

Mauro Mascarenhas de Araújo

# **Heurísticas computacionais para extração de conhecimento em problemas de maratona de programação**

Santo André

2020, v-3.2.5

Mauro Mascarenhas de Araújo

**Heurísticas computacionais para extração de  
conhecimento em problemas de maratona de  
programação**

Monografia de conclusão de graduação apresentada ao curso de Ciência da Computação, como parte dos requisitos necessários para a obtenção do título de Bacharel.

Universidade Federal do ABC – UFABC  
Centro de Matemática, Computação e Cognição  
Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Monael Pinheiro Ribeiro  
Coorientador: Prof. Dr. Jesús Pascual Mena-Chalco

Santo André  
2020, v-3.2.5

---

Mauro Mascarenhas de Araújo

Heurísticas computacionais para extração de conhecimento em problemas de maratona de programação/ Mauro Mascarenhas de Araújo. – Santo André, 2020,  
v-3.2.5-

79 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Monael Pinheiro Ribeiro  
Coorientador: Prof. Dr. Jesús Pascual Mena-Chalco

Trabalho de graduação – Universidade Federal do ABC – UFABC  
Centro de Matemática, Computação e Cognição  
Bacharelado em Ciência da Computação, 2020, v-3.2.5.

1. Análise de enunciados. 2. Processamento de Linguagem Natural. 3. URI Online Judge. 4. Maratona de programação. 5. Recuperação de informação. I. Pinheiro Ribeiro, Monael. II. Mena Chalco, Jesús Pascual. III. Bacharelado em Ciência da Computação, 2020. IV. Heurísticas computacionais para extração de conhecimento em problemas de maratona de programação.

---

# Resumo

Dentro do contexto de recuperação de informação, este trabalho apresenta a aplicação de alguns métodos clássicos de processamento de linguagem natural sobre um conjunto de questões obtidas do *URI Online Judge* por meio de um *web scraper*, a fim de servir como base para o desenvolvimento futuro de um sistema de recomendação de desafios.

A partir das estruturas de dados obtidas e da aplicação de técnicas de PLN (matriz termo-documento e matriz TF-IDF) sobre o *corpus*, são traçadas as redes de co-ocorrência que correlacionam os desafios com base na similaridade semântica existente entre eles, dado que a métrica utilizada para o cálculo de semelhança foi a distância do cosseno.

Ainda utilizando as matrizes termo-documento e TF-IDF, é realizada uma análise exploratória utilizando a dispersão espacial dos problemas em ambiente bidimensional (para tal, diferentes técnicas de redução de dimensionalidade foram empregadas), com o intuito de buscar por agrupamentos (*clusters*) ao evidenciar outros atributos disponíveis: categoria, nível de dificuldade e taxa de resolução.

Dentre os resultados obtidos, destacam-se:

- O sucesso da aplicação das técnicas de PLN sobre o corpus analisado, uma vez que os grafos de co-ocorrência obtidos representam a similaridade semântica entre os desafios do URI de forma precisa;
- A identificação da formação de padrões com base na dispersão espacial dos enunciados, onde foram observadas algumas características dos vocabulários presentes nos enunciados e da similaridade semântica entre eles, através da visualização de categorias e níveis de dificuldade aos quais cada um deles pertencem;
- Para a geração dos grafos de co-ocorrência, a vetorização termo-documento apresentou melhores resultados, enquanto a TF-IDF saiu-se melhor na inspeção visual, na busca por correlações com os demais atributos disponíveis.

O principal fator de relevância deste trabalho é a obtenção do êxito na aplicação das heurísticas de PLN sobre um conjunto de dados inédito, bem como o foco exclusivo na análise dos enunciados dos problemas típicos de maratona de programação.

**Palavras-chaves:** análise de enunciados, processamento de linguagem natural, URI Online Judge, maratona de programação, recuperação de informação.

# Abstract

Considering the information retrieval context, the purpose of this work is to apply some classic natural language processing methods over a statement set which have been extracted from URI Online Judge with a web scraper, so as to provide a concrete basis to the future development of a recommendation system.

It is possible to map co-occurrence networks accordingly to the existing semantic similarity, which is calculated using the cosine similarity metric, among the challenges (documents), just by using the same data structures (term-document and TF-IDF matrixes) which have been acquired after applying some NLP techniques to the *corpus*.

Still considering the same term-document and TF-IDF matrixes, a “visual” analysis regarding the spatial distribution of the challenges in a bi-dimensional space is performed (the application of different dimensionality reduction techniques have been made in order to perform it), so as to search for clusters based on additional available attributes such as category, difficulty level and correct answers ratio.

The following results are worth being highlighted:

- Success when applying NLP techniques over the *corpus* (dataset), once the generated co-occurrence networks presented an accurate representation of the semantic similarity among URI Online Judge problem statements;
- Identification of clusters based on the spatial distribution of the statements, once not only have different characteristics been found over the vocabulary of the challenges with one-to-one inspections, but by emphasizing the categories and difficulty levels of each one of them;
- While the term-document vectorization method presented better results when generating co-occurrence networks, the TF-IDF method presented better results when searching for cluster formations focusing on additional attributes.

The most relevant aspect of this work is that, in addition to the success of the application of NLP heuristics over an unprecedented dataset, it also focused exclusively on the analysis of the statements rather than on the answers as well.

**Keywords:** statement analysis, natural language processing, URI Online Judge, programming marathon, information retrieval.

# Listas de ilustrações

Figura 1 – Diagrama de fluxo do sistema de pontuação automática.	16
Figura 2 – Arquitetura de sistema do processo de classificação de questões.	17
Figura 3 – Exemplo de projeção de subconjunto de dados.	29
Figura 4 – Exemplo de decomposição de matriz.	31
Figura 5 – Exemplo de projeção de subconjunto de dados.	32
Figura 6 – Metodologia.	36
Figura 7 – Alguns enunciados após o processo de tokenização.	39
Figura 8 – Alguns enunciados após o processo de remoção de <i>stopwords</i> .	39
Figura 9 – Alguns enunciados após o processo de aplicação de classificação gramatical.	40
Figura 10 – Alguns enunciados após o processo de aplicação de lematização.	41
Figura 11 – Matriz termo-documento.	43
Figura 12 – Conjunto de vetores de frequência para os problemas.	43
Figura 13 – Histograma de classes gramaticais de acordo com as categorias dos problemas.	48
Figura 14 – Maior componente conexa do grafo de co-ocorrência (considerando apenas a descrição do problema).	50
Figura 15 – Maior componente conexa do segundo grafo de co-ocorrência (considerando a descrição do problema, dos dados de entrada e de saída).	51
Figura 16 – Comparação de componentes conexas do grafo.	52
Figura 17 – Visualização da distribuição espacial dos problemas por categoria.	54
Figura 18 – Visualização da distribuição espacial dos problemas por nível de dificuldade.	55
Figura 19 – Visualização da distribuição espacial dos problemas por taxa de resolução	57
Figura 20 – Maior componente conexa do grafo de co-ocorrência (utilizando a matriz TF-IDF e considerando apenas a descrição do problema).	66
Figura 21 – Distribuição espacial dos problemas por categoria (PCA em matriz termo-documento).	67
Figura 22 – Distribuição espacial dos problemas por categoria (PCA em matriz TF-IDF).	68
Figura 23 – Distribuição espacial dos problemas por categoria (MDS em matriz termo-documento).	68
Figura 24 – Distribuição espacial dos problemas por categoria (MDS em matriz TF-IDF).	69
Figura 25 – Distribuição espacial dos problemas por categoria (SVD em matriz termo-documento).	69

Figura 26 – Distribuição espacial dos problemas por categoria (SVD em matriz TF-IDF). . . . .	70
Figura 27 – Distribuição espacial dos problemas por nível de dificuldade (PCA em matriz termo-documento). . . . .	70
Figura 28 – Distribuição espacial dos problemas por nível de dificuldade (PCA em matriz TF-IDF). . . . .	71
Figura 29 – Distribuição espacial dos problemas por nível de dificuldade (MDS em matriz termo-documento). . . . .	71
Figura 30 – Distribuição espacial dos problemas por nível de dificuldade (MDS em matriz TF-IDF). . . . .	72
Figura 31 – Distribuição espacial dos problemas por nível de dificuldade (SVD em matriz termo-documento). . . . .	72
Figura 32 – Distribuição espacial dos problemas por nível de dificuldade (SVD em matriz TF-IDF). . . . .	73
Figura 33 – Distribuição espacial dos problemas por taxa de resolução (PCA em matriz termo-documento). . . . .	73
Figura 34 – Distribuição espacial dos problemas por taxa de resolução (PCA em matriz TF-IDF). . . . .	74
Figura 35 – Distribuição espacial dos problemas por taxa de resolução (MDS em matriz termo-documento). . . . .	74
Figura 36 – Distribuição espacial dos problemas por taxa de resolução (MDS em matriz TF-IDF). . . . .	75
Figura 37 – Distribuição espacial dos problemas por taxa de resolução (SVD em matriz termo-documento). . . . .	75
Figura 38 – Distribuição espacial dos problemas por taxa de resolução (SVD em matriz TF-IDF). . . . .	76
Figura 39 – Grafo de co-ocorrência (matriz termo-documento). . . . .	77
Figura 40 – Grafo de co-ocorrência (matriz TF-IDF). . . . .	78

# Listas de tabelas

Tabela 1 – Exemplos de mapeamento de verbos para sua forma inflexionada.	21
Tabela 2 – Exemplos de palavras antes e depois de passar pelo processo de <i>stemming</i> .	21
Tabela 3 – Exemplos de palavras que sofreram <i>overstemming</i> .	22
Tabela 4 – Exemplos de palavras que sofreram <i>understemming</i> .	22
Tabela 5 – Exemplos de palavras com prefixo após passar pelo processo de <i>stemming</i> .	22
Tabela 6 – Classes gramaticais do <i>Penn Treebank</i> .	24
Tabela 7 – Exemplo de matriz termo-documento.	25
Tabela 8 – Exemplo de matriz termo-frequência.	27
Tabela 9 – Exemplo de matriz unidimensional do inverso da frequência nos documentos (transposta).	27
Tabela 10 – Exemplo de matriz TF-IDF.	27

# Listas de abreviaturas e siglas

DOM	<i>Document Object Model</i> (Modelo de Objeto de Documento)
FAQ	<i>Frequently Asked Questions</i> (Perguntas frequentes)
HTML	<i>Hypertext Markup Language</i>
ICPC	<i>International Collegiate Programming Contest</i> (Concurso de Programação Universitário Internacional)
JSON	JavaScript Object Notation
K-NN	K-Nearest Neighbours (K-Vizinhos mais próximos)
NLTK	Natural Language Toolkit (Kit de Ferramentas para Linguagem Natural)
PCA	Principal Components Analysis (Análise de componentes principais)
PLN	Processamento de Linguagem Natural
POS	<i>Part Of Speech</i> (Marcação/Classificação gramatical)
SBC	Sociedade Brasileira de Computação
SVM	<i>Support Vector Machine</i> (Máquina de Vetores de Suporte)
TF-IDF	<i>Term Frequency – Inverse Document Frequency</i> (Frequência do Termo – Inverso da Frequência nos Documentos)

# Sumário

<b>Introdução</b>	12
<b>Objetivo</b>	13
<b>1 TRABALHOS RELACIONADOS</b>	14
<b>1.1 Utilizando similaridade de sentenças e de tipos de questão para responder a perguntas similares em comunidades de compartilhamento de conhecimento</b>	14
<b>1.2 Análise comparativa de similaridade de cadeia de caracteres e baseada em corpus para sistema de pontuação automática de redações em 'gamificação' em aprendizado eletrônico</b>	15
<b>1.3 Algoritmo baseado em PLN para sistema de perguntas e respostas</b>	15
<b>1.4 Aprendendo a dar notas a perguntas de respostas curtas utilizando medidas de similaridade semântica e alinhamentos de grafos de dependência</b>	15
<b>1.5 Um classificador automático para questões de prova com WordNet e similaridade do cosseno</b>	17
<b>2 FUNDAMENTOS</b>	18
<b>2.1 Técnicas de PLN</b>	18
<b>2.1.1 Tokenização</b>	18
<b>2.1.2 Stemming e Lemmatization</b>	19
<b>2.1.2.1 Lemmatization</b>	19
<b>2.1.2.2 Stemming</b>	21
<b>2.1.3 POS tagging</b>	22
<b>2.1.4 Matriz termo-documento</b>	23
<b>2.1.5 Matriz TF-IDF</b>	24
<b>2.2 Similaridade semântica entre textos</b>	28
<b>2.2.1 Similaridade do cosseno</b>	28
<b>2.3 Redução de dimensionalidade e visualização gráfica</b>	28
<b>2.4 Analise de Componentes Principais - PCA</b>	29
<b>2.5 Escalonamento Multidimensional - MDS</b>	30
<b>2.6 Decomposição em Valores Singulares - SVD</b>	30
<b>2.7 Grafo/rede de co-ocorrência</b>	32
<b>3 COLETA DE DADOS</b>	33

