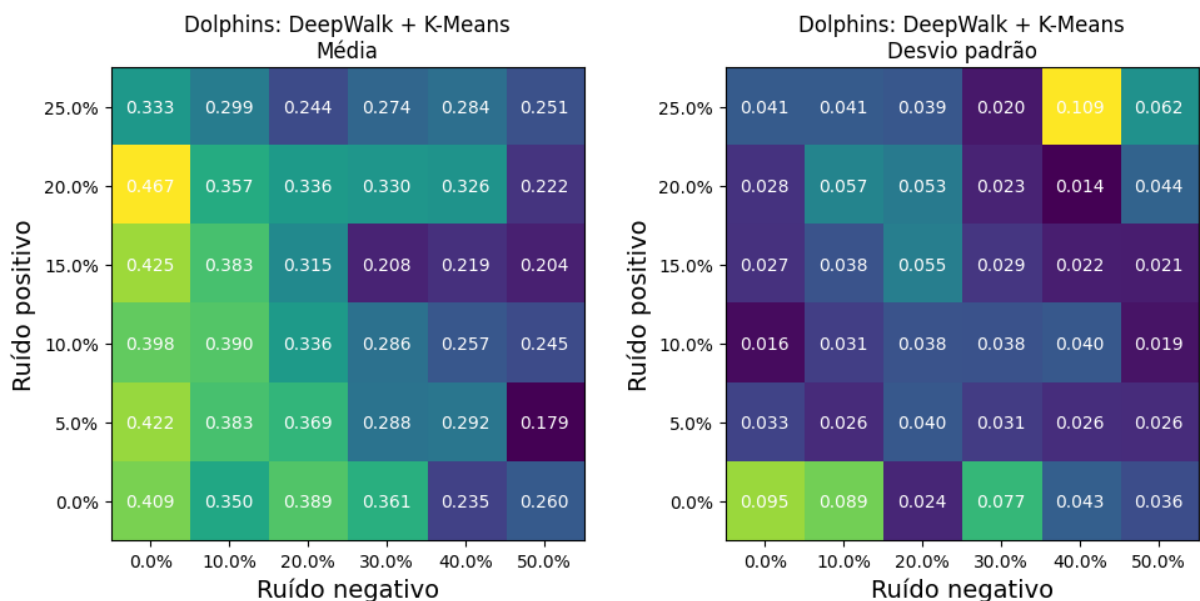


Fonte: Autor.

B.2 Dolphins

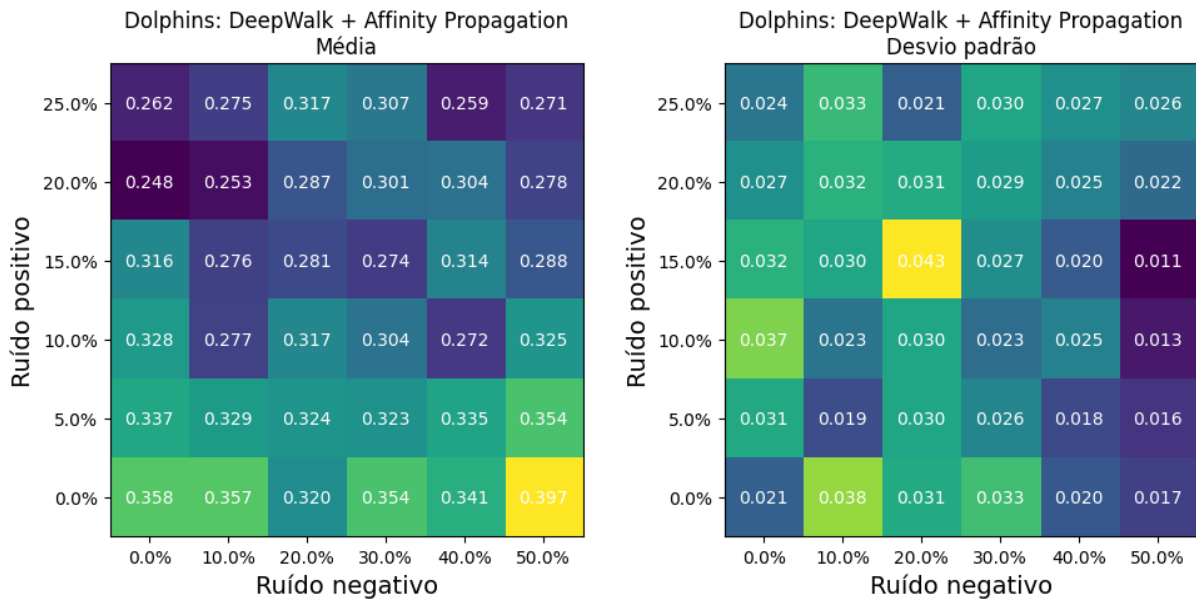
B.2.1 DeepWalk

Figura 34 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Dolphins com diferentes ruídos.



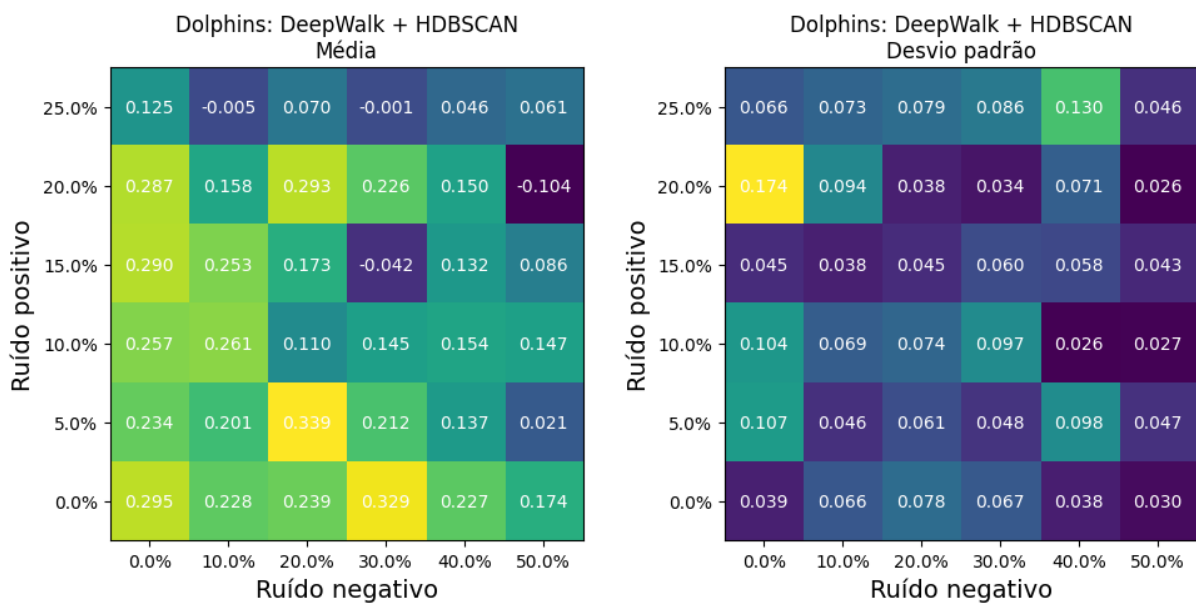
Fonte: Autor.

Figura 35 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Dolphins com diferentes ruídos.



Fonte: Autor.

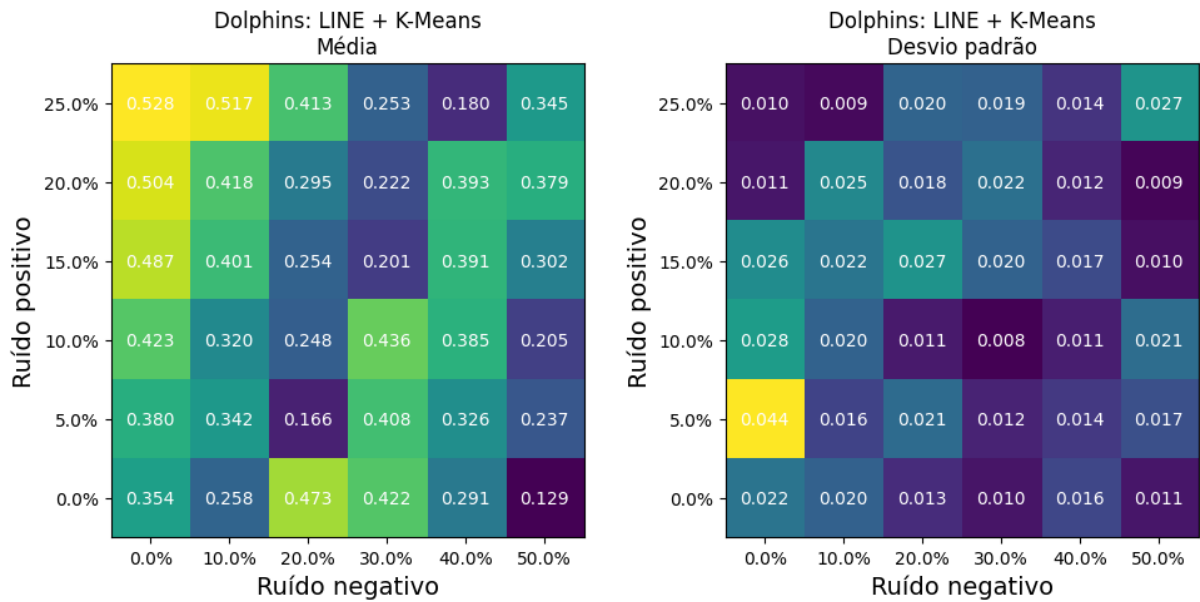
Figura 36 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Dolphins com diferentes ruídos.



Fonte: Autor.

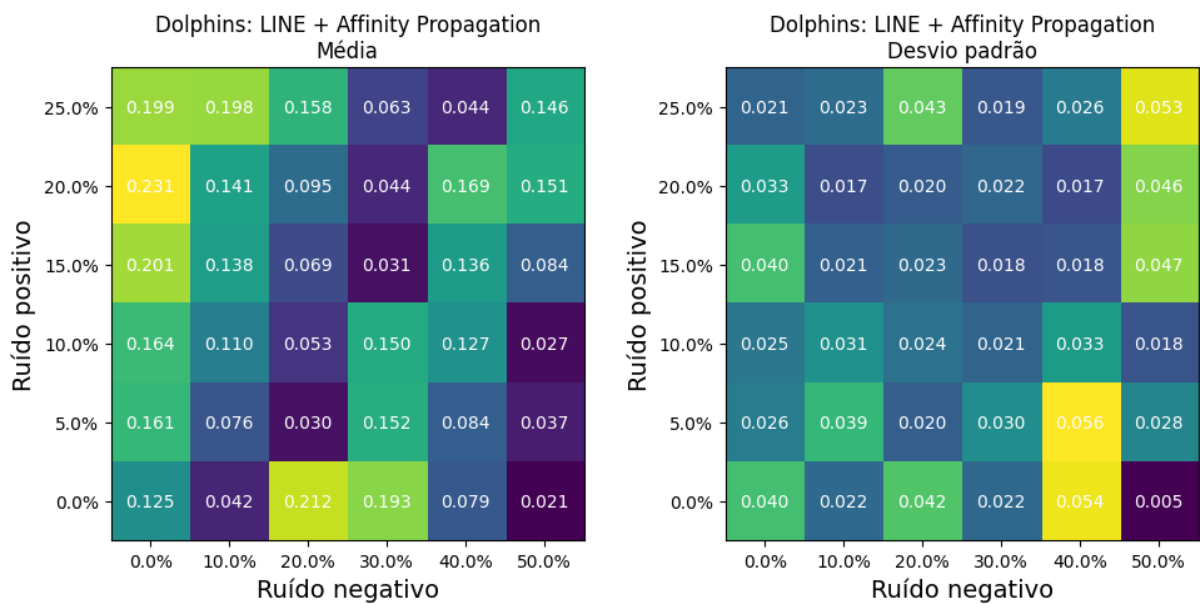
B.2.2 LINE

Figura 37 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Dolphins com diferentes ruídos.



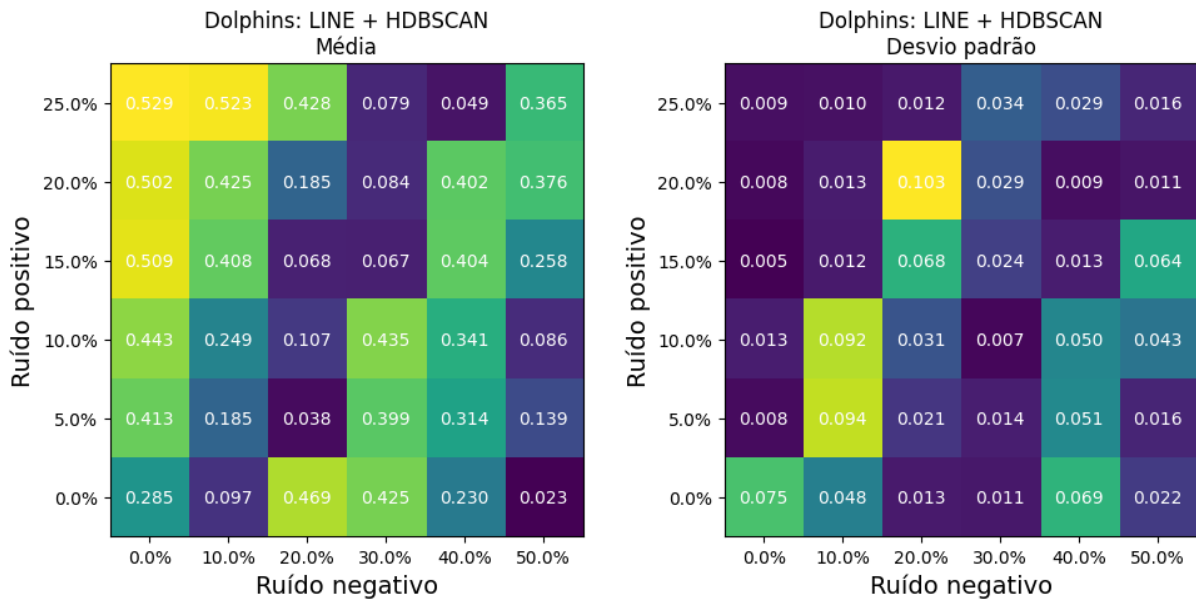
Fonte: Autor.

Figura 38 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Dolphins com diferentes ruídos.



Fonte: Autor.

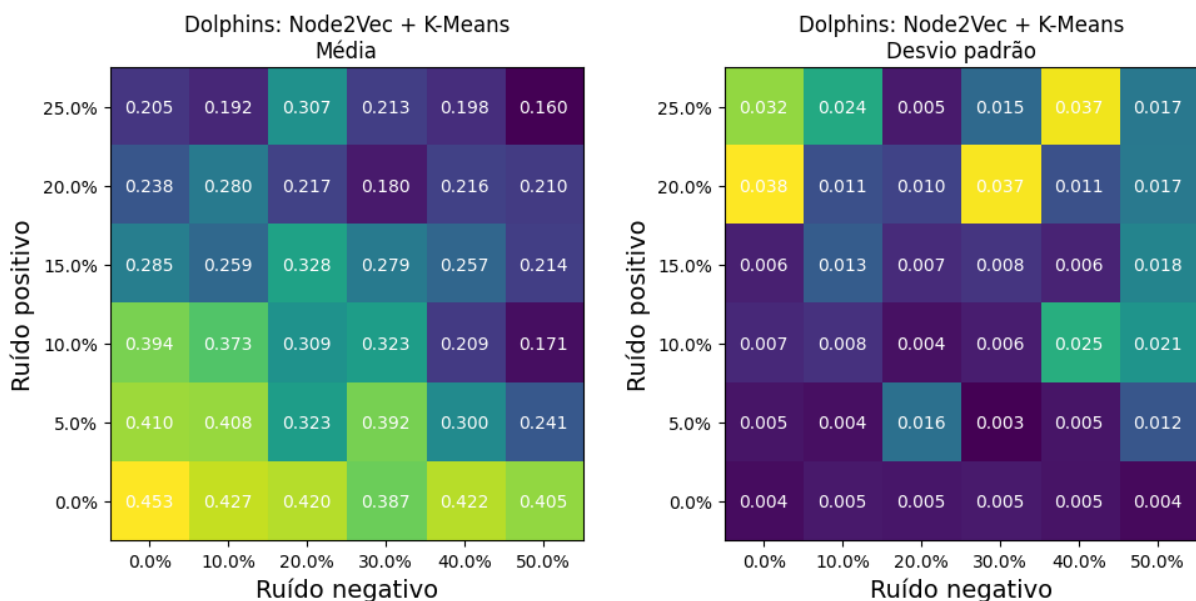
Figura 39 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Dolphins com diferentes ruídos.



Fonte: Autor.

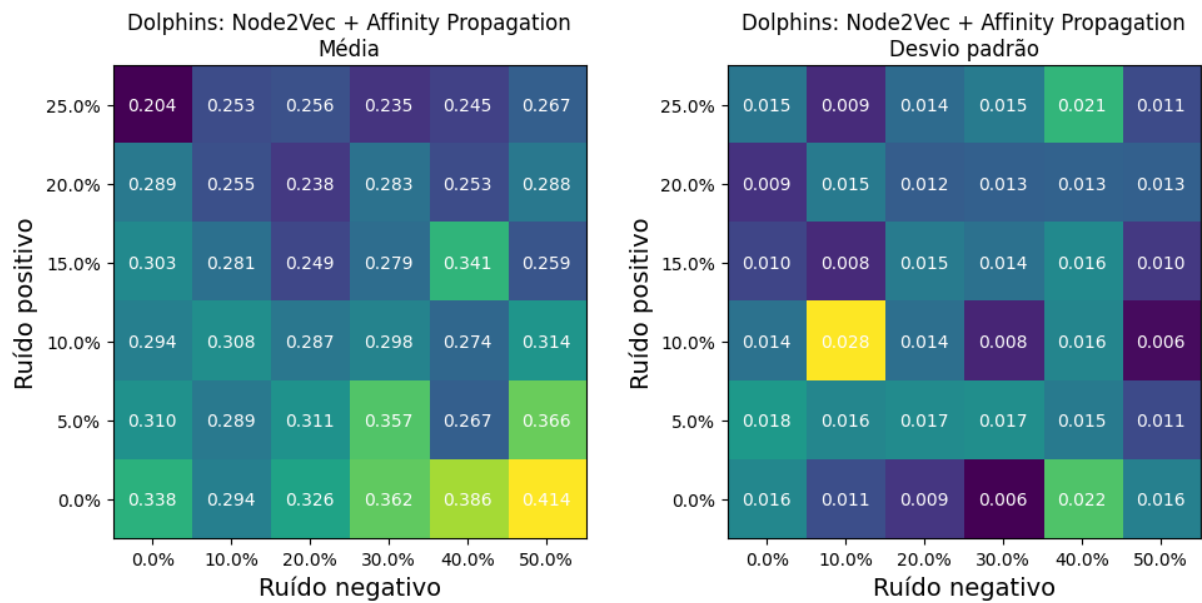
B.2.3 node2vec

Figura 40 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Dolphins com diferentes ruídos.



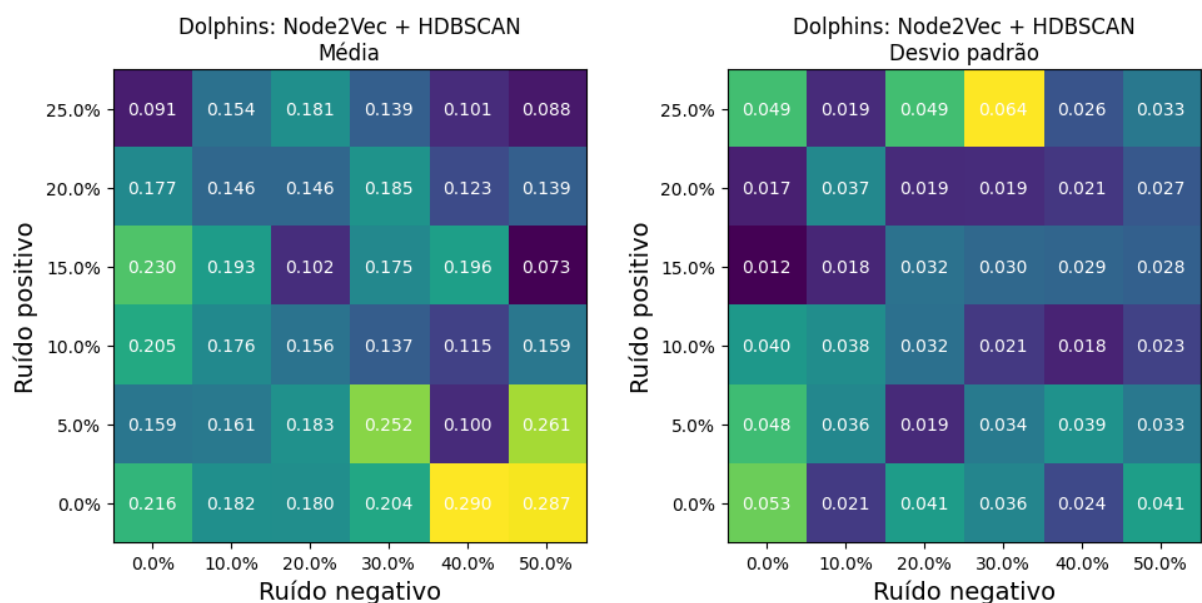
Fonte: Autor.

Figura 41 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Dolphins com diferentes ruídos.



Fonte: Autor.

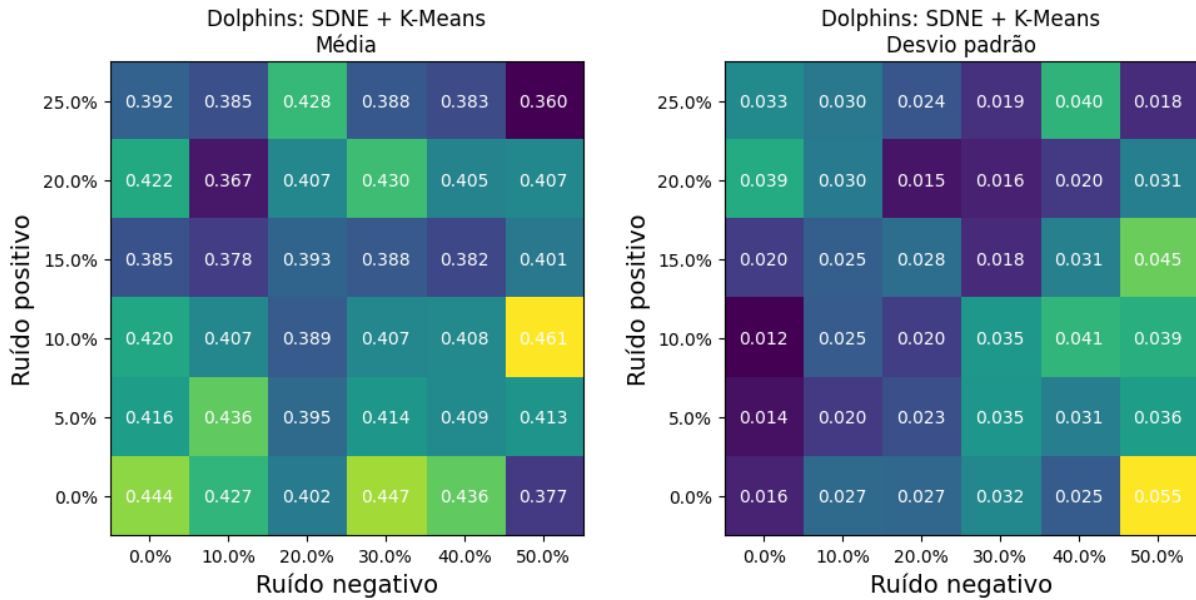
Figura 42 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Dolphins com diferentes ruídos.



Fonte: Autor.

B.2.4 SDNE

Figura 43 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Dolphins com diferentes ruídos.