<b>3.1</b>	<b>Elaboração do <i>web scraper</i></b>	<b>33</b>
<b>3.2</b>	<b>Aquisição de problemas do URI</b>	<b>35</b>
<b>4</b>	<b>METODOLOGIA</b>	<b>36</b>
<b>4.1</b>	<b>Preprocessamento</b>	<b>37</b>
4.1.1	Leitura e conversão dos dados extraídos	37
4.1.2	Remoção de problemas com dados faltantes	37
4.1.3	Tokenização	38
4.1.4	Remoção de <i>stopwords</i>	38
4.1.5	POS tagging e lematização	40
<b>4.2</b>	<b>Modelo analítico</b>	<b>41</b>
4.2.1	Geração de matriz termo-documento	42
4.2.2	Geração de grafos de similaridade semântica entre os problemas	43
4.2.3	Busca por correlação entre a similaridade semântica e outros atributos presentes nos desafios	45
<b>5</b>	<b>RESULTADOS E ANÁLISES</b>	<b>47</b>
<b>5.1</b>	<b>Classificações gramaticais</b>	<b>47</b>
<b>5.2</b>	<b>Grafos de co-ocorrência</b>	<b>49</b>
<b>5.3</b>	<b>Distribuição espacial e busca por correlações</b>	<b>52</b>
5.3.1	Categoria	53
5.3.2	Nível de dificuldade	55
5.3.3	Taxa de resolução	56
	<b>CONSIDERAÇÕES FINAIS</b>	<b>58</b>
	<b>Conclusões</b>	<b>58</b>
	<b>Limitações</b>	<b>59</b>
	<b>Trabalhos futuros</b>	<b>59</b>
	<b>REFERÊNCIAS</b>	<b>60</b>
	<b>APÊNDICES</b>	<b>65</b>
	<b>APÊNDICE A – COMPONENTES PRINCIPAIS DE GRAFOS DE CO-OCORRÊNCIA</b>	<b>66</b>
	<b>APÊNDICE B – GRÁFICOS DE DISPERSÃO DOS PROBLEMAS ANALISADOS (DISTRIBUIÇÃO ESPACIAL)</b>	<b>67</b>
	<b>APÊNDICE C – GRAFOS DE CO-OCORRÊNCIA</b>	<b>77</b>



# Introdução

Antes de implementar um sistema de recomendações, seja ele qual for, é necessário conhecer a fundo a base de dados utilizada como insumo para a elaboração do modelo, para que as sugestões geradas sejam, efetivamente, úteis. Ao trabalhar com conteúdo textual, estes sistemas tendem a utilizar técnicas de PLN junto a outros métodos de recuperação de informação, tanto para realizar a análise dos documentos, quanto para classificá-los (normalmente acompanhadas do uso de algoritmos de aprendizado de máquina).

Em ambiente educacional, a aplicação de técnicas de recuperação de informação não é inédita. Dentre as diversas linhas de pesquisa desta área, existem as que buscam por possíveis correlações entre obras literárias, as que visam encontrar respostas textuais mais adequadas a um determinado conjunto de perguntas, além das que focam em descobrir atributos que compõem uma boa resposta (ou uma resposta aceitável) para determinados tipos de questões, como é o caso de [Meurers et al. \(2011\)](#). Outro foco muito explorado neste contexto é a detecção de plágio em respostas de atividades avaliativas ([CHONG; SPECIA; MITKOV, 2010](#)) ([PAUL; JAMAL, 2015](#)) ([GUPTA; VANI; SINGH, 2014](#)).

Dentre as prováveis aplicações no contexto educacional de ciência da computação destacam-se as que envolvem análise de banco de questões de atividades, provas, ou ainda, como no caso deste trabalho, de desafios típicos de maratona de programação.

A maratona de programação é um evento destinado a alunos de graduação ou que estejam iniciando a pós-graduação na área de tecnologia (Ciência da Computação, Sistemas de Informação, etc.), promovendo criatividade na busca por novas soluções de software. Este evento é promovido desde 1996 pela Sociedade Brasileira de Computação no Brasil, onde os campeões têm vaga garantida na maratona internacional promovida pelo *International Collegiate Programming Contest* (ICPC) ([OLIVEIRA, 2019](#)).

Desafios que apresentam formatos similares aos propostos nestas maratonas podem ser encontrados em sites de teste online, como o *HackerRank* e o *URI Online Judge*, onde normalmente há a descrição do problema junto às especificações de entrada e de saída e, em alguns casos, exemplos de execução. Estes problemas são utilizados tanto por quem ainda está aprendendo a programar, quanto por programadores experientes que estão treinando para participar de *hackathons* ou que simplesmente buscam desenvolver ainda mais suas habilidades.

Uma aproximação natural de análises que buscam extrair conhecimento de um banco de questões, neste caso, típicos de maratona de programação, é buscar correlacionar os enunciados dos problemas com as demais informações disponíveis, como textos de descrição de entrada e de saída ou ainda com os dados estatísticos dos desafios, uma vez

que a aplicação de técnicas de PLN não seriam suficientes para identificar relações entre conteúdo textual dos enunciados com as respostas, que são, quase sempre, fornecidas como códigos escritos em determinada linguagem de programação.

Tendo em vista que as informações fornecidas como entrada para processos de análise de questionários tendem a ser puramente textuais, a aplicação de técnicas de PLN torna-se, praticamente, indispensável, onde o uso de cada uma delas depende do objetivo a ser alcançado.

No caso de busca por similaridade semântica entre documentos, por exemplo, técnicas clássicas de vetorização de texto e o uso da distância do cosseno como métrica de similaridade são normalmente utilizadas.

Por fim, é válido ressaltar que o conjunto de desafios analisado oferece diversas informações que podem ser utilizadas para trabalhos que extrapolam o escopo aqui apresentado. Note que elas vão além dos enunciados, como é o caso dos dados estatísticos, informações obtidas no decorrer da elaboração do modelo analítico (classes gramaticais às quais pertencem as palavras que compõem os enunciados, por exemplo) e dos demais atributos presentes nas bases extraídas da fonte (*URI Online Judge*).

## Objetivo

Este trabalho tem por objetivo realizar a análise do conjunto de problemas existentes na plataforma URI Online Judge, que contém desafios típicos de maratona de programação, a fim de servir como base para a futura criação de um sistema de recomendação de problemas.

Para tal, serão construídas matrizes termo-documento tanto com base na “contagem bruta” dos termos, quanto na frequência juntamente com o fator de penalização ‘IDF’, para então traçar a rede de co-ocorrência, considerando a similaridade semântica entre os enunciados dos desafios utilizando a métrica da distância do cosseno.

Outra meta após a construção da rede de co-ocorrência, será buscar por agrupamentos através de métodos visuais, evidenciando alguns dos atributos disponíveis: categoria, nível de dificuldade e taxa de resolução.

# 1 Trabalhos relacionados

A seguir são apresentados alguns trabalhos/artigos relacionados ao escopo deste, sendo o fator em comum, na maioria dos casos, trabalhos relacionados ao cálculo de similaridade entre documentos.

É possível notar que, na maioria dos casos, a abordagem consiste na análise das respostas às questões, mas não aos seus enunciados, como é o caso dos quatro primeiros trabalhos apresentados a seguir.

## 1.1 Utilizando similaridade de sentenças e de tipos de questão para responder a perguntas similares em comunidades de compartilhamento de conhecimento

Em “*Utilizing sentence similarity and question type similarity to response to similar questions in knowledge-sharing community*”, é apresentada a ideia de reutilizar informações redundantes em pares pergunta-resposta, de forma que, se a mesma pergunta for feita múltiplas vezes por diferentes usuários, então a resposta dada pelo sistema deve estar associada à pergunta redundante. Porém, é enfatizado que a tarefa não é tão simples quanto parece, uma vez que, medidas tradicionais de semelhança não são efetivas e nem eficientes em detectar semelhanças a nível de sentença/frase, pois, as técnicas de similaridade de documentos tendem a não apresentar bons resultados ao lidar textos pequenos, uma vez que eles tendem a apresentar poucas palavras em comum. O foco do trabalho está em identificar questões que expressam a necessidade pela mesma informação, sendo o principal objetivo, relacionar questões com suas paráfrases. Para atingi-lo, foi proposto um método de similaridade híbrido, que combina semântica, sintática e tipo de questão. As informações semânticas e sintáticas são medidas levando em consideração a similaridade e ordenação das palavras, além de informações de classificação gramatical, enquanto as informações sobre os tipos de questão são derivadas de um classificador SVM. Os resultados obtidos mostraram que o método apresentado é altamente efetivo em detecção de perguntas/questões redundantes ([ACHANANUPARP et al., 2008](#)).

## 1.2 Análise comparativa de similaridade de cadeia de caracteres e baseada em corpus para sistema de pontuação automática de redações em ‘gamificação’ em aprendizado eletrônico

Em “*Comparative Analysis of String Similarity and Corpus-Based Similarity for Automatic Essay Scoring System on E-Learning Gamification*” é proposto um sistema de pontuação automática para redações conduzidas em plataformas de aprendizado eletrônico utilizando métodos não supervisionados. Para isso, são utilizadas e comparadas duas métricas de similaridade: a do cosseno e a análise semântica latente, conforme exibido na Figura 1 ([SAKTI; FAUZI, 2016](#)).

Os parâmetros utilizados para medir o desempenho destes métodos foram a complexidade computacional (medido pelo uso de CPU e memória, além do tempo de carregamento da páginas) e a acurácia (medida pela correlação de Pearson e o erro médio absoluto). Os testes mostraram que ambos os algoritmos apresentaram resultados similares quanto à acurácia. Porém, a métrica do cosseno apresentou melhor desempenho computacional no servidor, fazendo com que fosse a escolhida para ser implementada no sistema de pontuação automática de redações em plataformas de aprendizado eletrônico.

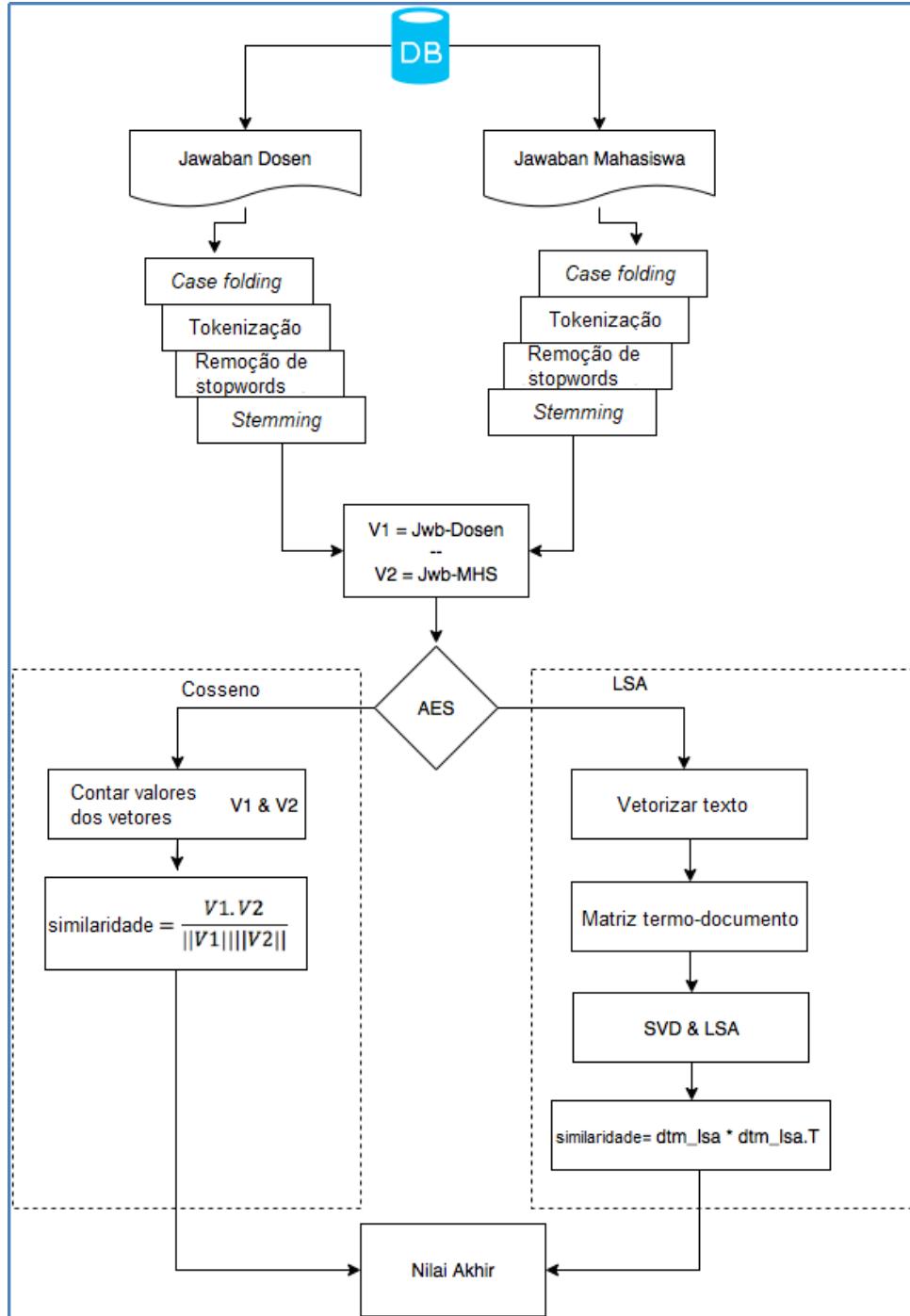
## 1.3 Algoritmo baseado em PLN para sistema de perguntas e respostas

Em “*NLP Algorithm Based Question and Answering System*”, foi apresentado um trabalho similar ao descrito na [seção 1.1](#), onde o objetivo é entender os termos de busca do usuário utilizando técnicas de PLN juntamente com um novo mecanismo de pontuação para extrair as informações relacionadas. Para testar os resultados, foi desenvolvido um simples modelo de FAQ no contexto de seguradora, que foi melhorado ao ponto de responder qualquer pergunta relacionada a seguro, de forma que foi testado com duzentas solicitações diferentes e realizada a validação cruzada com três especialistas da área de seguros ([SARKAR et al., 2015](#)).

## 1.4 Aprendendo a dar notas a perguntas de respostas curtas utilizando medidas de similaridade semântica e alinhamentos de grafos de dependência

O trabalho apresentado em “*Learning to Grade Short Answer Questions Using Semantic Similarity Measures and Dependency Graph Alignments*” é ligeiramente similar

Figura 1 – Diagrama de fluxo do sistema de pontuação automática.



Fonte: [Sakti e Fauzi \(2016, p. 03\).](#)

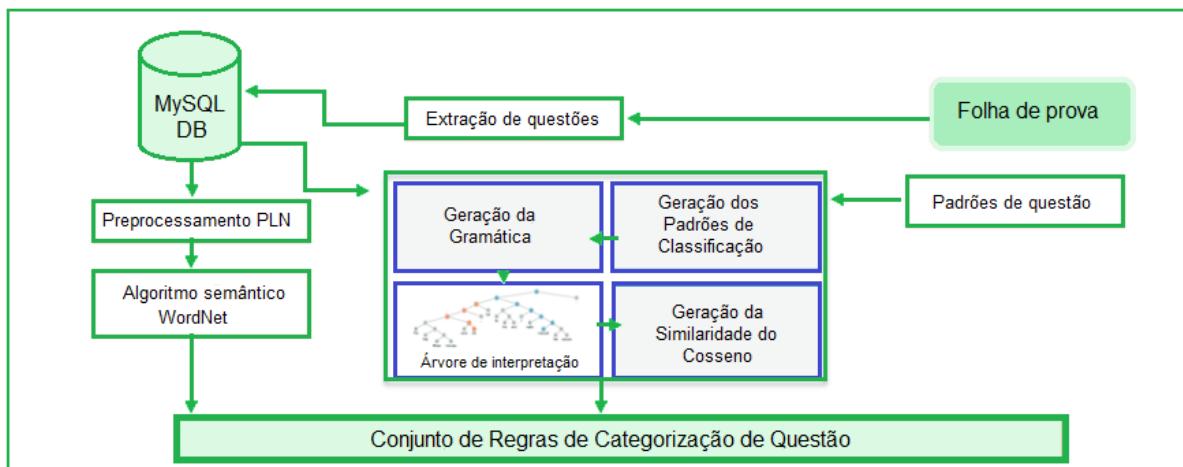
ao apresentado na [seção 1.2](#), onde o objetivo é atribuir notas para as respostas dadas para cada problemas, mas utilizando métricas de similaridade léxica e técnicas de aprendizado de máquina, ao invés de utilizar somente similaridade semântica, além de focar na análise de respostas cujos textos são pequenos, ou seja, há baixa possibilidade de correspondência léxica entre a resposta correta e as fornecidas pelos estudantes.

Por fim, é apresentado um esboço de alinhamento entre os grafos de dependência das respostas dos estudantes e do instrutor, a fim de fazer uso de um componente estrutural na pontuação automática para as respostas dos alunos (MOHLER; BUNESCU; MIHALCEA, 2011).

## 1.5 Um classificador automático para questões de prova com *WordNet* e similaridade do cosseno

Diferentemente dos demais trabalhos, em “*An automatic classifier for exam questions with WordNet and Cosine similarity*” o foco está na análise dos enunciados dos problemas ao invés das respostas, onde foi implementado um algoritmo que classifica questões de prova de acordo com a taxonomia de Bloom, a fim de que as provas/atividades possam estar de acordo com a curva de aprendizado esperada dos estudantes. O algoritmo utiliza tanto o algoritmo de similaridade do *WordNet*, que depende dos verbos extraídos das questões, quanto a similaridade do cosseno para identificar padrões entre as questões, sendo, este último, mais útil para classificar questões de prova que não contêm verbo algum (JAYAKODI; BANDARA; MEEDENIYA, 2016).

Figura 2 – Arquitetura de sistema do processo de classificação de questões.



Fonte: Adaptado de Jayakodi, Bandara e Meedeniya (2016, p. 16).

O sistema desenvolvido apresentou uma acurácia de aproximadamente 71% na classificações dos problemas, contrastando-as com as providas por especialistas da área. Tendo em vista que os educadores nem sempre têm tempo hábil ou recursos disponíveis para certificarem-se que as perguntas redigidas estão de acordo com a curva de aprendizado que eles buscam atingir, esta ferramenta pode ser de grande utilidade para avaliar estas questões presentes em seus testes (JAYAKODI; BANDARA; MEEDENIYA, 2016).

## 2 Fundamentos

Na primeira parte deste capítulo, principalmente na [seção 2.1](#), são apresentadas algumas técnicas de PLN que são aplicadas ao conjunto de dados, a fim de remover informações inconsistentes ou faltantes, que causariam interferência no modelo analítico, além de normalizar os dados restantes, a fim de melhorar a eficácia da análise semântica dos problemas.

Na segunda parte ([subseção 2.2.1](#)), são discutidas algumas maneiras de calcular a similaridade semântica entre textos diferentes, explicando algumas métricas mais usadas.

Por fim, na [seção 2.7](#), é dada a definição de grafos ou redes de co-ocorrência, enfatizando aplicações no contexto de PLN.

### 2.1 Técnicas de PLN

Na [subseção 2.1.1](#) é apresentado o conceito geral de tokenização, uma das primeiras etapas realizadas no preprocessamento textual. Ela contém alguns exemplos que evidenciam o trabalho que o algoritmo deve realizar ao exibir os dados de entrada e saída, além dos parâmetros esperados.

A [subseção 2.1.2](#) apresenta os processos de *stemming* e lematização, que são ferramentas utilizadas na normalização textual, permitindo que palavras diferentes, cujos significados sejam similares ou idênticos possam ser agrupadas, evitando que haja perda de informação na criação do modelo. Também são evidenciadas as diferenças existentes entre estes processos.

Na [subseção 2.1.3](#) é apresentado o conceito geral de POS *tagging*, ou classificação gramatical de palavras. Nela também são contrastados os principais grupos de algoritmos que realizam este trabalho (os baseados em regras e os estocásticos), além de disponibilizar a lista de possíveis classificações existentes na implementação da biblioteca NLTK.

A [subseção 2.1.4](#) contém a definição de matriz termo-documento, bem como a explicação sobre sua importância na análise de similaridade semântica entre documentos no contexto de PLN.

#### 2.1.1 Tokenização

Dada uma sequência de caracteres (*String*) e uma unidade definida de documento (Parágrafo/Frase/Oração/Palavra/etc.), tokenização (do inglês *tokenization*) é a tarefa de

quebrar a sequência fornecida em pedaços (chamados *tokens*), normalmente descartando caracteres de pontuação. Exemplos de tokenização para a frase “Olá, como vai?”:

- Sendo “palavra”, a definição de *token*, a frase será quebrada na seguinte forma: [“Olá”, “como”, “vai”];
- Sendo “letra/caractere”, a definição de *token*, a frase será quebrada na seguinte forma: [“O”, “l”, “á”, “c”, “o”, “m”, “o”, “v”, “a”, “i”].

Note que a frase foi dividida em três partes (palavras), no primeiro caso, e em dez partes (caracteres), no segundo, de forma que foram descartados os sinais de pontuação em ambas as situações.

Por vezes, o *tokens* são erroneamente definidos como termos ou palavras. Porém, é importante fazer uma distinção entre os tipos de *tokens*. Um *token* é uma sequência de caracteres presentes em um *corpus* (no contexto de PLN, um *corpus* é um conjunto de documentos (ou de frases) geralmente anotados e utilizados para aprendizado (análise) e/ou validação (verificação) ([MENA-CHALCO, 2019](#)) que é agrupada como uma unidade útil para processamento semântico ([MANNING; RAGHAVAN; SCHÜTZE, 2008](#)).

## 2.1.2 *Stemming* e *Lemmatization*

No contexto de PLN, é necessário que duas representações distintas de uma palavra comportem-se de forma similar (uma forma de agrupá-las). Para atingir tal feito, são utilizadas técnicas que buscam reduzir a palavra em sua forma básica ou literalmente “cortá-las”, a fim de que, palavras similares assumam o mesmo formato e/ou comportamento durante o processamento do *corpus* ([JURAFSKY; MARTIN, 2008](#)).

### 2.1.2.1 *Lemmatization*

*Lemmatization*, ou lematização, é a técnica que busca transformar as palavras em suas formas básicas, ou seja, é a tarefa de determinar que duas palavras possuem a mesma raiz, independentemente de suas diferenças superficiais (conjugação, por exemplo) ([JURAFSKY; MARTIN, 2008](#)). O algoritmo que implementa esta técnica normalmente faz o uso de um dicionário que contém a forma básica das palavras da linguagem a ser analisada, além de realizar uma análise morfológica delas ([MANNING; RAGHAVAN; SCHÜTZE, 2008](#)), todavia, também pode ser implementado como um automato finito (ou transdutor de estados finitos) ([PADMANABHAN, 2019](#)).

O primeiro algoritmo de lematização foi publicado por Grzegorz Chrupała em 2006, em que foi modelado como um problema de classificação, onde o algoritmo escolhe uma dentre várias “árvores de edição” que podem transformar uma palavra ao seu *lemma* ([CHRUPAŁA, 2006](#)).

A [Tabela 1](#) contém exemplos de transformação de palavras aparentemente distintas, mas cuja forma inflexionada é a mesma:

Tabela 1 – Exemplos de mapeamento de verbos para sua forma inflexionada.

Verbo	Verbo inflexionado
É	Ser
Era	Ser
Seja	Ser
Sido	Ser

Tabela contendo algumas formas flexionadas do verbo “ser”.

### 2.1.2.2 *Stemming*

Diferentemente da *lematização*, o processo de *stemming* consiste em “cortar” uma palavra de acordo com um conjunto de regras preestabelecidas (normalmente elaboradas com a ajuda de, ou por, um linguista), reduzindo-a a sua raiz, chamada de ‘*stem*’. As palavras resultantes não devem ser, necessariamente, iguais à forma básica da palavra original (como em um dicionário) (HEIDENREICH, 2018) ou ter algum significado aparente, pois, o objetivo é simplesmente normalizá-las para que as que forem similares possam ser agrupadas.

Em um contexto histórico, o primeiro algoritmo de *stemming*, desenvolvido para a língua inglesa, foi publicado por Julie Beth Lovins em 1968 (LOVINS, 1968), que, por ser um trabalho de suma importância para a área de PLN, tornou-se referência para diversos trabalhos posteriores. Porém, o algoritmo mais utilizado atualmente para a língua inglesa foi publicado em 1980 por Martin Porter (PORTER, 1980), que teve a implementação oficial do código escrito em C distribuída nos anos 2000 (PORTER, 2006).

A seguir, há um exemplo onde diferentes palavras são convertidas para o mesmo *stem* após passar pelo algoritmo de *stemming*:

Tabela 2 – Exemplos de palavras antes e após passar pelo processo de *stemming*.

Palavra	Stem
Amigo	Amig
Amiga	Amig
Amigão	Amig

Esta tabela exibe palavras cujo radical é semelhante sendo truncadas para o mesmo *stem*.

Por ser mais simples e por não precisar fazer o uso de dicionários, os algoritmos que implementam este processo costumam ser bem menos custosos que os de lematização (HARMAN; CANDELA, 1990)(HEIDENREICH, 2018). Logo, ele acaba sendo o mais utilizado para realizar um dos passos de normalização do corpus.

Por normalmente serem baseados em heurísticas, existem alguns problemas que podem ocorrer no processo de *stemming*. São eles o *overstemming* e *understemming*.