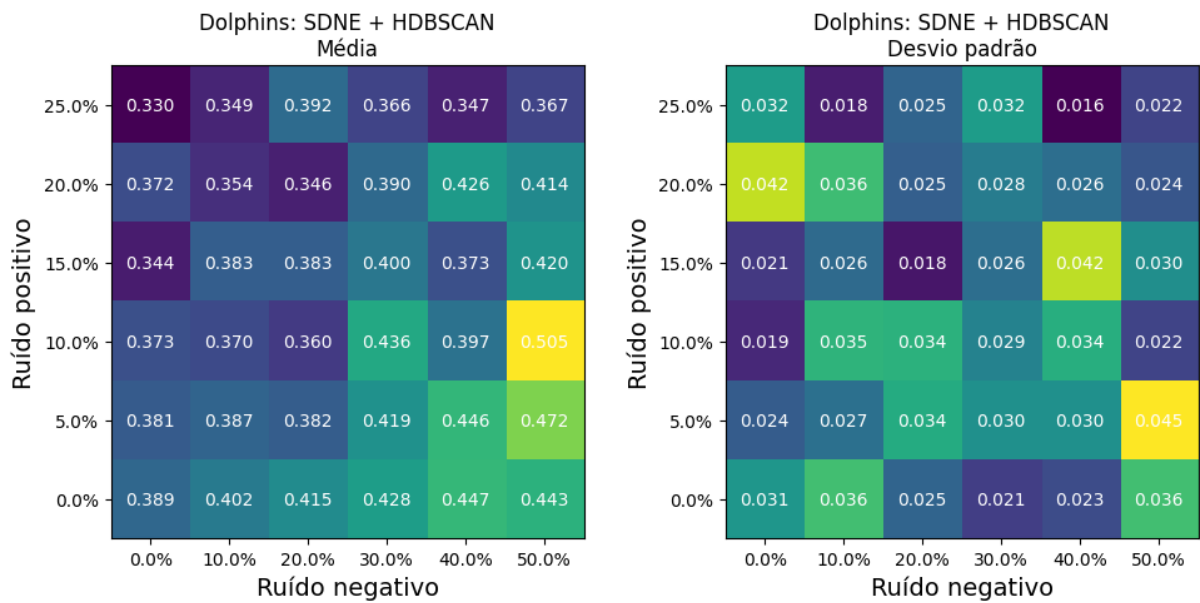
Fonte: Autor.

Figura 44 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Dolphins com diferentes ruídos.



Fonte: Autor.

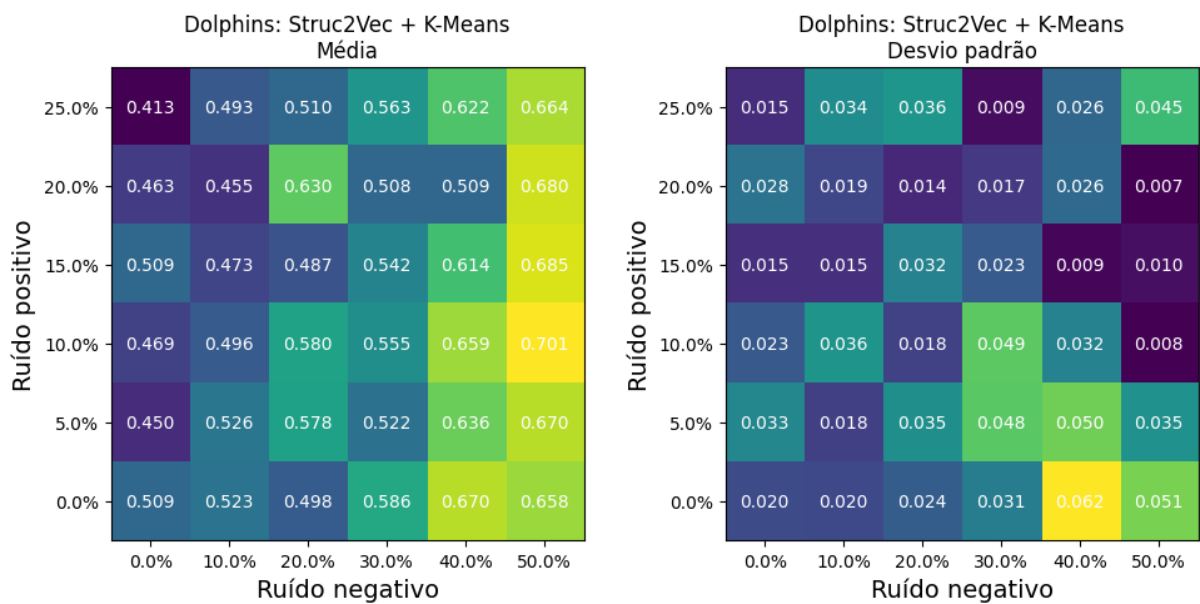
Figura 45 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Dolphins com diferentes ruídos.



Fonte: Autor.

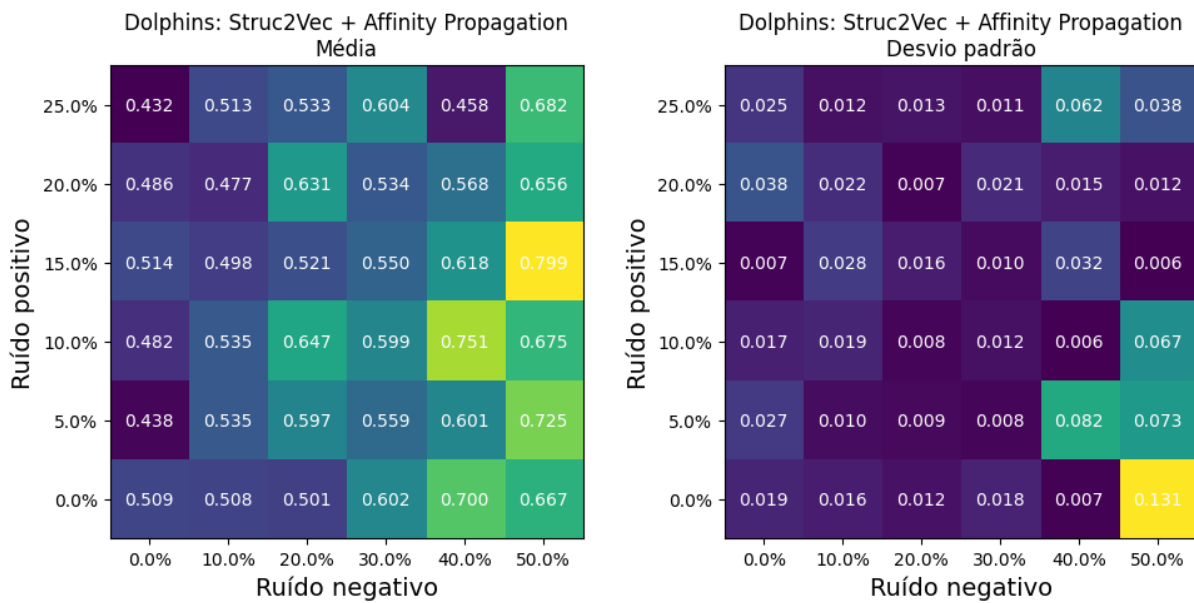
B.2.5 struc2vec

Figura 46 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e K-Means sobre Dolphins com diferentes ruídos.



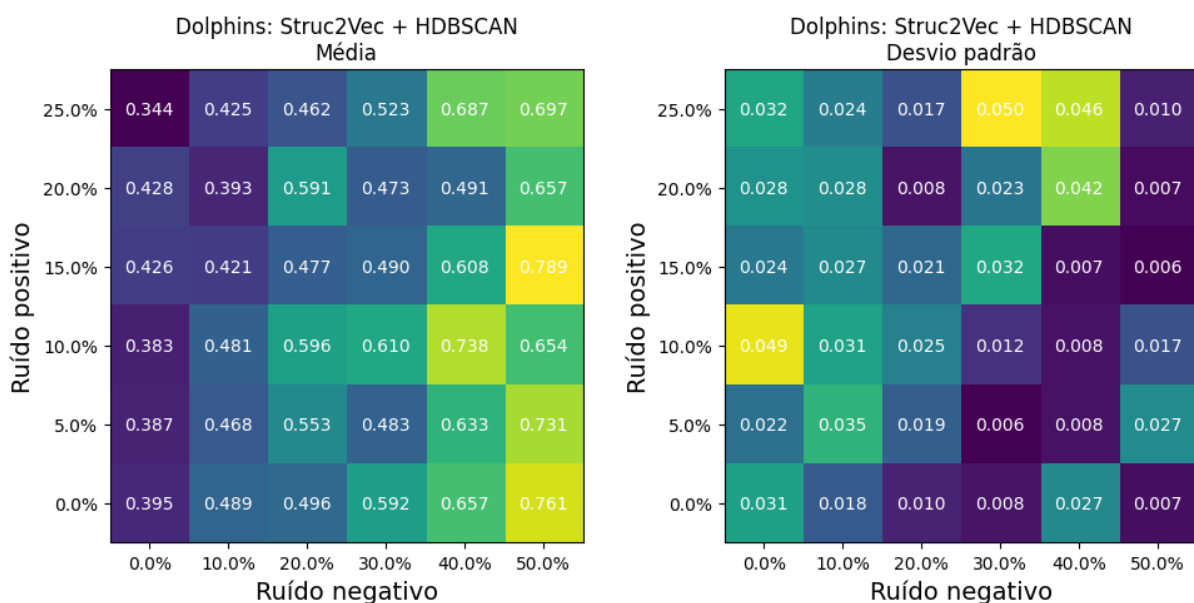
Fonte: Autor.

Figura 47 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e Affinity Propagation sobre Dolphins com diferentes ruídos.



Fonte: Autor.

Figura 48 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e HDBSCAN sobre Dolphins com diferentes ruídos.

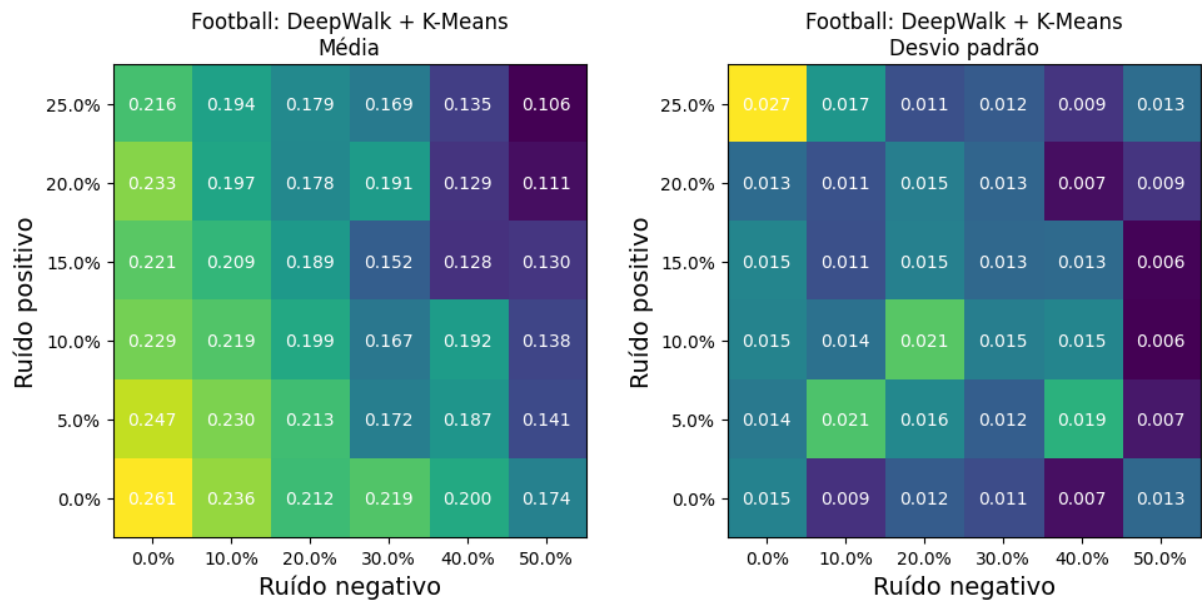


Fonte: Autor.