O *overstemming* ocorre quando uma palavra é demasiadamente cortada, implicando que o *stem* gerado seja algo sem significado, ou seja, o sentido original da palavra é perdido ou alterado. Isso pode fazer com que palavras com sentidos totalmente diferentes sejam “mapeadas” para o mesmo *stem*. Já o *understemming* ocorre quando duas palavras flexionadas deveriam ser “mapeadas” para a mesma raiz, mas que, por “falha” do algoritmo, não são (HEIDENREICH, 2018). A Tabela 3 apresenta alguns exemplos de palavras que sofrem com o problema de *overstemming*, enquanto a Tabela 4 apresenta exemplos de palavras que sofrem com o problema de *understemming*:

Tabela 3 – Exemplos de palavras que sofreram *overstemming*.

Palavra	Stem
Universal	Univers
Universidade	Univers
Universo	Univers

Relação de palavras cujo radical é semelhante, sendo truncadas para o mesmo *stem*, mesmo possuindo significados completamente distintos.

Tabela 4 – Exemplos de palavras que sofreram *understemming*

Palavra	Stem
Alumnus	Alumnu
Alumni	Alumni
Alumna/Alumnae	Alumna

As palavras exemplificadas deveriam ser reduzidas ao mesmo *stem*. Porém, em alguns casos, o algoritmo não realiza o “mapeamento” de forma correta.

Embora este processo seja computacionalmente mais barato, também possui algumas limitações como a falta de eliminação de prefixos, como exemplificado na Tabela 5:

Tabela 5 – Exemplos de palavras com prefixo após passar pelo processo de *stemming*.

Palavra	Stem
Felicidade	Fel
Infelicidade	Infel

Tabela contendo exemplo de palavras com significados relacionados, mas com *stems* diferentes.

Ao passo que, para a língua inglesa, o algoritmo mais famoso (e mais utilizado) é o *Porter Stemmer* (PORTER, 1980), para a língua portuguesa, o mais utilizado é o RSLP *Stemmer*, desenvolvido por Viviane Moreira Orengo e Christian Huyck em 2001 (MOREIRA, 2010).

### 2.1.3 POS *tagging*

O POS *tagger* é um *software* que lê um *corpus* em certo idioma e atribui a classe gramatical (e subclasse, dependendo do algoritmo) a cada palavra ou *token* presente no

texto (Ex.: substantivo, adjetivo, verbo etc.) baseando-se na tanto na definição da palavra quanto no contexto onde ela está inserida (Stanford NLP Group, 2019). Logo, POS *tagging* é a aplicação do algoritmo sobre um *corpus* a fim de obter as classes gramaticais das palavras (Lexical Computing, 2019).

Algoritmos de POS *tagging* podem ser classificados em dois grupos distintos: os baseados em regra e os estocásticos.

Os algoritmos baseados em regras utilizam informações contextuais para atribuir *tags* a palavras desconhecidas ou ambíguas. A desambiguação é feita ao analisar as características linguísticas da palavra atual, da palavra predecessora e da palavra sucessora. Por exemplo: se a palavra predecessora for um artigo e a palavra atual for desconhecida ou ambígua, então é provável que ela seja um substantivo (GODAYAL; MALHOTRA, 2018).

Por outro lado, algoritmos estocásticos são algoritmos que, de alguma forma, utilizam a frequência ou probabilidade para classificar as palavras. As versões mais simples deste grupo desambiguam as palavras baseado-se na probabilidade de uma palavra ser pertencente a uma determinada classe. Outra alternativa é calcular a frequência da ocorrência de certa sequência de classes (normalmente chamada de abordagem por n-gramas). A última alternativa faz muito mais sentido que a primeira, uma vez que considera as classificações para palavras individuais baseada no contexto (GODAYAL; MALHOTRA, 2018).

A Tabela 6 representa as possíveis classificações que podem ser obtidas ao utilizar o POS *tagger* (em inglês) no NLTK (RACHIELE, 2018).

#### 2.1.4 Matriz termo-documento

Uma matriz termo-documento ou documento-termo é uma matriz matemática que representa a frequência com a qual determinados termos ocorrem em um conjunto de documentos/textos (*corpus*). Em uma matriz termo-documento, as linhas correspondem aos termos/palavras e as colunas aos documentos/textos (FACER, 2017). Ela também pode ser interpretada como uma implementação do conceito de *Bag Of Words* (BISHOP, 2017), uma forma de representar os dados do texto na modelagem de algoritmos de aprendizado de máquina (BROWNLEE, 2017).

Sejam  $D_1$  e  $D_2$  dois documentos:

- $D_1$  = “Eu amo sorvete de morango”
- $D_2$  = “Eles não gostam de sorvete de morango”

A matriz termo-documento para estes dois pequenos textos é apresentada na Tabela 7.

Tabela 6 – Classes gramaticais do *Penn Treebank*.

Classificação	Significado
CC	Conjunção coordenativa
CD	Dígito cardinal
DT	Determinante
EX	'there' existencial, como em ' <i>there is</i> '
FW	Palavra estrangeira
IN	Preposição/conjunção subordinativa
JJ	Adjetivo. Ex.: 'big'
JJR	Adjetivo comparativo. Ex.: 'bigger'
JJS	Adjetivo superlativo. Ex.: 'biggest'
LS	Marcador de lista. Ex.: '1)
MD	Modal. Ex.: 'could', 'will'
NN	Substantivo singular. Ex.: 'desk'
NNS	Substantivo no plural. Ex.: 'desks'
NNP	Substantivo próprio no singular. Ex.: 'Harrison'
NNPS	Substantivo próprio no plural. Ex.: 'Americans'
PDT	Predeterminante. Ex.: 'all the kids'
POS	Final possessivo. Ex. 'parent's'
PRP	Pronome pessoal. Ex.: 'I', 'he', 'she'
PRP\$	Pronome possessivo. Ex.: 'my', 'his', 'hers'
RB	Advérbio. Ex.: 'very', 'silently'
RBR	Advérbio comparativo. Ex.: 'better'
RBS	Advérbio superlativo. Ex.: 'best'
RP	Partícula. Ex.: 'give up'
TO	'to go' 'to' the store'
UH	Interjeição. Ex.: 'errrrrrrrrm'
VB	Verbo na forma básica. Ex.: 'take'
VBD	Verbo no passado. Ex.: 'took'
VBG	Verbo no gerúndio. Ex.: 'taking'
VBN	Verbo no particípio passado. Ex.: 'taken'
VBP	Verbo no presente - singular. Ex.: 'non-3d take'
VBZ	Verbo no presente - terceira pessoa no singular. Ex.: 'takes'
WDT	Determinante que inicia-se com 'wh'. Ex.: 'which'
WP	Pronome que inicia-se com 'wh'. Ex.: 'who', 'what'
WP\$	Pronome possessivo que inicia-se com 'wh'. Ex.: 'whose'
WRB	Advérbio que inicia-se com 'wh'. Ex.: 'where', 'when'

Tabela de classificação *POS* para a língua inglesa do *Penn Treebank* disponível na NLTK.

Note que a [Tabela 7](#) mostra quais termos estão contidos em quais documentos e quantas vezes eles aparecem.

Uma matriz termo-documento pode ficar muito grande e esparsa dependendo da quantidade de documentos e do número de termos que cada *corpus* possui.

### 2.1.5 Matriz TF-IDF

Existem outras formas de vetorizar textos além de construir uma matriz termo-documento. Uma dessas formas é construir uma matriz TF-IDF (do inglês *term frequency-inverse document frequency*), que considera a frequência dos termos, bem como o inverso da frequência nos documentos, de forma que ela é muito utilizada em problemas de

Tabela 7 – Exemplo de matriz termo-documento.

	$D_1$	$D_2$
Eu	1	0
Amo	1	0
Sorvete	1	1
De	1	2
Morango	1	1
Eles	0	1
Não	0	1
Gostam	0	1

Matriz termo-documento para os textos  $D_1$  e  $D_2$ .

recuperação de informação, bem como em mineração de texto.

Para calcular a matriz TF-IDF são utilizadas duas fórmulas: a primeira delas corresponde ao cálculo da frequência dos termos em cada documento ( $tf(t, d)$ ), que pode apresentar diversas variações, dentre elas:

$$tf(t, d) = \begin{cases} 0, & \text{se } t \text{ não estiver em } d \\ 1, & \text{caso contrário.} \end{cases} \quad (2.1)$$

$$tf(t, d) = f_{t,d} \quad (2.2)$$

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (2.3)$$

A [Equação 2.1](#) apresenta o caso booleano, onde o peso da frequência do termo para o documento é binário, ou seja, será 1, caso esteja no documento, ou 0, caso contrário. Já a [Equação 2.2](#) é utilizada quando deseja-se obter a contagem bruta de vezes que determinado termo  $t$  aparece em um documento  $d$ , denominada por  $f_{t,d}$  (como no caso da matriz termo-documento, apresentada na [subseção 2.1.4](#)). Por fim, na [Equação 2.3](#), o peso gerado é dado pela frequência bruta do termo ( $f_{t,d}$ ) dividida pela quantidade de palavras no documento, ou seja, normaliza os valores para suas representações relativas (contidas no intervalo  $[0, 1]$ ) ([MANNING; RAGHAVAN; SCHÜTZ, 2008](#)).

Utilizar a contagem bruta para realizar cálculos de similaridade entre documentos apresenta um problema crítico: todos os termos são considerados igualmente importantes (utilizar o método binário acaba agravando ainda mais este problema). A fim de reduzir a relevância para termos banais (comuns à maioria dos documentos), utiliza-se o cálculo da frequência nos documentos ( $df_t$ , do inglês *document frequency*), definida pelo número de documentos no *corpus* ( $N$ ) que contêm o termo  $t$ . Porém, como o objetivo é penalizar os termos banais (e não aumentar sua relevância), utiliza-se o inverso da frequência nos

documentos, dado pela seguinte fórmula (sendo esta, a segunda etapa no cálculo da matriz TF-IDF) (MANNING; RAGHAVAN; SCHÜTZE, 2008):

$$idf(t) = \ln \frac{N}{df_t} \quad (2.4)$$

Assim como no caso do cálculo da frequência do termo em documentos, existem variações na forma como o inverso da frequência nos documentos é calculado, sendo algumas delas:

$$idf(t) = 1 \quad (2.5)$$

$$idf(t) = \ln \left( \frac{N}{df_t + 1} \right) + 1 \quad (2.6)$$

A Equação 2.5 é utilizada quando não há penalização, ou seja, o fator de multiplicação é igual a 1, como no caso da matriz termo-documento padrão. Já a Equação 2.6, conhecida como fórmula do inverso da frequência nos documentos suavizada, é uma variação utilizada para evitar problemas com denominadores iguais a 0 (pode acontecer, caso o termo não apareça em qualquer um dos documentos) e para definir um valor base (neste caso, 1), a fim de evitar que termos presentes em grande parte dos documentos sejam muito penalizados, a ponto de ter seu peso reduzido a zero (a única exceção é quando o termo está presente em todos os documentos, que é quando o valor retornado pelo logaritmo é menor que zero, fazendo com que o termo seja, de fato, penalizado). Neste caso, a função deixa de retornar um fator de penalização, mas um fator de “reforço” para termos menos banais (FOUNDATION, 2017).

Por fim, o último passo para obter a matriz TF-IDF é multiplicar as duas matrizes obtidas com as operações anteriores (MANNING; RAGHAVAN; SCHÜTZE, 2008), ou seja:

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \quad (2.7)$$

Utilizando os textos  $D_1$  e  $D_2$  como documentos que compõem o corpus  $\mathcal{C}$  para o exemplo de construção de uma matriz TF-IDF, são obtidas duas matrizes intermediárias, a de termo-frequência, exemplificada na Tabela 8, e a de inverso da frequência nos documentos, exemplificada na Tabela 9.

Ao aplicar a Equação 2.7, a matriz TF-IDF é obtida, conforme exemplificado na Tabela 10.

Ao contrastar a Tabela 7 com a Tabela 9, pode-se observar que o módulo dos valores acabou diminuindo devido à forma com que o cálculo da frequência foi realizado, além de

Tabela 8 – Exemplo de matriz termo-frequênciaria.

	$D_1$	$D_2$
Eu	0.2	0
Amo	0.2	0
Sorvete	0.2	0.1428
De	0.2	0.2857
Morango	0.2	0.1428
Eles	0	0.1428
Não	0	0.1428
Gostam	0	0.1428

Matriz termo-frequênciaria para  $\mathcal{C}$ . Neste caso, foi a [Equação 2.3](#) foi utilizada como fórmula de cálculo para a frequênciaria dos termos.

Tabela 9 – Exemplo de matriz unidimensional do inverso da frequênciaria nos documentos (transposta).

	$idf$
Eu	1
Amo	1
Sorvete	0.5945
De	0.5945
Morango	0.5945
Eles	1
Não	1
Gostam	1

Matriz unidimensional do inverso da frequênciaria nos documentos (transposta) para  $\mathcal{C}$ . Neste caso, os valores  $IDF$  são dados pela [Equação 2.6](#).

Tabela 10 – Exemplo de matriz TF-IDF.