B.3 Football

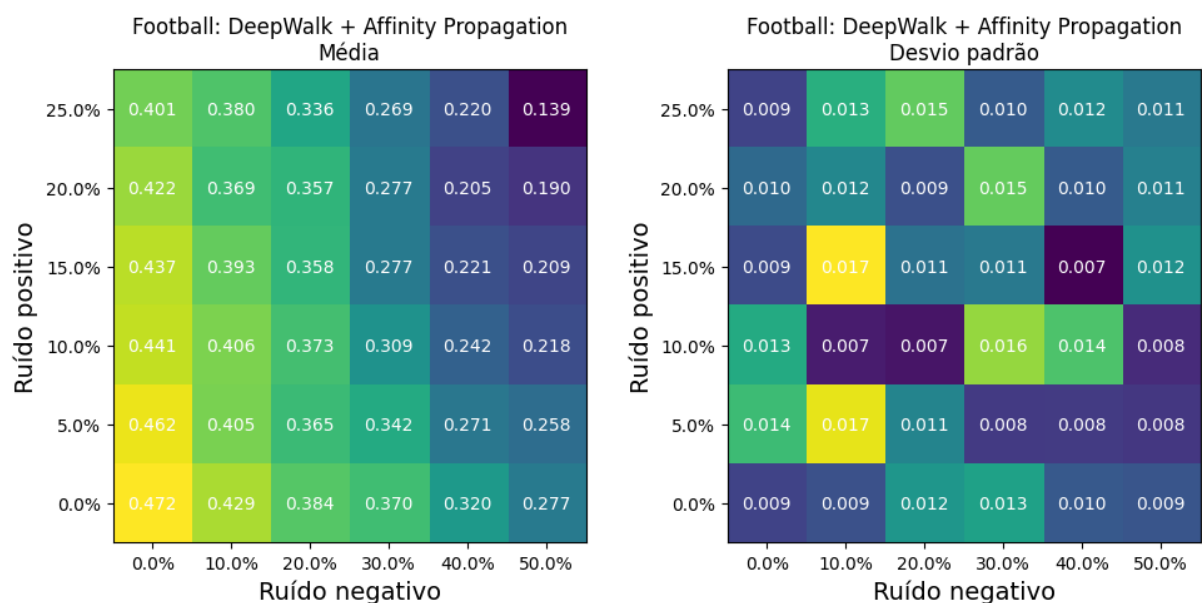
B.3.1 DeepWalk

Figura 49 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Football com diferentes ruídos.



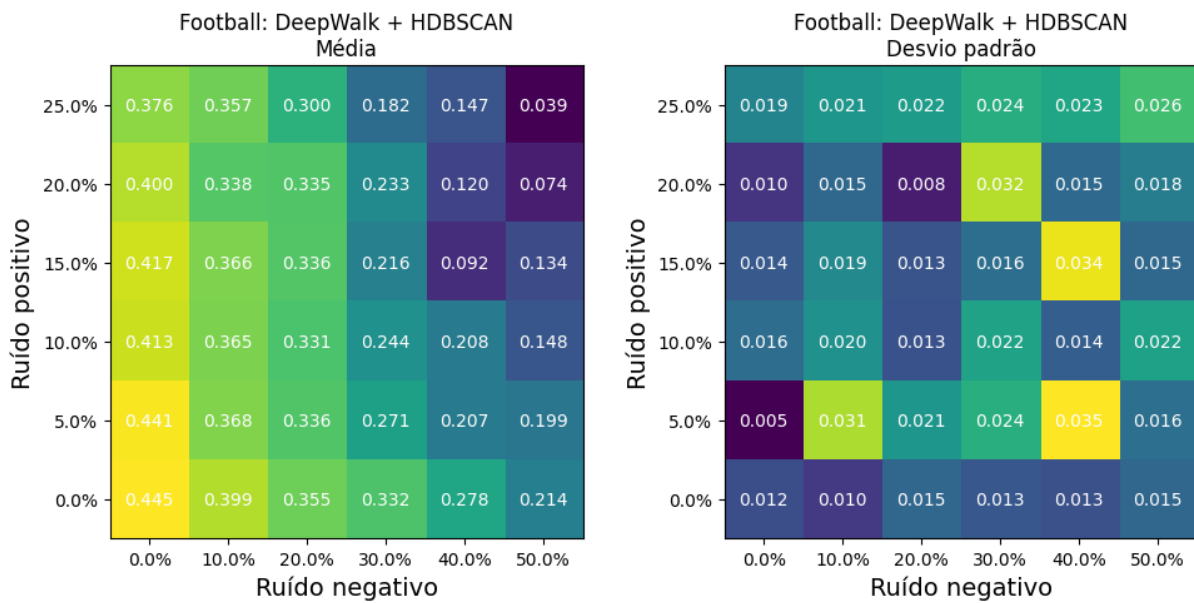
Fonte: Autor.

Figura 50 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Football com diferentes ruídos.



Fonte: Autor.

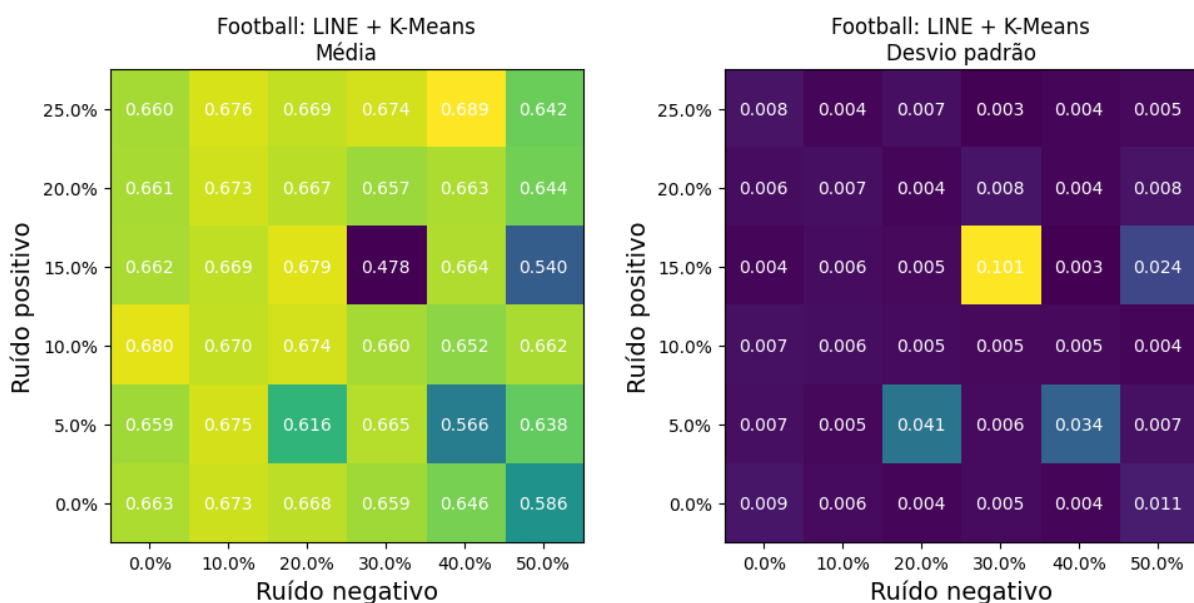
Figura 51 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Football com diferentes ruídos.



Fonte: Autor.

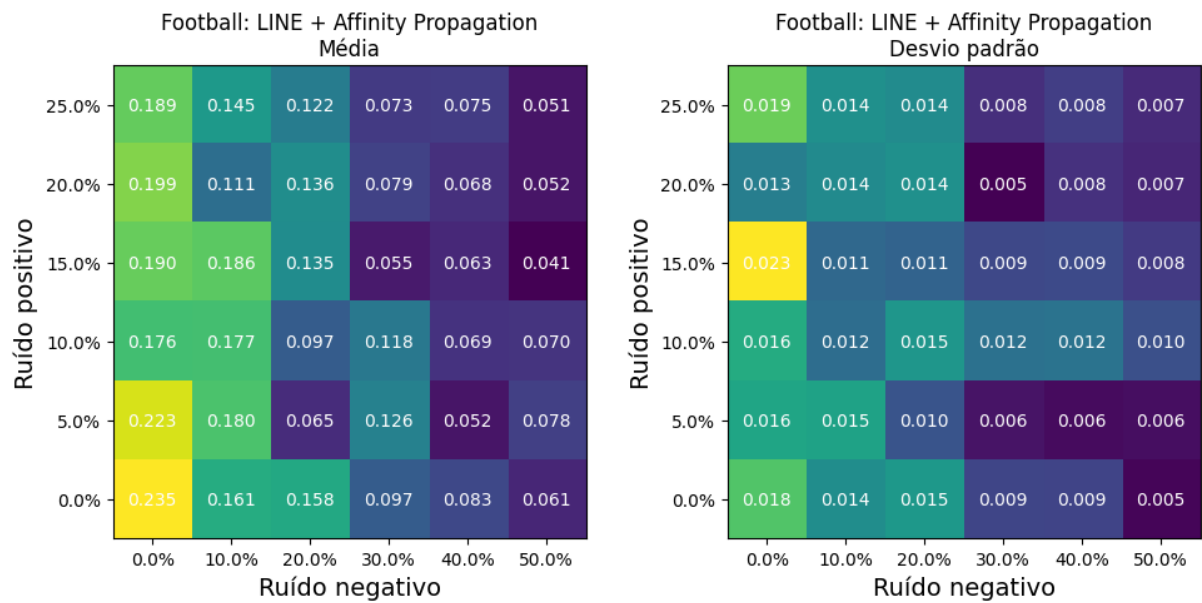
B.3.2 LINE

Figura 52 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Football com diferentes ruídos.



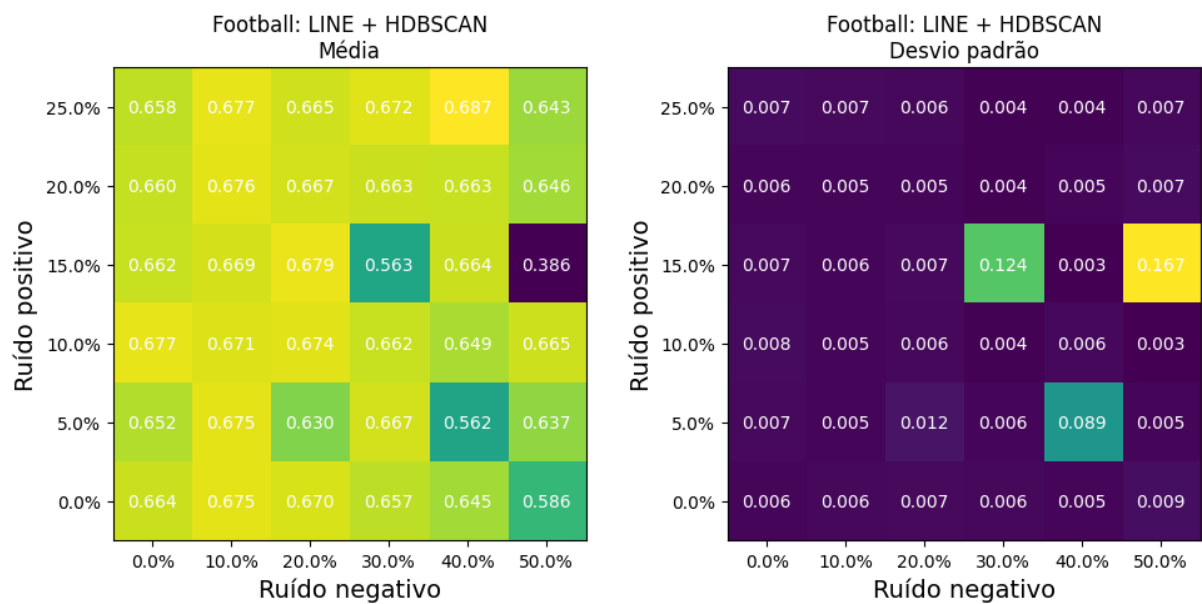
Fonte: Autor.

Figura 53 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Football com diferentes ruídos.



Fonte: Autor.

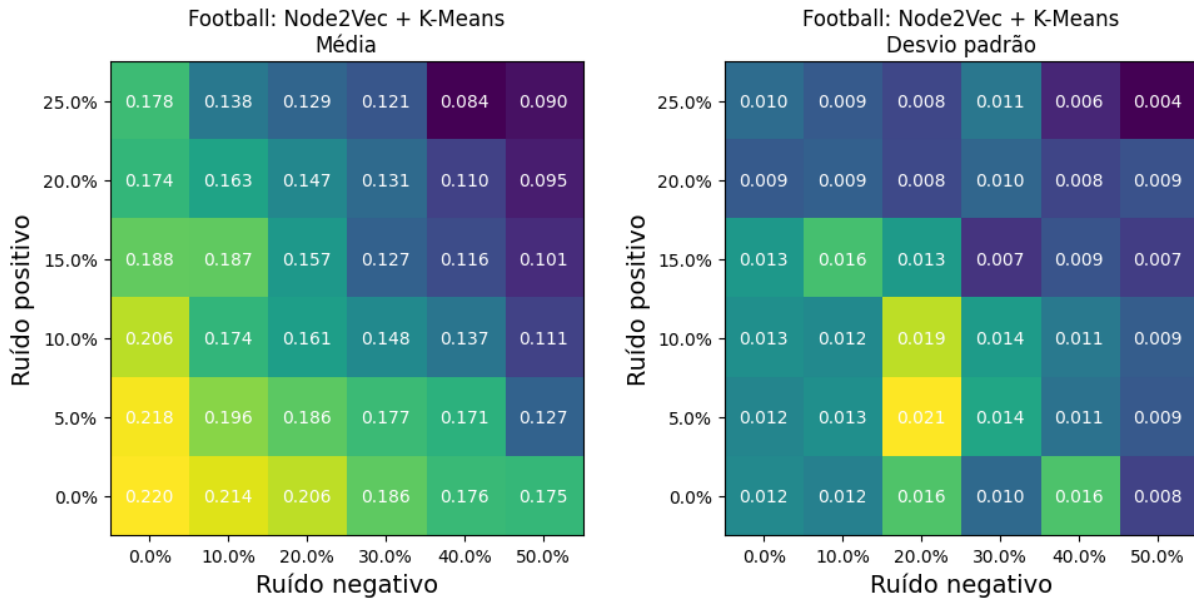
Figura 54 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Football com diferentes ruídos.



Fonte: Autor.

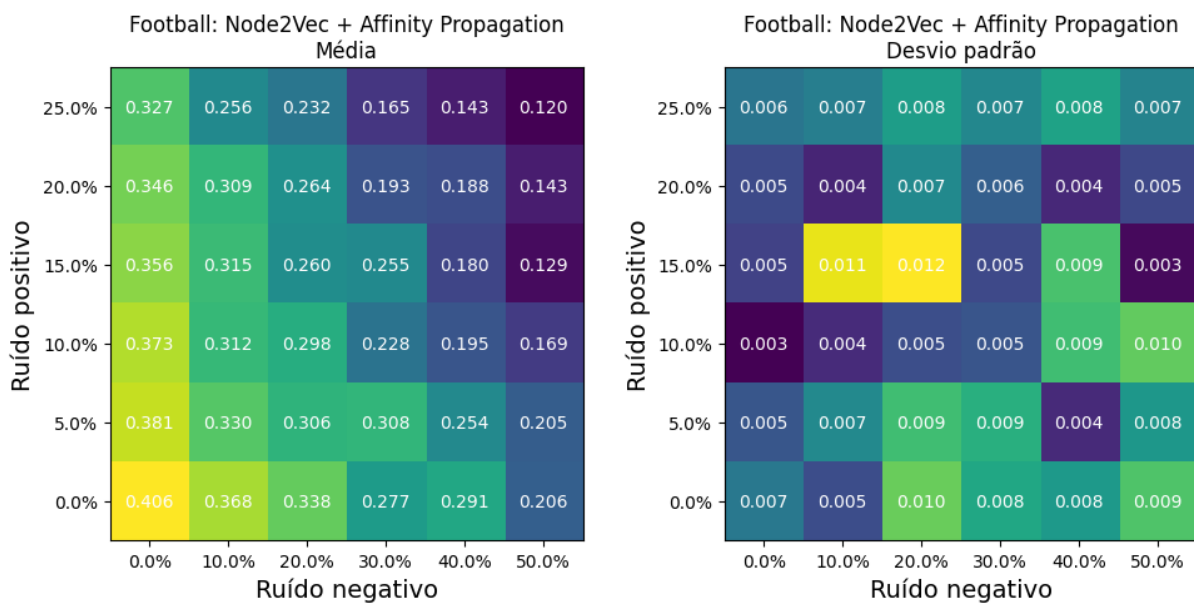
B.3.3 node2vec

Figura 55 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Football com diferentes ruídos.



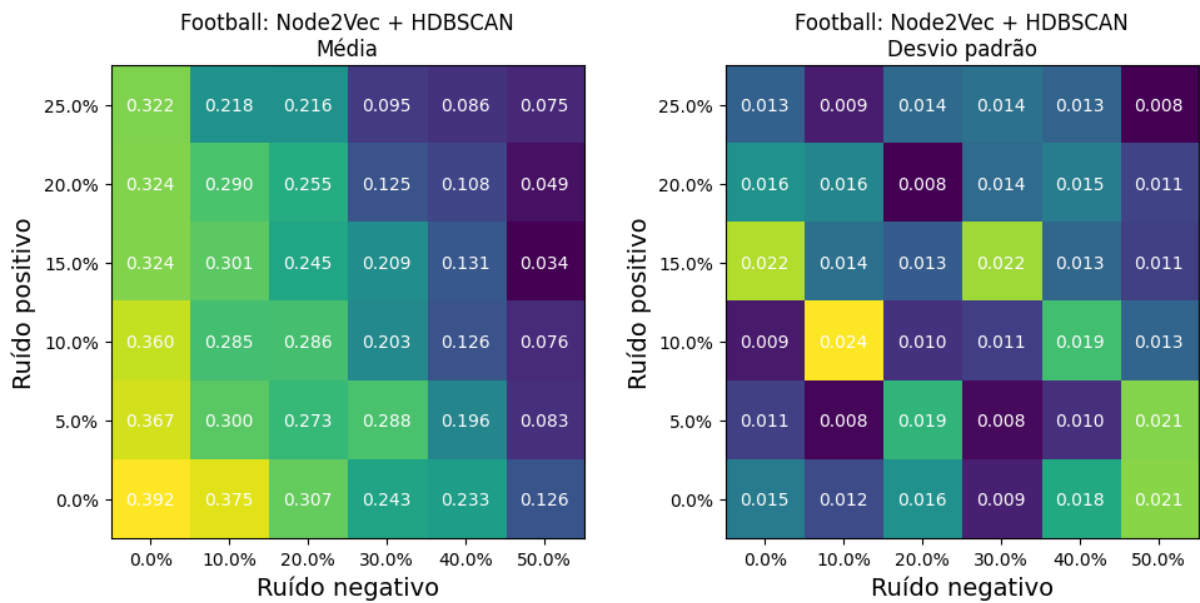
Fonte: Autor.

Figura 56 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Football com diferentes ruídos.



Fonte: Autor.

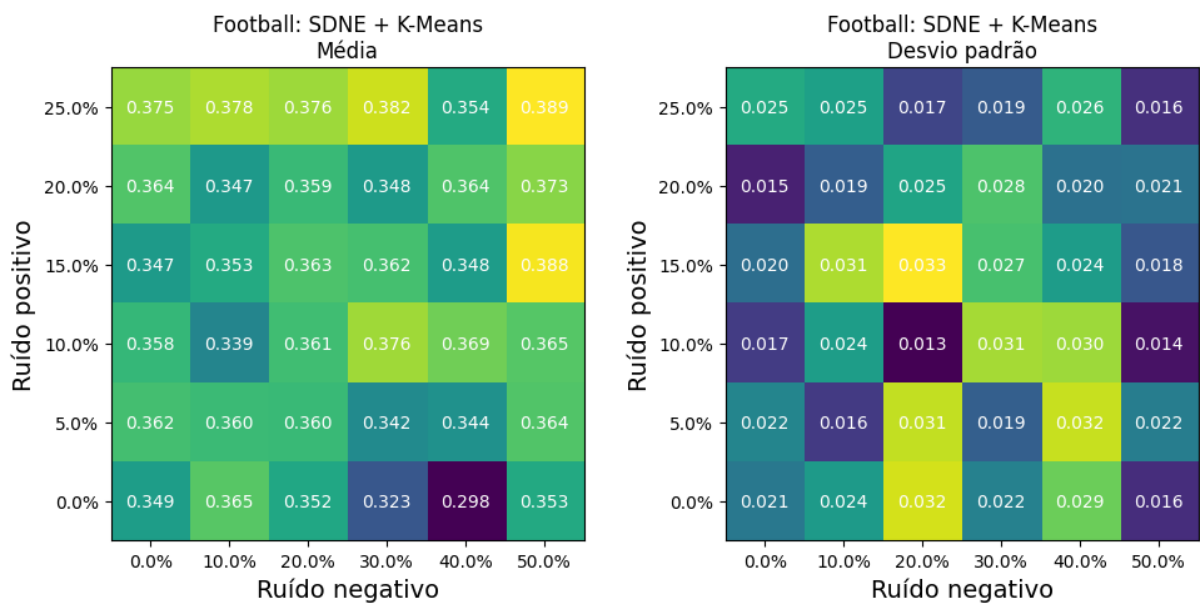
Figura 57 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Football com diferentes ruídos.



Fonte: Autor.

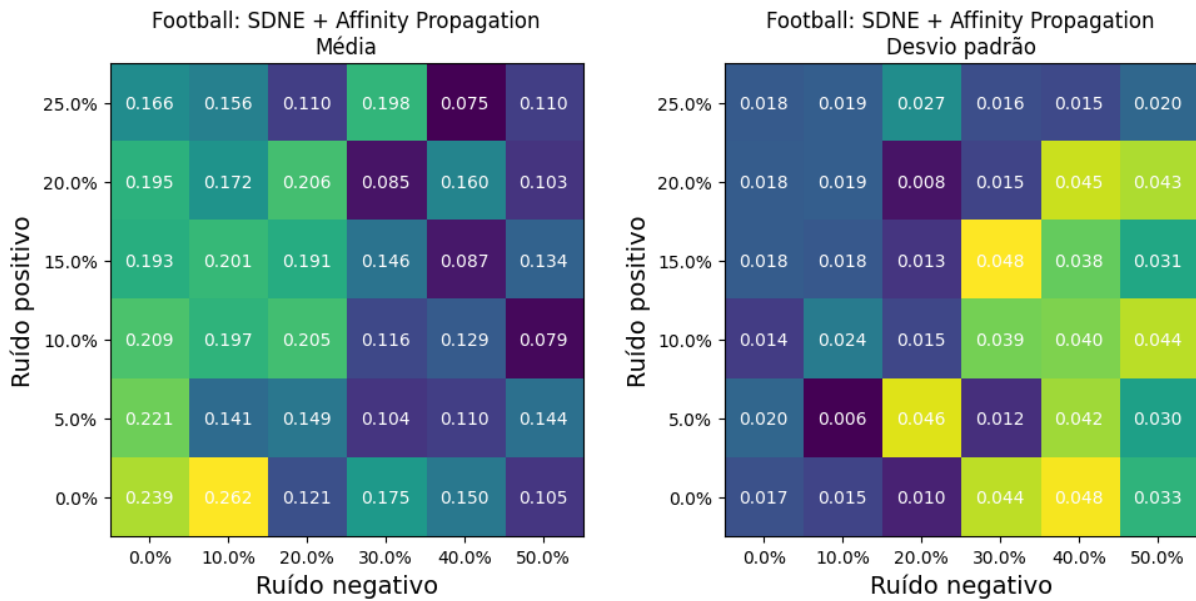
B.3.4 SDNE

Figura 58 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Football com diferentes ruídos.



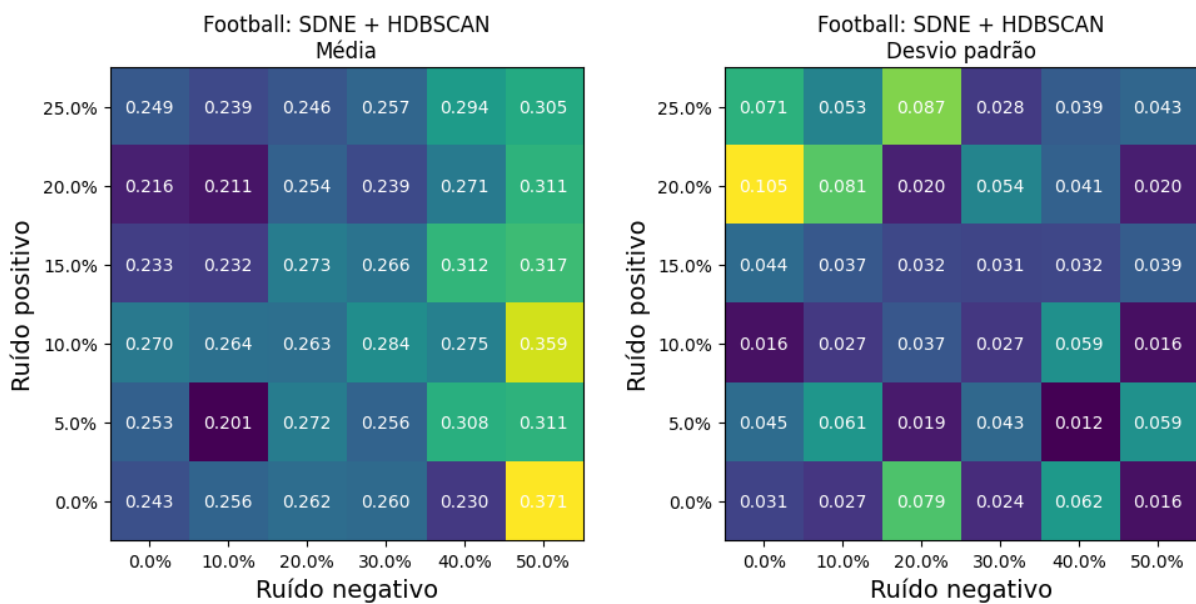
Fonte: Autor.