	$D_1$	$D_2$
Eu	0.2	0
Amo	0.2	0
Sorvete	0.1189	0.0849
De	0.1189	0.1698
Morango	0.1189	0.0849
Eles	0	0.1428
Não	0	0.1428
Gostam	0	0.1428

Exemplo de matriz TF-IDF para  $\mathcal{C}$ .

também ser possível notar que os termos “Sorvete”, “De” e “Morango” foram penalizados por aparecerem em ambos os documentos, enquanto os demais foram preservados.

As diferenças observadas acabam fazendo diferença no cálculo de similaridade semântica entre os documentos que compõem um dado corpus, uma vez que, além de o cosseno ser sensível ao módulo dos vetores de entrada, termos mais “banais” foram penalizados. Em *corpus* pequenos (como no caso analisado), o uso da matriz TF-IDF como representação vetorial para eles pode não ser o mais adequado, uma vez que informações

importantes acabam por ser tomadas como menos relevantes e, até mesmo, tendo pesos diferentes (como é o caso de “Sorvete” e “Morango”), impactando negativamente o cálculo de similaridade entre eles.

## 2.2 Similaridade semântica entre textos

Existem diversas maneiras de calcular a similaridade semântica entre dados textuais. As formas mais simples baseiam-se no uso de uma matriz termo-documento, sobre a qual é aplicada uma determinada métrica de distância, como as de Jaccard, *Manhattan*, Euclideana, cosseno, ou outras. A ideia é conseguir mensurar o quanto similares ou parecidos são dois textos distintos ([SIEG, 2018](#)).

### 2.2.1 Similaridade do cosseno

A métrica de similaridade do cosseno é utilizada para determinar o quanto similares são os documentos analisados, independentemente de seus tamanhos.

Matematicamente, ela mede o cosseno do ângulo entre dois vetores projetados em um espaço multidimensional. No contexto atual, os dois vetores utilizados seriam as colunas da matriz, que contêm a contagem das palavras dos dois documentos.

Ao plotar em um espaço multidimensional, onde cada dimensão corresponde a uma palavra do texto, a similaridade do cosseno captura a orientação (ângulo) existente entre os documentos, e não a magnitude.

Esta métrica é vantajosa, pois, mesmo que dois documentos similares forem tomados como distantes pela distância Euclidiana devido ao seus tamanhos, eles ainda podem ter um ângulo relativamente pequeno entre si. De forma que, quanto menor for o ângulo, maior a similaridade entre eles ([PRABHAKARAN, 2019](#)).

Sabendo que o cosseno entre dois vetores é dado por  $\cos = \frac{u \cdot v}{|u||v|}$ , temos que a distância entre os textos  $D_1$  e  $D_2$ , dada por  $1 - \cos(D_1, D_2)$ , é cerca de 0.4, ao utilizar os vetores gerados na construção da matriz termo-documento “bruta”, ou seja, eles apresentam aproximadamente 60% de similaridade, por exemplo.

## 2.3 Redução de dimensionalidade e visualização gráfica

A redução de dimensionalidade pode ser necessária em diversos contextos, principalmente quando os vetores de atributos tornam-se muito grandes e esparsos (ocorrência muito comum em problemas de mineração de texto e aprendizado de máquina, uma vez que quanto maior a quantidade de colunas de uma matriz (atributos), maior tende a ser a taxa de erro do modelo).

Ela também pode ser utilizada para o simples propósito de possibilitar a visualização gráfica de elementos dispersos em espaços com mais de três dimensões.

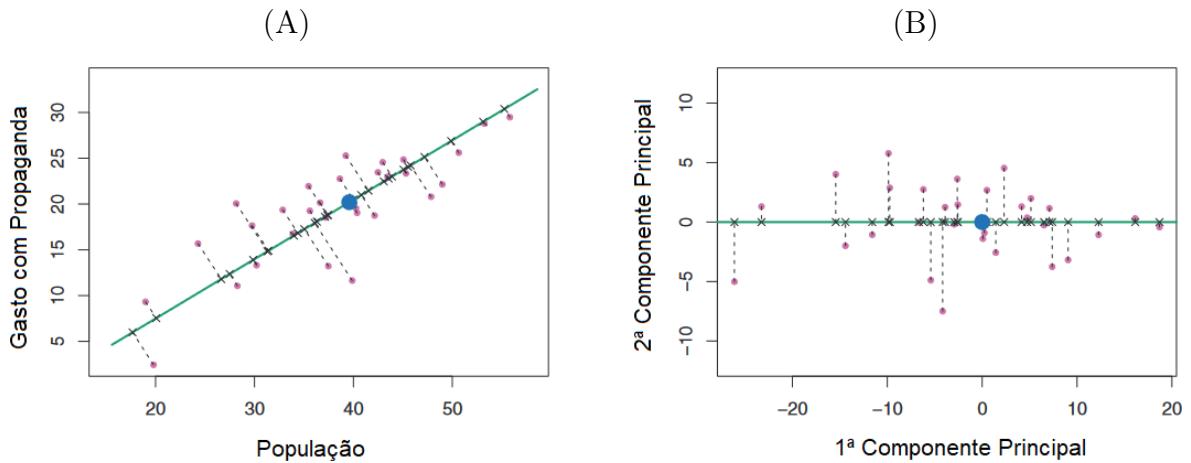
## 2.4 Analise de Componentes Principais - PCA

O PCA, do inglês *Principal Component Analysis*, é um método popular utilizado para derivar um pequeno conjunto de atributos de um grande conjunto de variáveis.

Trata-se de uma técnica para reduzir a dimensão dos dados de uma matrix  $X_{n \times p}$ . A direção da primeira componente principal dos dados é aquela na qual as observações mais variam.

Tomemos um cenário de exemplo onde temos o tamanho da população (*pop*) em milhares de habitantes e o gasto com propagandas de uma corporação popular (*ad*, do inglês *advertisement*), para 100 cidades Figura 3. Podemos observar (simplesmente visualizando a distribuição espacial), que a direção descrita pela reta verde (painel esquerdo), a primeira componente principal, é a que apresenta a maior variação de dados. Em outras palavras, se as 100 observações fossem projetadas sobre esta reta, então o resultado da projeção das observações apresentariam a maior variância possível; ao projetar as observações em qualquer outra reta acabaríamos obtendo observações com menor variância (JAMES et al., 2013).

Figura 3 – Exemplo de projeção de subconjunto de dados.



A mediana da população (*pop*) e de gastos em propaganda (*ad*) estão indicadas por um círculo sobre a reta ( $\overline{pop}, \overline{ad}$ ). A: A direção da primeira componente principal é exibida pela reta. Ela é a dimensão na qual os dados mais variam, além de ser a que define a linha mais próxima às n observações. As distâncias de cada observação para a componente principal estão representadas pelos traçados que ligam o ponto à reta. B: O painel esquerdo foi rotacionado de forma que a primeira componente principal coincide com o eixo  $x$ . Fonte: (JAMES et al., 2013).

Note que no exemplo apresentado, houve uma redução de um cenário de duas para uma dimensão, mas o PCA pode ser aplicado várias vezes, realizando projeções sucessivas a fim de reduzir um problema cujo conjunto de dados apresente dimensão  $m$  para uma dimensão menor  $n$ , onde  $m, n \in N^*, m > n$ .

## 2.5 Escalonamento Multidimensional - MDS

Para dados cuja dispersão não seja dada de forma linear, o método PCA pode não ser o mais recomendado, pois, reduz os dimensões através de sucessivas projeções em hiperplanos, ou seja, de forma linear. Desta forma, para estes casos, a perda de informação tem grandes chances de ser maior.

O Escalonamento Multidimensional, do inglês *Multidimensional Scaling*, é uma técnica para realizar análise de similaridade ou diferença em um conjunto de objetos. Tais dados podem ser correlações de itens de teste, classificações de similaridade de candidatos políticos ou ainda índices de negociação para um conjunto de países. O MDS tenta modelar tal dado como distâncias entre pontos em um espaço geométrico. O principal motivo para fazer isso é a busca por uma representação gráfica da estrutura dos dados, uma que seja muito mais fácil de se entender que um vetor de números e que, portanto, exiba a informação central (essencial) dos dados, de forma a amenizar os ruídos (BORG, 2005).

Existem dois tipos de algoritmos MDS, o métrico e o não métrico. No MDS métrico, a matriz de similaridade de entrada emerge de uma métrica e, portanto, respeita a desigualdade triangular, de forma que as distâncias entre dois pontos da saída devem estar o mais próximo possível à similaridade ou diferença dos dados. Já na versão não métrica, os algoritmos tentam preservar a ordem das distâncias e, portanto, buscam por uma relação monótona entre as distâncias no espaço incorporado e as similaridades/diferenças (PEDREGOSA et al., 2020b).

## 2.6 Decomposição em Valores Singulares - SVD

A fim de decompor uma matriz termo-documento  $C_{m \times n}$ , onde  $m \neq n$ , há uma extensão da decomposição da diagonal principal conhecida como decomposição em valores singulares, do inglês *singular value decomposition* (SVD), de forma que ela pode ser utilizada para construir uma versão aproximada de  $C$  (MANNING; RAGHAVAN; SCHÜTZE, 2008). A decomposição em valores singulares é descrita no Teorema 1, dado que sua prova encontra-se disponível no livro “*Introduction to Information Retrieval*” de Manning, Raghavan e Schütze (2008).

**Teorema 1** Seja  $r$  o rank da matriz  $C_{m \times n}$ , então há uma decomposição em valores singulares de  $C$  na forma:

$$C = U\Sigma V^T \quad (2.8)$$

em que:

1. Os autovalores  $\Lambda_1, \dots, \Lambda_r$  de  $CC^T$  são os mesmos autovalores de  $C^TC$ ;
2. Para  $1 \leq i \leq r$  seja  $\sigma_i = \sqrt{\Lambda_i}$ , com  $\Lambda_i \geq \Lambda_{i+1}$ . Então a matriz  $\Sigma_{m \times n}$  é composta ao definir  $\Sigma_{i,i} = \sigma_i$  para  $1 \leq i \leq r$ , ou zero, caso contrário.

Ao escrever os valores numéricos do SVD, normalmente representa-se  $\Sigma$  como uma matriz  $r \times r$ , com os valores singulares nas diagonais, uma vez que todos os demais elementos fora desta sub-matriz são zeros. Da mesma forma, normalmente também são omitidas as  $M - r$  colunas mais à direita de  $U$ , correspondentes às linhas omitidas de  $\Sigma$ . Da mesma forma, omite-se as  $N - r$  colunas mais à direita de  $V$ , uma vez que elas correspondem às linhas de  $V^T$  que serão multiplicadas pelas  $N - r$  colunas de zeros de  $\Sigma$ . Esta forma convencional do SVD também é conhecida como SVD reduzido (*reduced SVD*) ou SVD truncado (*truncated SVD*) e encontra-se exemplificada na [Figura 4](#).

Como este método não centraliza os valores antes de calcular os valores singulares, diferentemente do PCA, ele acaba por trabalhar de forma mais eficiente com matrizes esparsas.

Figura 4 – Exemplo de decomposição de matriz.