Figura 59 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Football com diferentes ruídos.



Fonte: Autor.

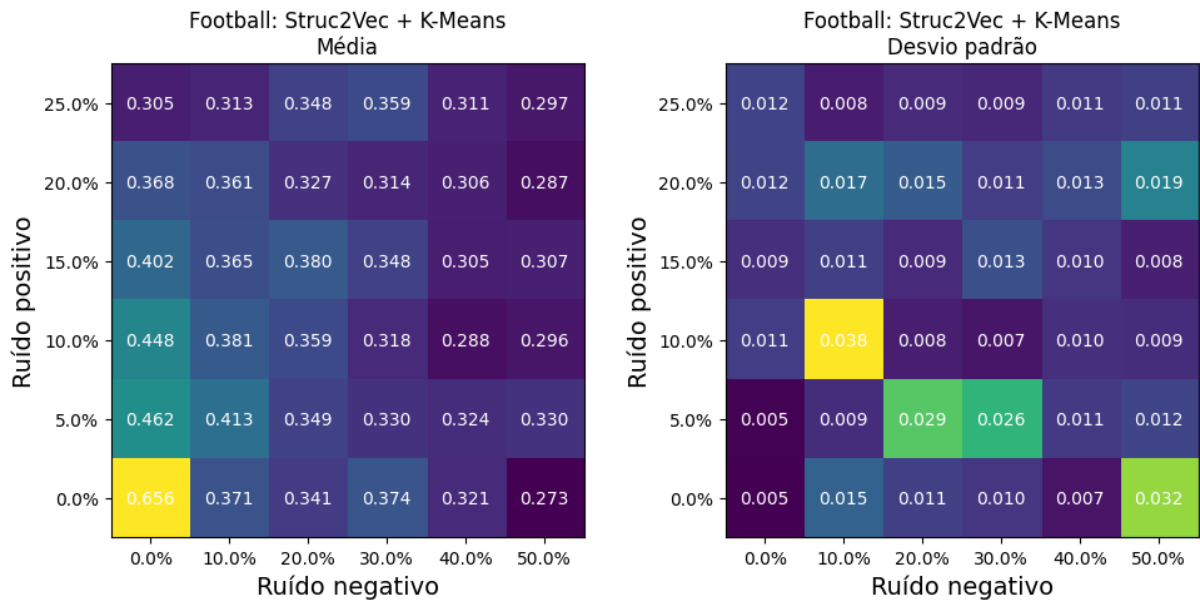
Figura 60 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Football com diferentes ruídos.



Fonte: Autor.

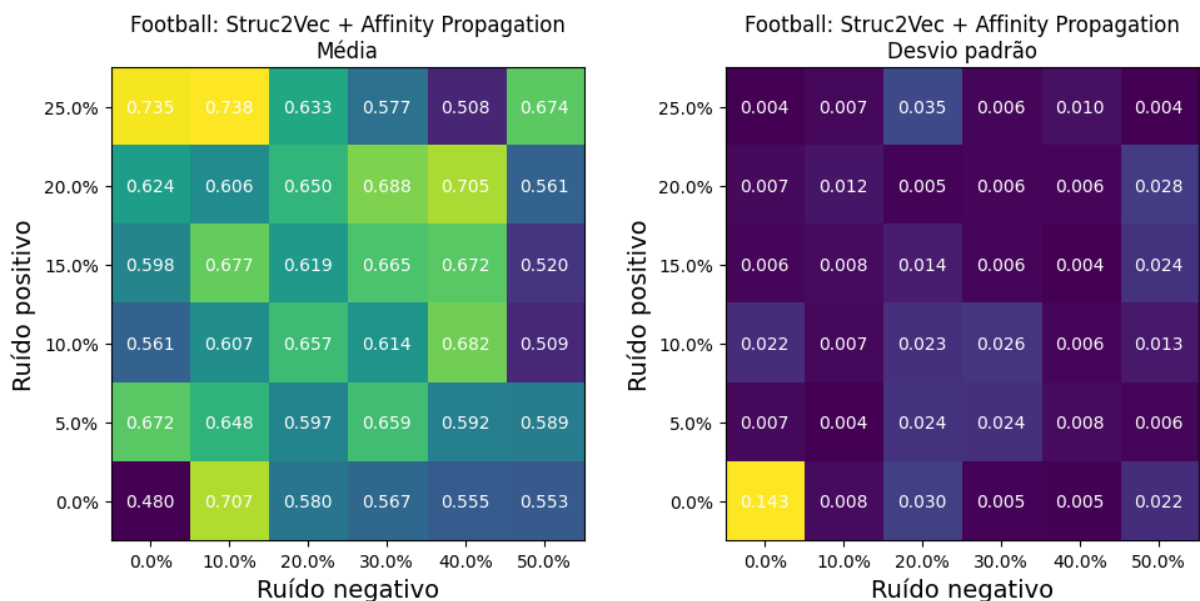
B.3.5 struc2vec

Figura 61 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e K-Means sobre Football com diferentes ruídos.



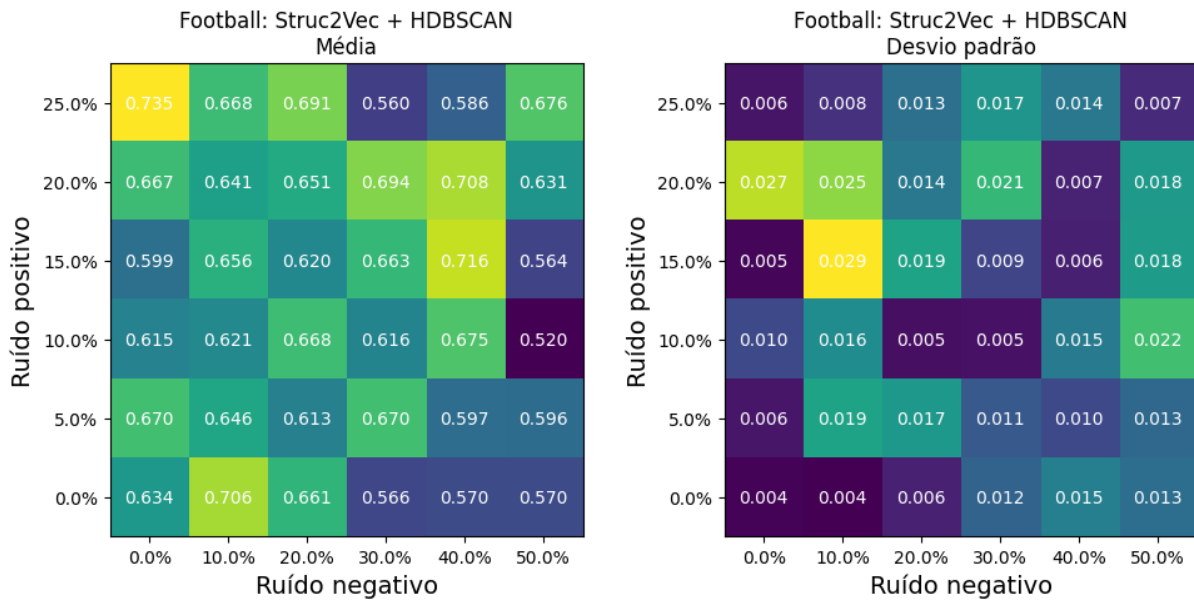
Fonte: Autor.

Figura 62 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e Affinity Propagation sobre Football com diferentes ruídos.



Fonte: Autor.

Figura 63 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por struc2vec e HDBSCAN sobre Football com diferentes ruídos.

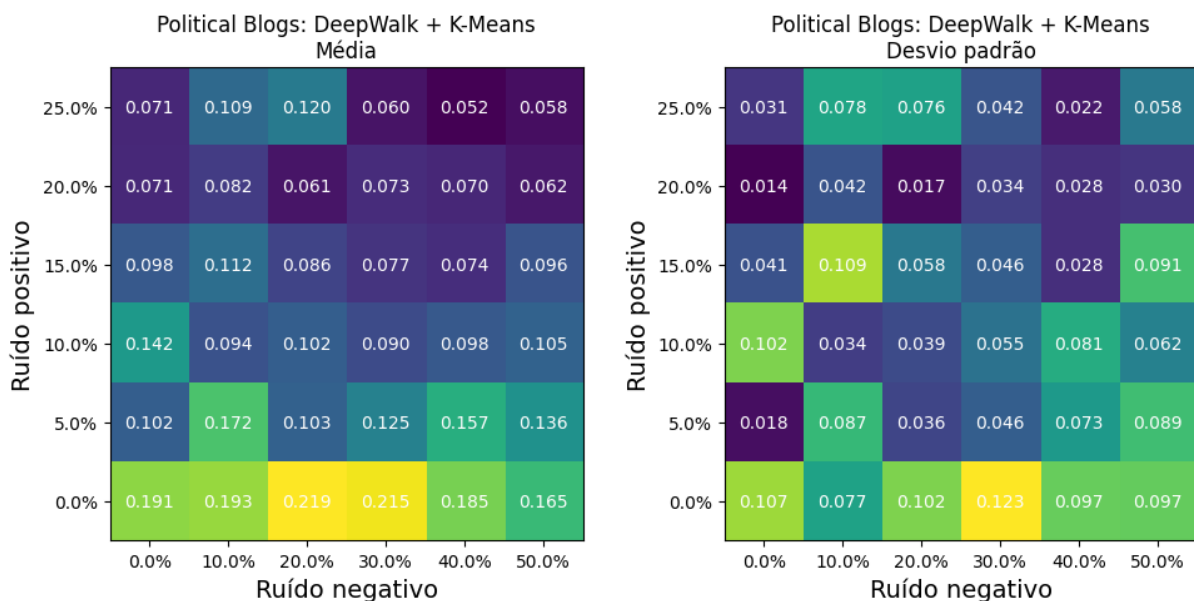


Fonte: Autor.

B.4 Political Blogs

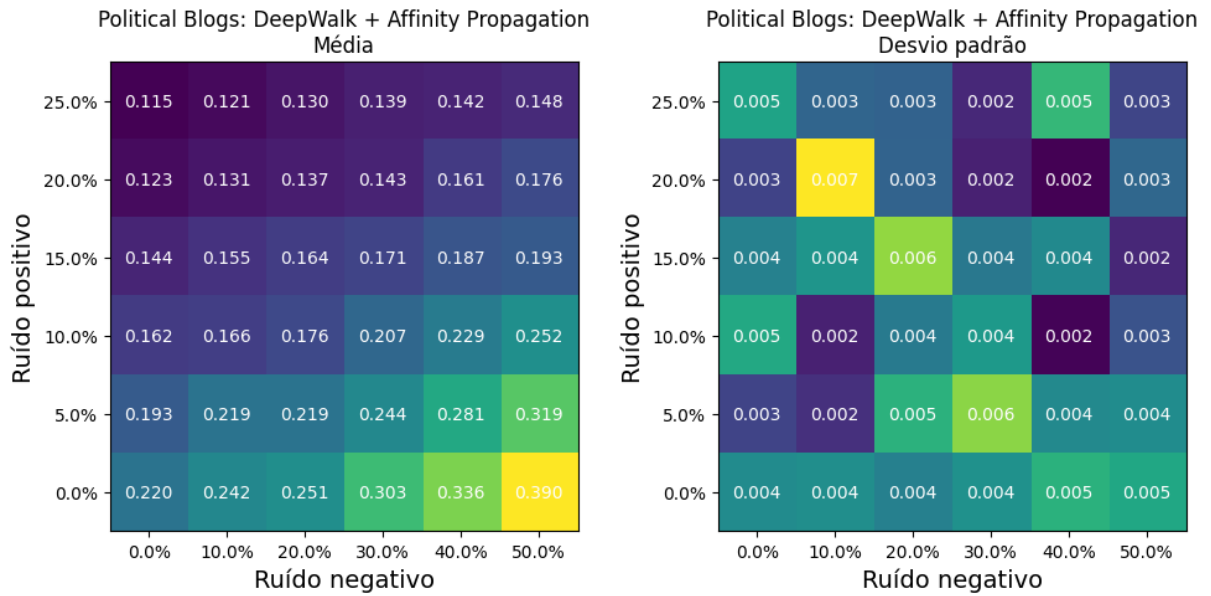
B.4.1 DeepWalk

Figura 64 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e K-Means sobre Political Blogs com diferentes ruídos.



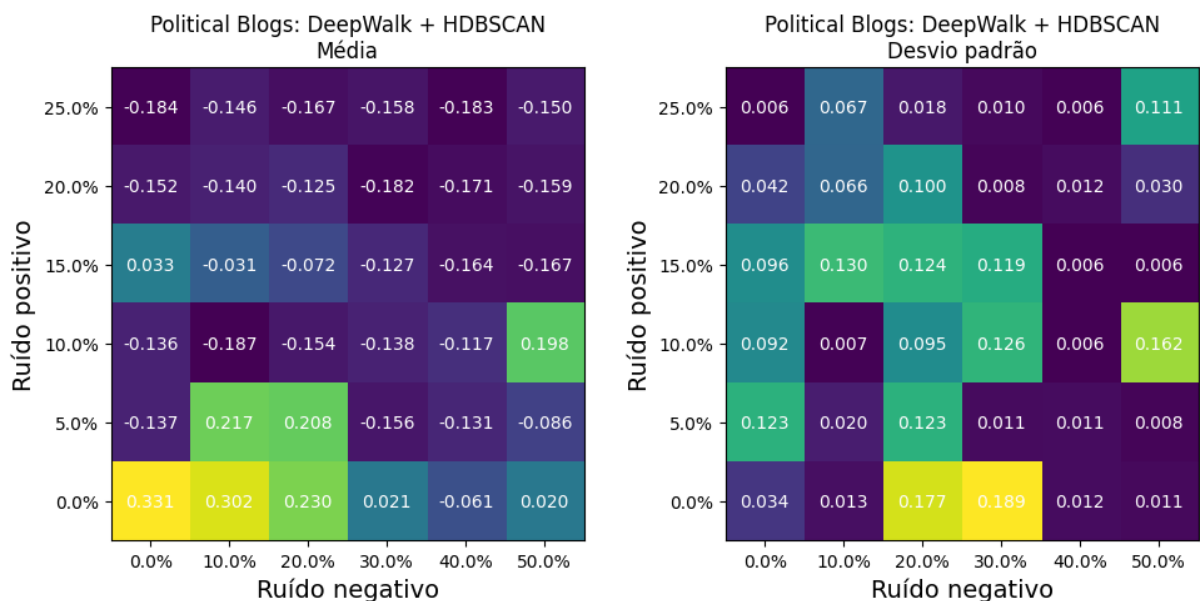
Fonte: Autor.

Figura 65 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e Affinity Propagation sobre Political Blogs com diferentes ruídos.



Fonte: Autor.

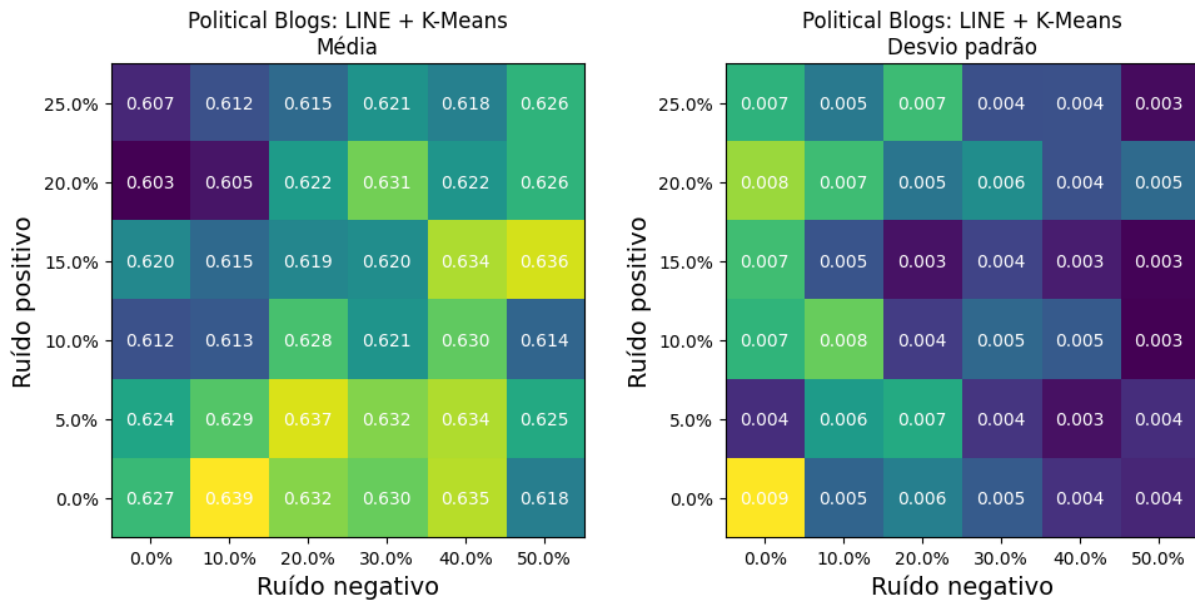
Figura 66 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por DeepWalk e HDBSCAN sobre Political Blogs com diferentes ruídos.



Fonte: Autor.

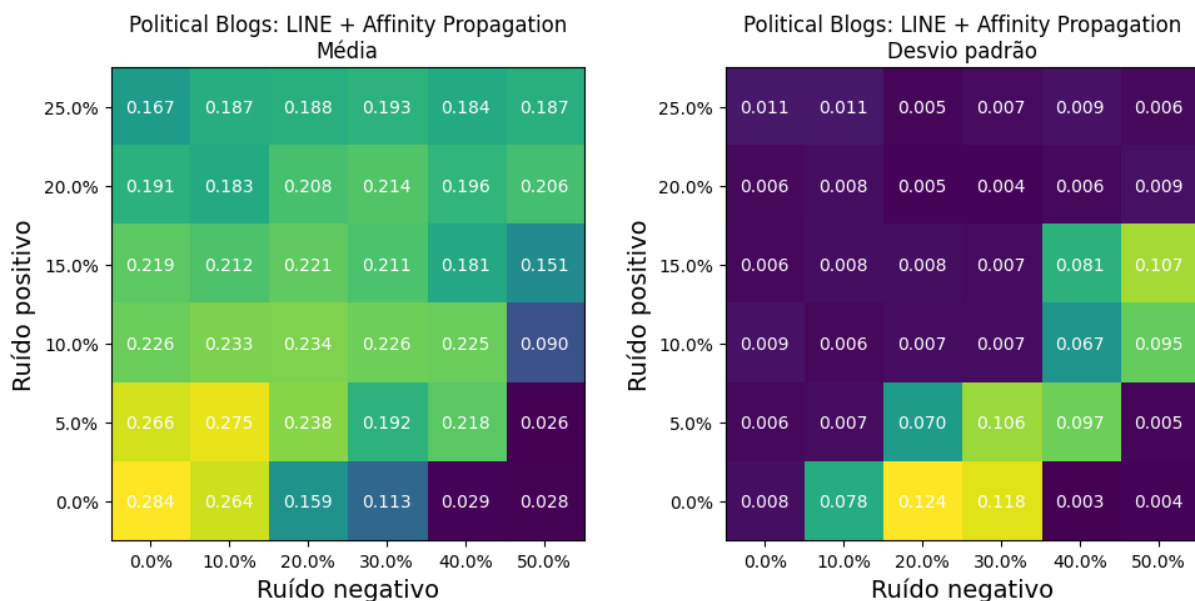
B.4.2 LINE

Figura 67 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e K-Means sobre Political Blogs com diferentes ruídos.



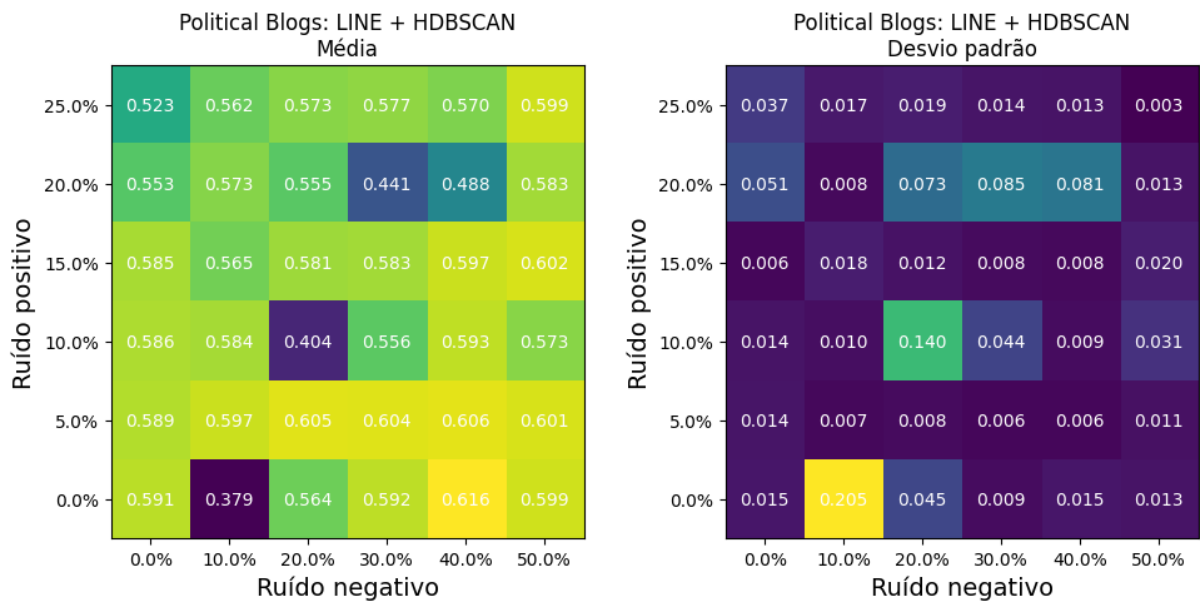
Fonte: Autor.

Figura 68 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e Affinity Propagation sobre Political Blogs com diferentes ruídos.



Fonte: Autor.

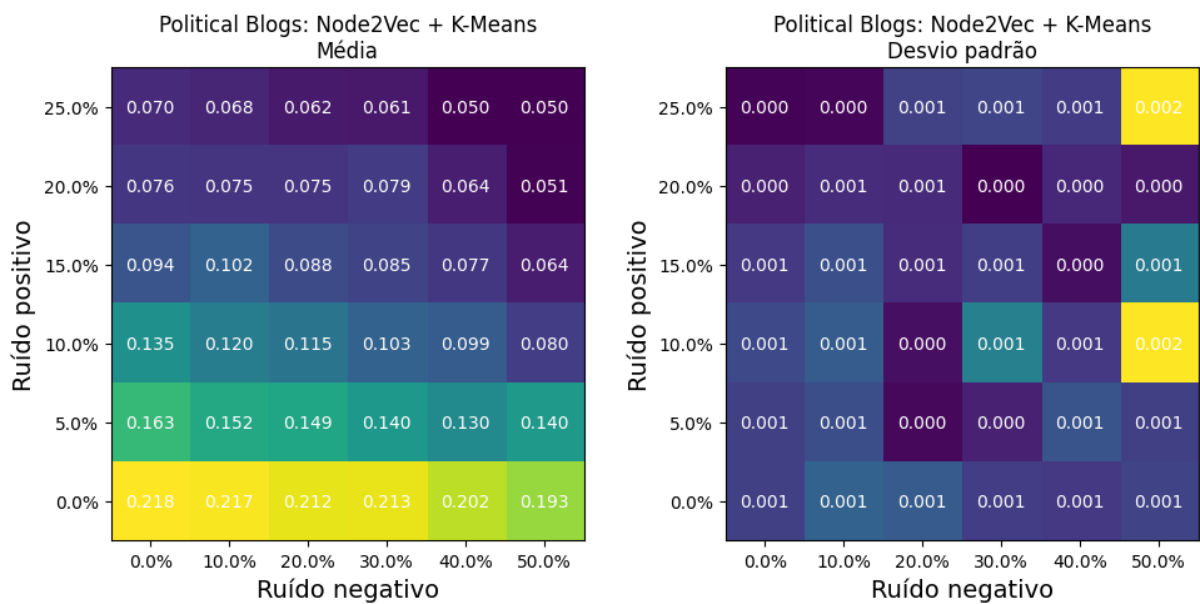
Figura 69 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por LINE e HDBSCAN sobre Political Blogs com diferentes ruídos.