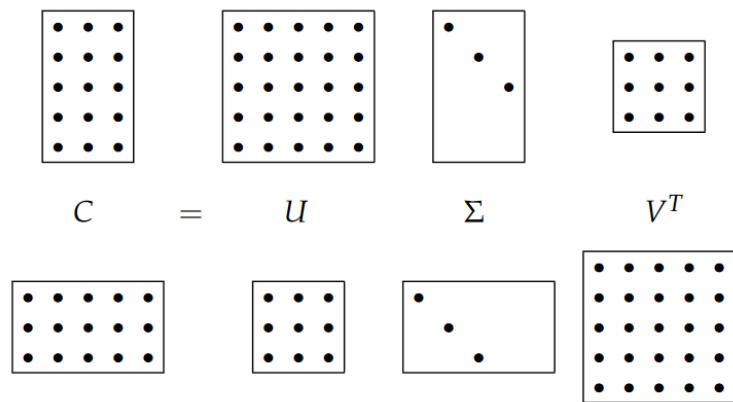


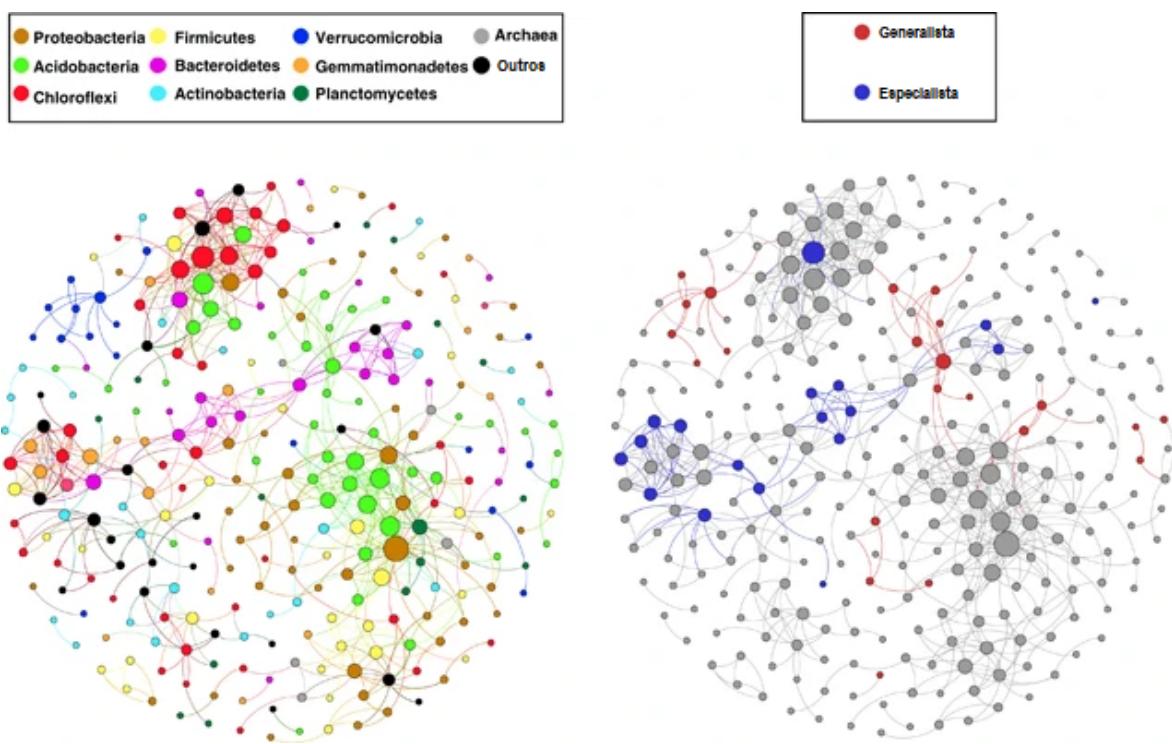
Ilustração da decomposição em valores singulares. Nesta ilustração esquemática da [Equação 2.8](#), é possível visualizar dois casos de decomposição. Na metade superior da figura, temos uma matriz  $C_{m \times n}$ , em que  $m > n$ . Já a metade inferior ilustra o caso em que  $n > m$ . Fonte: ([MANNING; RAGHAVAN; SCHÜTZ, 2008](#))

## 2.7 Grafo/rede de co-ocorrência

Redes de co-ocorrência são normalmente utilizadas para visualizar relações potenciais entre pessoas, organizações, conceitos e organismos biológicos, por exemplo.

No contexto da biologia, por exemplo, há diversos trabalhos que utilizam esta estrutura a fim de analisar padrões de co-ocorrência em comunidades de indivíduos, como é o caso do artigo “Usando análise de rede para explorar padrões de co-ocorrência em comunidades microbiais do solo” ([BARBERÁN et al., 2012](#)).

Figura 5 – Exemplo de projeção de subconjunto de dados.



Uma rede de co-ocorrência com corte de 90% das relações, baseada na análise de componentes. Fonte: ([BARBERÁN et al., 2012](#)).

Pode-se dizer, então, que o grafo de co-ocorrência é a estrutura de dados (grafo) que armazena as relações da rede de co-ocorrência, podendo ser representada graficamente, como no exemplo da [Figura 5](#).

No contexto de PLN, grafos de co-ocorrência são normalmente utilizados para visualizar relações entre palavras (termos) em um dado corpus ou até mesmo entre textos ([KUBEK; UNGER; DUSIK, 2015](#)). Um exemplo clássico é a geração de um grafo onde cada vértice representa um documento (ou termo) e as arestas traçam determinada correlação entre eles (similaridade semântica, por exemplo).

### 3 Coleta de dados

A primeira etapa do trabalho consistiu em adquirir problemas típicos de maratona de programação de algum repositório ou conjunto de dados (*dataset*) pronto disponível na *internet* para que a análise fosse realizada. Porém, como não haviam informações já extraídas e tratadas, foi necessário elaborar uma ferramenta para adquiri-las antes de aplicar a metodologia descrita no [Capítulo 4](#).

Como o foco do trabalho proposto é realizar a análise de problemas típicos de maratona de programação, optou-se por obtê-los do site/plataforma *online URI Online Judge*.

A [seção 3.1](#) explica as características do *web scraper* elaborado (*URI Scraper*), além de detalhar os elementos DOM que ele utiliza para realizar a extração de informação.

A [seção 3.2](#), por sua vez, descreve o funcionamento do *URI Scraper* e como ele foi utilizado para realizar a extração do conjunto de dados utilizado na análise, detalhando cada atributo adquirido.

#### 3.1 Elaboração do *web scraper*

O *scraper* foi desenvolvido em Java utilizando a JSOUP, uma biblioteca que disponibiliza uma API conveniente para extrair e manipular dados de páginas HTML através da manipulação de elementos DOM e CSS com métodos similares ao jQuery (uma biblioteca JavaScript) ([HEDLEY, 2019](#)).

Após adquirir a lista de problemas disponíveis para o idioma selecionado, o *scraper* acessa as páginas específicas de cada um deles, tanto a de descrição, quanto a de estatísticas, a fim de realizar a extração.

A lista de problemas é adquirida através de uma página acessível através seguinte padrão de URL: "<https://www.urionlinejudge.com.br/judge/<LANG>/problems/all>", onde o parâmetro "<LANG>" deve ser substituído pela sigla do idioma desejado: "en", "pt" ou "es", para inglês, português ou espanhol, respectivamente.

Os dados de descrição, entrada e saída dos problemas estão contidos nas *tags* que fazem uso das seguintes classes, respectivamente: "*description*", "*input*" e "*output*". Já os dados estatísticos estão presentes nos valores das *tags* "dd", filhas do elemento "div" cujo *id* é igual a "*problem-statistics*".

A seguir há um exemplo de trecho em HTML contendo a estrutura do elemento onde estão localizadas as informações gerais (descrição, entrada e saída) de cada problema:

Listagem 3.1 – Estrutura de elemento HTML do enunciado de problemas do URI

```

0 <div class="problem">
1   <div class="description">
2     <p>
3       <!-- DESCRICAO DO PROBLEMA -->
4     </p>
5   </div>
6
7   <h2>Input</h2>
8   <div class="input">
9     <p>
10      <!-- DESCRICAO DA ENTRADA -->
11    </p>
12  </div>
13
14  <h2>Output</h2>
15  <div class="output">
16    <p>
17      <!-- DESCRICAO DA SAIDA -->
18    </p>
19  </div>
20  <!-- ... -->
21 </div>
```

E um exemplo de trecho em HTML contendo a estrutura do elemento onde estão localizadas as informações estatísticas de cada problema:

Listagem 3.2 – Estrutura de elemento HTML de estatísticas de problemas do URI.

```

0 <div class="st-small-box" id="problem-statistics">
1   <h3>Statistics</h3>
2   <p><!-- NUMERO DE VEZES RESOLVIDO --></p>
3
4   <dl>
5     <dt>Level:</dt>
6     <dd><!-- NIVEL --> / 10</dd>
7     <dt>Submissions:</dt>
8     <dd><!-- NUMERO DE TENTATIVAS --></dd>
9     <dt>Solved:</dt>
10    <dd><!-- NUMERO DE VEZES RESOLVIDO --></dd>
11    <dt>Ratio:</dt>
12    <dd><!-- PORCENTAGEM DE ACERTO -->%</dd>
13  </dl>
14 </div>
```

A ferramenta desenvolvida levou o nome de *URI Scraper*, uma vez que sua função é extrair os dados dos problemas disponíveis no *URI Online Judge*.

## 3.2 Aquisição de problemas do URI

Utilizando a ferramenta desenvolvida, *URI Scraper*, foi possível obter todos os problemas da plataforma nos três idiomas disponíveis (espanhol, inglês e português).

Conforme adquire os dados, o *scraper* salva-os em um arquivo no formato JSON em um dos seguintes diretórios, dependendo do idioma escolhido para extração:

- “%HOME\_PATH%/URIScraper\_EN”;
- “%HOME\_PATH%/URIScraper\_ES”;
- “%HOME\_PATH%/URIScraper\_PT”.

Cada arquivo JSON contém os dados de um único problema, apresentando o seguinte formato:

Listagem 3.3 – Formato do problema extraído em JSON.

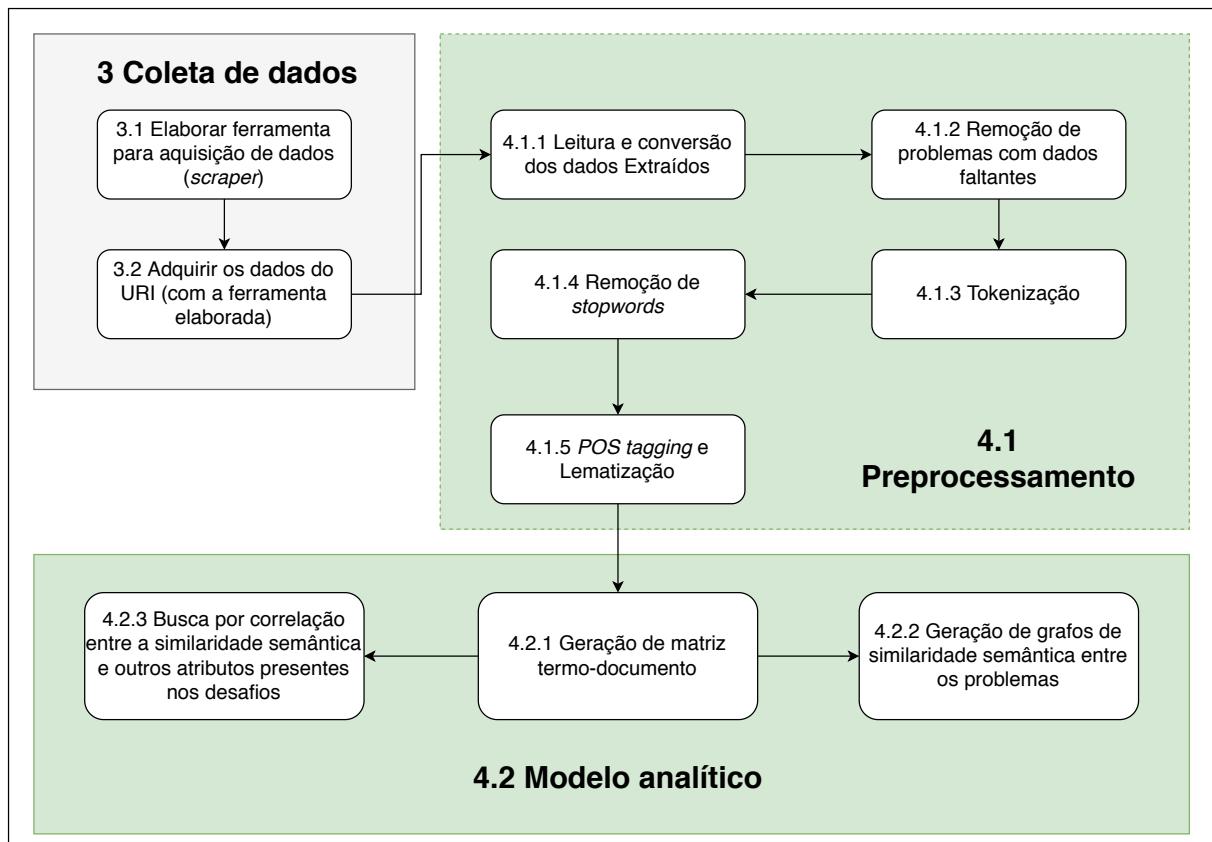
```
0 {
1   "output": "<informacao de dados de saida>",
2   "input": "<informacao de dados de entrada>",
3   "level": <numero>,
4   "name": "<nome>",
5   "has_images": { false / true },
6   "description": "<descricao do problema>",
7   "id": "<id>",
8   "category": "<categoria>",
9   "statistics": {
10     "level": "<numero> / 10",
11     "submissions": <numero>,
12     "solved": <numero>,
13     "ratio": "<numero>%"
14   }
15 }
```

Dada a capacidade de aquisição de dados do scraper, foram extraídos todos os problemas disponíveis em todos os idiomas disponibilizados pelo URI (espanhol, inglês e português), para que a metodologia descrita no [Capítulo 4](#) pudesse ser aplicada.

## 4 Metodologia

A metodologia adotada neste trabalho é descrita pela [Figura 6](#) a seguir:

Figura 6 – Metodologia.



Representação gráfica dos passos tomados no desenvolvimento do projeto. Fonte: Produzido pelo autor.

É importante enfatizar que, embora esteja explicado no [Capítulo 3](#), o grupo “Coleta de dados”, é de extrema importância para a aplicação da metodologia, pois, todo o modelo analítico é construído sobre o conjunto de dados nele obtido.

Ainda conforme abordado no [Capítulo 3](#), a coleta de dados consistiu em elaborar uma ferramenta que extraísse informações de páginas da *web* (*scraper*)<sup>1</sup> a fim de utilizá-las na construção do conjunto de dados (*dataset*).