Fonte: Autor.

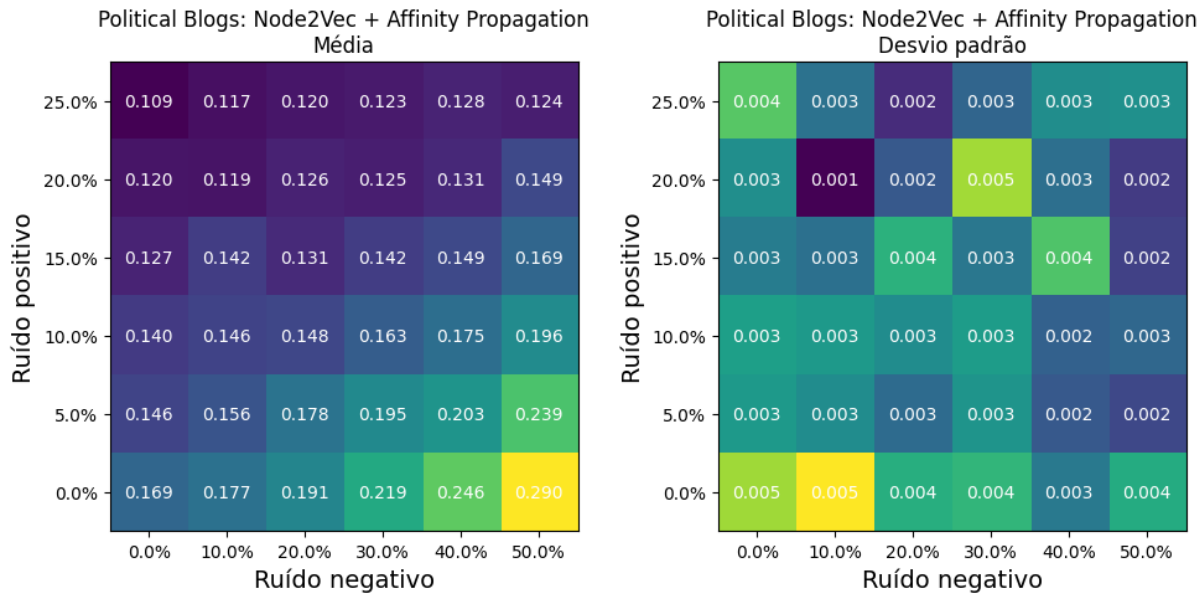
B.4.3 node2vec

Figura 70 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e K-Means sobre Political Blogs com diferentes ruídos.



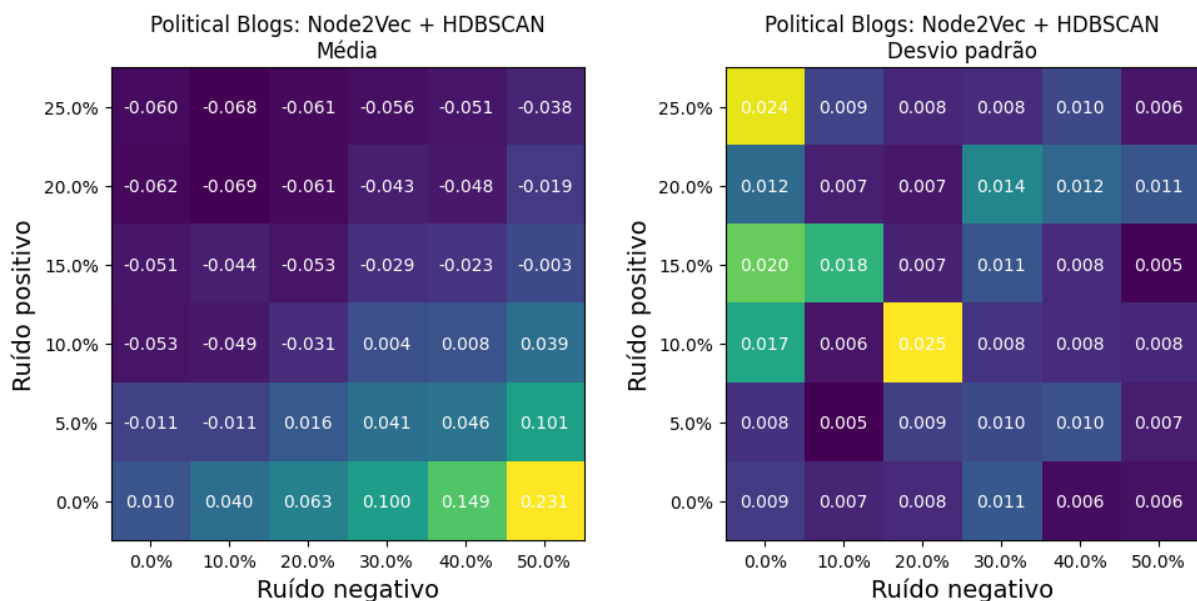
Fonte: Autor.

Figura 71 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e Affinity Propagation sobre Political Blogs com diferentes ruídos.



Fonte: Autor.

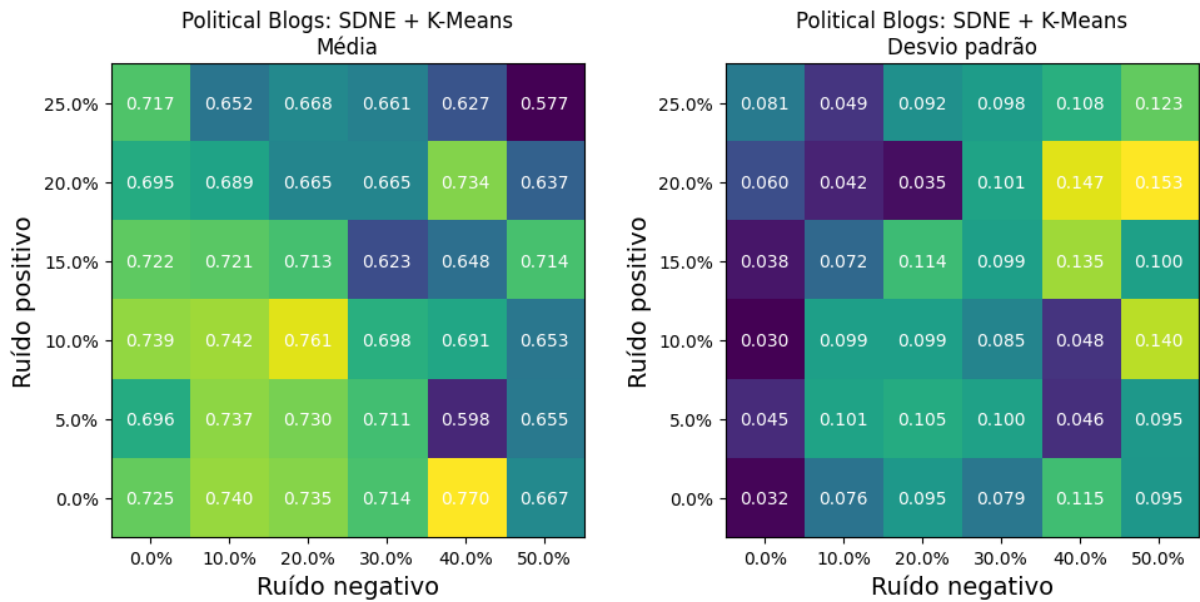
Figura 72 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por node2vec e HDBSCAN sobre Political Blogs com diferentes ruídos.



Fonte: Autor.

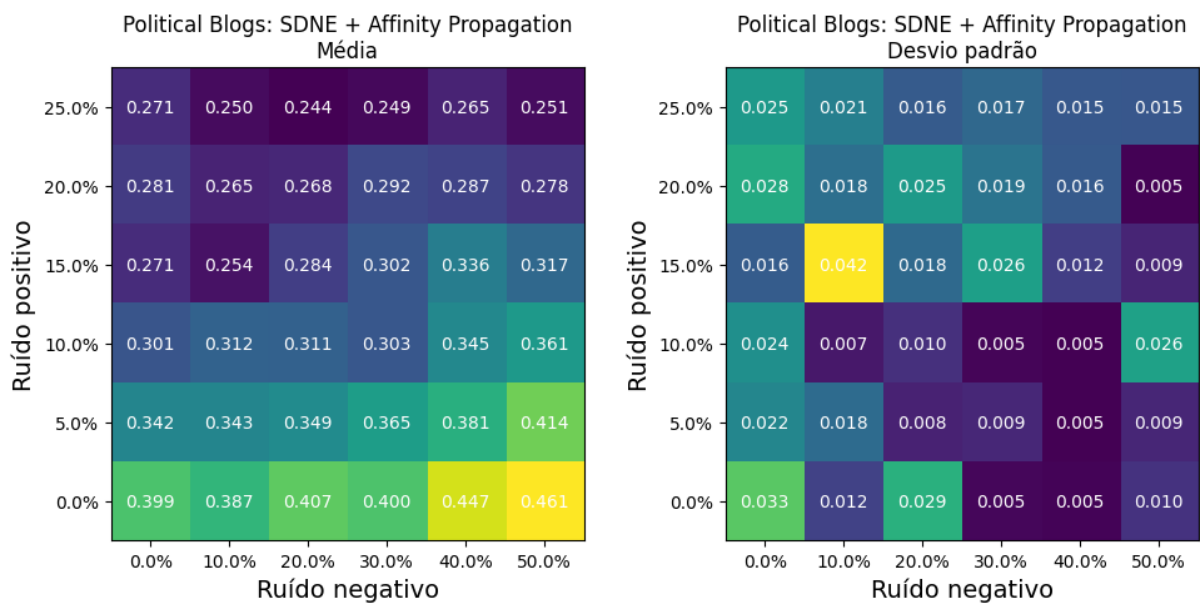
B.4.4 SDNE

Figura 73 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e K-Means sobre Political Blogs com diferentes ruídos.



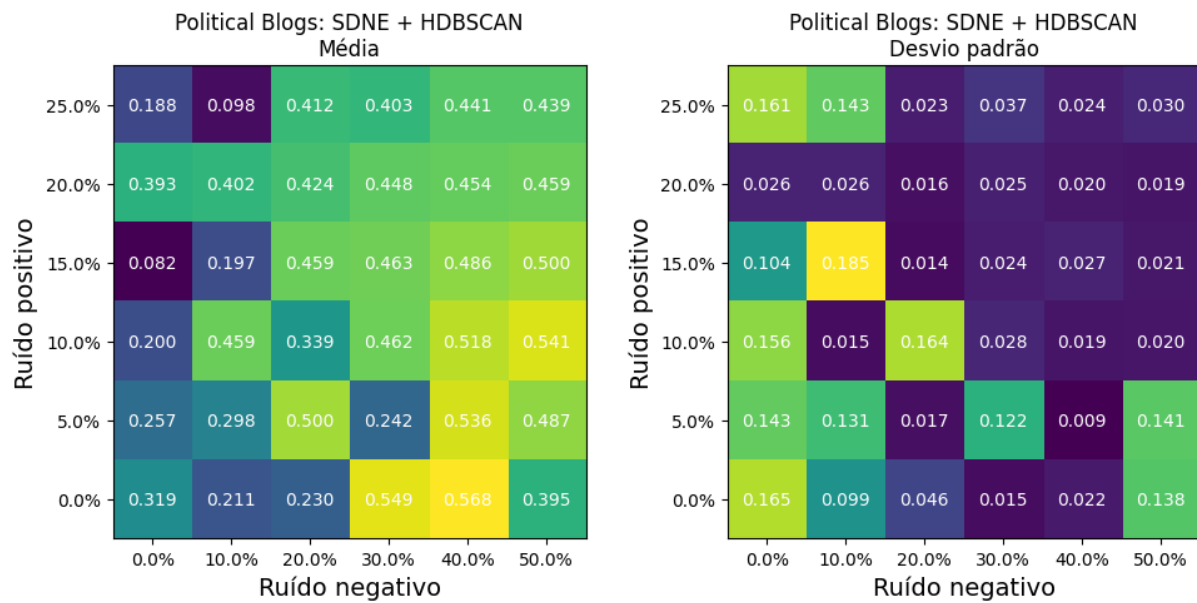
Fonte: Autor.

Figura 74 – *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e Affinity Propagation sobre Political Blogs com diferentes ruídos.



Fonte: Autor.

Figura 75 — *Heatmaps* de médias e desvios padrões dos scores de agrupamentos gerados por SDNE e HDBSCAN sobre Political Blogs com diferentes ruídos.



Fonte: Autor.