O primeiro grupo, abordado na [seção 4.1](#), consiste na etapa de preprocessamento do conjunto de dados obtido, onde foram aplicadas diversas técnicas de processamento de

<sup>1</sup> Neste caso, do *URI Online Judge*.

bases utilizando alguns algoritmos de PLN, a fim de limpá-lo e normalizá-lo, permitindo que uma melhor modelagem seja feita e que haja, consequentemente, uma análise mais consistente.

O segundo grupo, descrito na [seção 4.2](#), consiste na etapa de geração do modelo analítico. Nele foram geradas as matrizes termo-documento utilizadas para a realização do cálculo de similaridade semântica entre os problemas, além de serem produzidos os grafos de co-ocorrência e de realizar a inspeção visual, buscando por possíveis formações de agrupamentos que pudessem indicar a existência de alguma correlação entre o enunciado e outros atributos dos problemas.

## 4.1 Preprocessamento

É provável que nem todos os dados adquiridos possuam atributos consistentes ou normalizados. Portanto, é necessário realizar a leitura, limpeza e normalização de todo o conjunto a ser analisado.

Cada subseção a seguir, descreve uma etapa do processo de preprocessamento dos dados obtidos, desde a leitura e conversão à aplicação de POS *tags*.

### 4.1.1 Leitura e conversão dos dados extraídos

A primeira etapa do processo de limpeza consiste em efetuar a leitura dos atributos de todos os problemas em uma estrutura de dados única. É recomendado que a estrutura em questão seja bidimensional, com tamanho mutável e eixos rotuláveis (linhas e colunas), que permita realizar filtros e operações aritméticas de forma simplificada em qualquer eixo (similar a uma entidade relacional) <sup>2</sup>.

É importante notar que, neste passo, os identificadores dos problemas (*ids*) devem ser definidos como índices, permitindo que seus atributos sejam facilmente acessados.

### 4.1.2 Remoção de problemas com dados faltantes

Com os dados já carregados na estrutura, é necessário remover os problemas que não possuem enunciado <sup>3</sup>, ou que não contêm informações de entrada ou de saída. Neste passo, também é recomendado eliminar aqueles que não têm dados estatísticos, para que a análise de agrupamentos (descrita na [subseção 4.2.3](#)) possa ser realizada.

<sup>2</sup> Neste trabalho, especificamente, foi utilizada uma estrutura chamada *DataFrame* ([MCKINNEY, 2019](#)), disponível na Pandas, uma biblioteca que provê estruturas e ferramentas para análise de dados para a linguagem de programação Python ([MCKINNEY et al., 2010](#)).

<sup>3</sup> No caso dos dados extraídos do URI, este problema é normalmente causado pelo fato de haverem desafios que estão disponíveis somente um dos idiomas suportados.

Após realizar esta primeira etapa da limpeza, todos os desafios devem estar com seus 11 (onze) atributos disponíveis:

- `name` : O nome atribuído ao problema;
- `category` : A categoria ao qual o problema pertence;
- `level` : Nível de dificuldade;
- `description` : Descrição do problema;
- `input` : Descrição dos dados/formato de entrada;
- `output` : Descrição dos dados/formato de saída;
- `has_images` : Valor-verdade que indica se há imagens no enunciado;
- `s_level` : Nível de dificuldade (adquirido das estatísticas);
- `submissions` : Quantidade de submissões (adquirido das estatísticas);
- `solved` : Quantidade de vezes que foi, efetivamente, resolvido (adquirido das estatísticas);
- `ratio` : Porcentagem da quantidade de vezes que foi solucionado (adquirido das estatísticas).

#### 4.1.3 Tokenização

A segunda etapa de normalização dos dados consiste em tokenizar tanto o enunciado, quanto as informações de entrada e de saída dos problemas a nível de palavras (cada palavra é definida como um *token*).

Esta etapa pode ser realizada de diversas formas: utilizando expressões regulares, métodos heurísticos, ou ainda a partir de funções disponíveis em bibliotecas específicas <sup>4</sup>.

#### 4.1.4 Remoção de *stopwords*

A terceira etapa, como o próprio nome diz, é onde as *stopwords* são removidas do texto. Porém, como tratamento adicional, é necessário remover todas as pontuações e valores numéricos que possam acabar passando pelo processo de tokenização.

<sup>4</sup> Neste trabalho, foi utilizada a função ‘`word_tokenize`’, disponível na biblioteca NLTK ([NLTK Project, 2019c](#)), que, ao receber um texto arbitrário como argumento, gera uma lista de cadeia de caracteres (*strings*), tal que cada elemento é uma palavra, número ou pontuação presente no texto.

Figura 7 – Alguns enunciados após o processo de tokenização.

	enunciado	entrada	saída
1001	[read, 2, variables, .. named, a, and, b, and...]	[the, input, file, will, contain, 2, integer, ...]	[print, the, letter, x, (, uppercase, ), with,...]
1002	[the, formula, to, calculate, the, area, of, a...]	[the, input, contains, a, value, of, floating,...]	[present, the, message, `` , a=, " , followed, ...]
1003	[read, two, integer, values, .. in, this, case...]	[the, input, file, contains, 2, integer, numbe...]	[print, the, variable, soma, with, all, the, c...]
1004	[read, two, integer, values, .. after, this, ....]	[the, input, file, contains, 2, integer, numbe...]	[print, prod, according, to, the, following, e...]
1005	[read, two, floating, points, ' values, of, d...]	[the, input, file, contains, 2, floating, poin...]	[print, media, (, average, in, portuguese, ), ...]

Parte inicial das listas de palavras obtidas após realizar o processo de tokenização para os problemas 1001 ao 1005 (note que estão inclusos: enunciado, dados de entrada e dados de saída). Fonte: Produzido pelo autor.

Existem diversas listas de palavras classificadas como "*stopwords*" *online*, além de também estarem normalmente disponíveis em bibliotecas dedicadas à realização de tarefas de PLN<sup>5</sup>.

O processo de remoção das *stopwords* consiste em remover todas as palavras que estiverem contidas na *stoplist* ou que não sejam compostas por caracteres do alfabeto latino, propriamente dito<sup>6</sup>.

Ao contrastar o estado atual apresentado na Figura 8 com o exibido na Figura 7, é possível notar que há a remoção de números, símbolos, preposições, artigos etc., melhorando ainda mais a qualidade dos dados utilizados para a realização da análise semântica.

Figura 8 – Alguns enunciados após o processo de remoção de *stopwords*.

	enunciado	entrada	saída
1001	[read, variables, named, b, make, sum, two, va...]	[input, file, contain, integer, numbers]	[print, letter, x, uppercase, blank, space, eq...]
1002	[formula, calculate, area, circumference, defi...]	[input, contains, value, floating, point, doubl...]	[present, message, followed, value, variable, ...]
1003	[read, two, integer, values, case, variables, ...]	[input, file, contains, integer, numbers]	[print, variable, soma, capital, letters, blan...]
1004	[read, two, integer, values, calculate, produc...]	[input, file, contains, integer, numbers]	[print, prod, according, following, example, b...]
1005	[read, two, floating, points, values, double, ...]	[input, file, contains, floating, points,valu...]	[print, media, average, portuguese, according,...]

Parte inicial das listas de palavras obtidas após realizar o processo de remoção das *stopwords* para os problemas 1001 ao 1005. Fonte: Produzido pelo autor.

<sup>5</sup> Novamente, neste caso em específico, foi utilizada a *stoplist* da biblioteca NLTK. A lista em questão pode ser acessada através da função ‘nlkt.corpus.stopwords.words(<lang>)’, que recebe o idioma para o qual a *stoplist* será gerada como argumento (neste caso, ‘english’).

<sup>6</sup> Para cumprir com a segunda condição, foi utilizado o método ‘isalpha()’ disponível para objetos do tipo *string* em Python, que verifica se cada caractere da cadeia é uma letra do alfabeto (Refsnes Data, 2019).

#### 4.1.5 POS tagging e lematização

Para obter um bom resultado no processo de lematização, é recomendado, e às vezes, necessário, que a palavra possua uma classe gramatical atribuída, a fim de evitar ambiguidade no momento em que ela for transformada para sua forma inflexionada.

O processo que envolve aplicar uma "etiqueta" contendo a classe gramatical de cada palavra é conhecido como "*POS tagging*" (do inglês "*Part Of Speech*").

Embora esta tarefa não seja tão trivial, existem alguns modelos já implementados (em diversas linguagens) que a cumprem. Conforme descrito na [subseção 2.1.3](#), estes modelos podem ser divididos em dois grupos de algoritmos que fazem essa classificação, os baseados em regra e os estocásticos. Porém, neste caso, qualquer "*POS tagger*" eficiente poderá ser utilizado <sup>7</sup>.

Figura 9 – Alguns enunciados após o processo de aplicação de classificação gramatical.

	enunciado	entrada	saída
1001	[(read, JJ), (variables, NNS), (named, VBN), (...]	[(input, NN), (file, NN), (contain, NN), (inte...]	[(print, NN), (letter, NN), (x, NNP), (upperca...
1002	[(formula, NN), (calculate, NN), (area, NN), (...]	[(input, NN), (contains, VBZ), (value, NN), (f...]	[(present, JJ), (message, NN), (followed, VBD)...
1003	[(read, VB), (two, CD), (integer, JJR), (value...]	[(input, NN), (file, NN), (contains, VBZ), (in...]	[(print, NN), (variable, JJ), (soma, NN), (cap...
1004	[(read, VB), (two, CD), (integer, JJR), (value...]	[(input, NN), (file, NN), (contains, VBZ), (in...]	[(print, NN), (prod, NN), (according, VBG), (f...
1005	[(read, VB), (two, CD), (floating, VBG), (poin...]	[(input, NN), (file, NN), (contains, VBZ), (fl...]	[(print, NN), (media, NNS), (average, JJ), (po...

Parte inicial das listas de palavras obtidas após realizar o processo de aplicação de *POS tags* para os problemas 1001 ao 1005. Fonte: Produzido pelo autor.

Dependendo da implementação do lematizador que será utilizado, pode ser necessário fazer a conversão das *tags* atribuídas às palavras pelo *POS tagger* para as suportadas pelo algoritmo de lematização, como é o caso do WordNet, que trabalha apenas com quatro tipos de classes gramaticais: adjetivo, substantivo, verbo e advérbio ([NLTK Project, 2019a](#)).

Um exemplo de regra de conversão de *tags* do classificador *Penn Treebank* a fim de que sirva como entrada para o lematizador do *WordNet*, é a seguinte (note que ‘<TAG>’ é a classificação de acordo com as regras do *Penn Treebank*):

<sup>7</sup> Neste trabalho, foi utilizado o algoritmo padrão da NLTK, bastando chamar a função ‘nltk.pos\_tagger(<tokenized\_words>)’ que recebe uma lista de palavras, preferencialmente tratadas, e retorna uma lista de tuplas que contêm a palavra e sua respectiva classificação gramatical, cujos possíveis valores são descritos na [subseção 2.1.3](#). De acordo com a documentação da biblioteca, faz uso do conjunto de *tags* disponível no *Penn Treebank* ([NLTK Project, 2019b](#)), um conjunto de regras que define os tipos de classes gramaticais e como classificar as palavras em alguma delas ([SANTORINI, 1990](#)).

$$\text{Classificação} = \begin{cases} \text{Adjetivo,} & \text{se } <\text{TAG}> \text{ começa com 'J'} \\ \text{Verbo,} & \text{se } <\text{TAG}> \text{ começa com 'V'} \\ \text{Advérbio,} & \text{se } <\text{TAG}> \text{ começa com 'R'} \\ \text{Substantivo,} & \text{caso contrário} \end{cases}$$

Já com as palavras propriamente classificadas, é possível lematizá-las, ou seja, transformar cada uma delas para sua respectiva forma inflexionada. Desta forma, evita-se que a mesma palavra com conjugações diferentes não sejam consideradas semanticamente iguais.<sup>8</sup>.

De acordo com a Figura 10, já é possível notar que as palavras foram transformadas para suas formas inflexionadas, como é o caso da palavra ‘*according*’, a terceira do problema 1004, que foi convertida em ‘*accord*’.

Figura 10 – Alguns enunciados após o processo de aplicação de lematização.

	enunciado	entrada	saída
1001	[(read, JJ), (variable, NNS), (name, VBN), (b,...	[[(input, NN), (file, NN), (contain, NN), (inte...	[[(print, NN), (letter, NN), (x, NNP), (upperca...
1002	[(formula, NN), (calculate, NN), (area, NN), (...	[[(input, NN), (contain, VBZ), (value, NN), (fl...	[[(present, JJ), (message, NN), (follow, VBD), ...
1003	[(read, VB), (two, CD), (integer, JJR), (value...	[[(input, NN), (file, NN), (contain, VBZ), (int...	[[(print, NN), (variable, JJ), (soma, NN), (cap...
1004	[(read, VB), (two, CD), (integer, JJR), (value...	[[(input, NN), (file, NN), (contain, VBZ), (int...	[[(print, NN), (prod, NN), (accord, VBG), (fol...
1005	[(read, VB), (two, CD), (float, VBG), (point, ...	[[(input, NN), (file, NN), (contain, VBZ), (flo...	[[(print, NN), (medium, NNS), (average, JJ), (p...

Parte inicial das listas de palavras obtidas após realizar o processo de lematização para os problemas 1001 ao 1005. Fonte: Produzido pelo autor.

## 4.2 Modelo analítico

O modelo deve representar a similaridade semântica entre todos os problemas disponíveis (os que restaram após a limpeza realizada no preprocessamento). Para realizar tal tarefa, é necessário calcular a distância entre cada um deles, utilizando a métrica do cosseno, a partir de matrizes termo-documento.

Tendo calculado a similaridade entre os problemas, é possível gerar um grafo de co-ocorrência, a fim de visualizar de forma gráfica “o quanto forte” é a similaridade entre cada um deles.

Por fim, é possível estudar expansões de pesquisa a partir do modelo elaborado, como a correlação entre a similaridade semântica entre problemas e os demais atributos

<sup>8</sup> Para tal, foi utilizado o lematizador *WordNet*, um grande banco de dados da língua inglesa (superficialmente, o *WordNet* lembra um tesauro, uma vez que agrupa as palavras baseado no significado delas) (FELLBAUM, 1998).

disponíveis, ou ainda um sistema de sugestões, que considere os enunciados dos problemas mais similares entre si.

As subseções a seguir descrevem cada um dos passos a serem tomados para a elaboração do modelo.

#### 4.2.1 Geração de matriz termo-documento

Ao todo, são construídas seis matrizes termo-documento distintas, duas para cada um dos atributos textuais dos problemas (uma, que leva em consideração a contagem total de palavras, descrita na [subseção 2.1.4](#) e outra que utiliza a técnica TF-IDF, de acordo com a [subseção 2.1.5](#)):

- Descrição do problema;
- Informação dos dados de entrada (*input*);
- Informação dos dados de saída (*output*).

Para gerá-las, é necessário seguir os seguintes passos:

1. Criar três conjuntos de palavras ( $\mathcal{D}$ ,  $\mathcal{E}$  e  $\mathcal{S}$ ) contendo o vocabulário das descrições de todos os problemas, das informações de dados de entrada e das informações de dados de saída, respectivamente;
2. Gerar três matrizes ( $M_{|\mathcal{D}| \times |\mathcal{P}|}$ ,  $N_{|\mathcal{E}| \times |\mathcal{P}|}$  e  $O_{|\mathcal{S}| \times |\mathcal{P}|}$ ), uma para cada conjunto, cujas colunas representam os problemas (conjunto de documentos, representados por  $\mathcal{P}$ ) e as linhas representam as palavras dos vocabulários (termos). Também é possível tratar estas matrizes como o produto cartesiano entre o conjunto de termos e o de documentos (vide [Figura 11](#));
3. Utilizando as matrizes geradas no passo anterior, é possível derivar três novas matrizes:  $P_{|\mathcal{D}| \times |\mathcal{P}|}$ ,  $Q_{|\mathcal{E}| \times |\mathcal{P}|}$  e  $R_{|\mathcal{S}| \times |\mathcal{P}|}$ , sendo estas obtidas através do cálculo da frequência do termo pelo inverso da frequência nos documentos (TF-IDF). Para calcular a  $tf \times idf$  (termo-frequência pelo inverso da frequência nos documentos) de cada termo, são utilizadas as respectivas equações: [Equação 2.3](#) (TF) e [Equação 2.6](#) (IDF);
4. A fim de ganhar desempenho nas operações de cálculo vetorial (distância do cosseno), as matrizes podem ser transformadas em uma série de vetores-linha, ou seja, cada vetor corresponde à frequência de palavras do problema. Para simplificar o acesso aos vetores, recomenda-se indexá-los pelo número do problema que representam, através de um mapa de dispersão (estrutura de dados também conhecida como *Hash Map*), vide [Figura 12](#).

Figura 11 – Matriz termo-documento.

	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	...	2952	2953	2954	2955	2956	2957	2958	2959	2960	2961
affect	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
similarity	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
miraculous	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
cheapest	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
boll	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Parte da matriz esparsa de frequência (termo-documento) para a descrição dos problemas ( $M_{|\mathcal{D}| \times |\mathcal{P}|}$ ) obtida ao término do passo 2. Fonte: Produzido pelo autor.

Figura 12 – Conjunto de vetores de frequência para os problemas.

```

{'1001': array([0, 0, 0, ..., 0, 0, 0]),
 '1002': array([0, 0, 0, ..., 0, 0, 0]),
 '1003': array([0, 0, 0, ..., 0, 0, 0]),
 '1004': array([0, 0, 0, ..., 0, 0, 0]),
 '1005': array([0, 0, 0, ..., 0, 0, 0]),
 '1006': array([0, 0, 0, ..., 0, 0, 0]),
 '1007': array([0, 0, 0, ..., 0, 0, 0]),
 '1008': array([0, 0, 0, ..., 0, 0, 0]),
 '1009': array([0, 0, 0, ..., 0, 0, 0]),
 '1010': array([0, 0, 0, ..., 0, 0, 0])}

```

Parte do conjunto de vetores obtidos da decomposição horizontal (eixo x) da matriz termo-documento de descrição dos problemas ( $M_{|\mathcal{D}| \times |\mathcal{P}|}$ ), tal que cada índice do vetor corresponde à frequência de algum termo no problema em questão.

#### 4.2.2 Geração de grafos de similaridade semântica entre os problemas

Para que os grafos sejam gerados, é necessário calcular a similaridade entre os problemas. Para tal, devem ser feitas  $\sum_{i=1}^{p-2} (p - i)$  comparações, tal que ‘p’ é um inteiro que representa a quantidade de problemas analisados.

A métrica utilizada para realizar os cálculos de similaridade é a distância do cosseno, dada pela seguinte fórmula (‘u’ e ‘v’ são os vetores-coluna obtidos da matriz termo-documento ou TF-IDF):

$$d = 1 - \frac{u \cdot v}{|u| |v|} \quad (4.1)$$

Interpretando os possíveis resultados da Equação 4.1, que estão contidos no intervalo  $[0, 1]$ , uma vez que  $d: \mathcal{N}^{|p|} \rightarrow [0, 1]$ , é possível notar que, quanto menor for ‘d’, mais similares são os problemas comparados.

Os resultados das comparações devem ser armazenados em uma estrutura de dados que facilite o acesso ao cálculo da distância entre os problemas <sup>9</sup>.

<sup>9</sup> A fim de facilitar a geração do grafo de co-ocorrência, foi utilizado um mapa de dispersão onde

A estrutura de dados obtida (matriz, mapa de dispersão, etc.) deve ser suficiente para gerar o grafo desejado <sup>10</sup>. Caso a quantidade de nós (problemas) seja muito grande, é possível definir um limiar para que a relação seja adicionada à figura.

A próxima etapa consiste em também considerar as descrições de entrada e de saída dos problemas para, então, gerar novos grafos, com as mesmas propriedades dos anteriores. Porém, como  $|\mathcal{E}|, |\mathcal{S}| \ll |D|$ , eles devem ser “penalizados”, de forma que seus pesos sejam diretamente proporcionais aos seus tamanhos. Uma possível forma de penalização é a atribuir o peso  $P$  proporcional ao tamanho de cada conjunto, conforme descrito nas fórmulas a seguir:

$$P_{max} = |\mathcal{D}| + |\mathcal{E}| + |\mathcal{S}| \quad (4.2)$$

$$P_E = \frac{|\mathcal{E}|}{P_{max}} \quad (4.3)$$

$$P_S = \frac{|\mathcal{S}|}{P_{max}} \quad (4.4)$$

$$P_D = 1 - (P_E + P_S) \quad (4.5)$$

em que  $P_E$ ,  $P_S$  e  $P_D$  são os pesos das descrições de entrada, de saída e do problema, respectivamente. Logo, a distância é calculada utilizando a seguinte fórmula:

$$Dist = d(D_i, D_j) * P_D + d(E_i, E_j) * P_E + d(S_i, S_j) * P_S \quad (4.6)$$

em que  $D$ ,  $E$ ,  $S$  são os vetores que descrevem a frequência dos termos dos vocabulários da descrição do problema, dos dados de entrada e de saída, respectivamente, e  $d(u, v)$  é a equação responsável por calcular a distância entre os vetores, descrita na [Equação 4.1](#).

Devido à remoção de *stopwords* e valores cujos caracteres não pertençam ao alfabeto latino, alguns problemas perderam as descrições de entrada e/ou de saída por completo e, a fim de evitar casos de divisão por zero no momento do cálculo da distância do cosseno, eles tiveram suas distâncias calculadas de acordo com a seguinte regra (derivada da [Equação 4.6](#)):

$$Dist = d(D_i, D_j) * P_D + h(E_i, E_j) * P_E + h(S_i, S_j) * P_S \quad (4.7)$$

---

a chave é o número de identificação do problema e o valor consistem em uma lista de tuplas “(numero\_de\_outro\_problema, d)” ordenados de forma crescente, utilizando o segundo elemento das tuplas (distância entre eles) como critério.

<sup>10</sup> Existem diversos recursos e bibliotecas disponíveis online que facilitam a geração de um grafo, tais como o GraphViz (DOT) e a NetworkX, uma biblioteca para Python.

em que  $h(v_1, v_2)$  é dado por:

$$h(v_1, v_2) = \begin{cases} 1, & \text{se } \left( \sum_{i=0} v_1[i] = 0 \wedge \sum_{i=0} v_2[i] \neq 0 \right) \vee \left( \sum_{i=0} v_1[1] \neq 0 \wedge \sum_{i=0} v_2[i] = 0 \right) \\ d(v_1, v_2), & \text{caso contrário} \end{cases} \quad (4.8)$$

É importante notar que para cada uma dessas três estruturas que armazenam a similaridade entre os problemas, também foi gerada uma estrutura correspondente, onde ao invés de utilizar a matriz termo-documento padrão, foi utilizada a matriz TF-IDF.

#### 4.2.3 Busca por correlação entre a similaridade semântica e outros atributos presentes nos desafios

A parte final da análise consiste em buscar padrões visuais que possam identificar possíveis correlações entre a similaridade semântica e três outros atributos disponíveis nos problemas contidos no conjunto de dados analisado: categoria, nível de dificuldade e taxa de resolução.

Será observado o comportamento da distribuição espacial dos vetores-documento, buscando identificar se possíveis agrupamentos (*clusters*) podem ser explicados por algum dos três outros atributos anteriormente mencionados. Porém, como os vetores estão em uma dimensão não observável ( $|\mathcal{D}|$ -ésima), torna-se necessário reduzi-la (para duas ou três dimensões, as quais são possíveis visualizar).

A fim de obter os melhores resultados possíveis, é recomendado que vários métodos sejam utilizados, principalmente devido às características dos vetores que estão sendo analisados, uma vez que eles tendem a ser muito esparsos e seus dados dificilmente representarem valores lineares.

O primeiro método a ser aplicado é o PCA, pois, embora não seja o mais recomendado para o cenário atual, pelo fato de as matrizes fornecidas como entrada serem esparsas e não apresentarem dados necessariamente lineares, ele é um dos algoritmos mais utilizados para fins de redução de dimensionalidade.

O segundo algoritmo a ser utilizado é o MDS, uma vez que, como uma das características das matrizes obtidas é a não linearidade dos dados, este método pode lidar melhor com elas, uma vez que ele busca por uma representação clara dos dados em uma dimensão menor, respeitando as distâncias existentes entre eles no espaço dimensional original. Sendo assim, ele evita sobreposição de pontos, permitindo uma visualização mais clara da distribuição dos atributos dos desafios analisados (PEDREGOSA et al., 2020b).

Já o terceiro, e último, algoritmo sugerido é o SVD, uma vez que ele tende a trabalhar com matrizes esparsas de forma mais eficiente (como é o caso de ambas as matrizes termo-

documento geradas), uma vez que, diferentemente do PCA, ele não centraliza os dados antes de computar a decomposição dos valores singulares [Pedregosa et al. \(2020a\)](#).

Após reduzir as dimensões, basta observar a dispersão alterando a cor dos pontos (ou o formato deles) de acordo com o atributo a ser analisado. Novamente, recomenda-se que o procedimento seja feito tanto com a matriz termo-documento, quanto com a versão TF-IDF, pois, a dispersão espacial pode acabar variando.

# 5 Resultados e análises

A análise de resultados consiste em três grandes partes principais, onde a primeira é a análise da frequência de classificações gramaticais presentes nos problemas (documentos), disponível na [seção 5.1](#); a segunda é a análise dos grafos de co-ocorrência gerados com base nos dados dos cálculos de similaridade semântica entre os textos, disponível na [seção 5.2](#) e a terceira é a análise da distribuição espacial dos problemas, disponível na [seção 5.3](#).

## 5.1 Classificações gramaticais

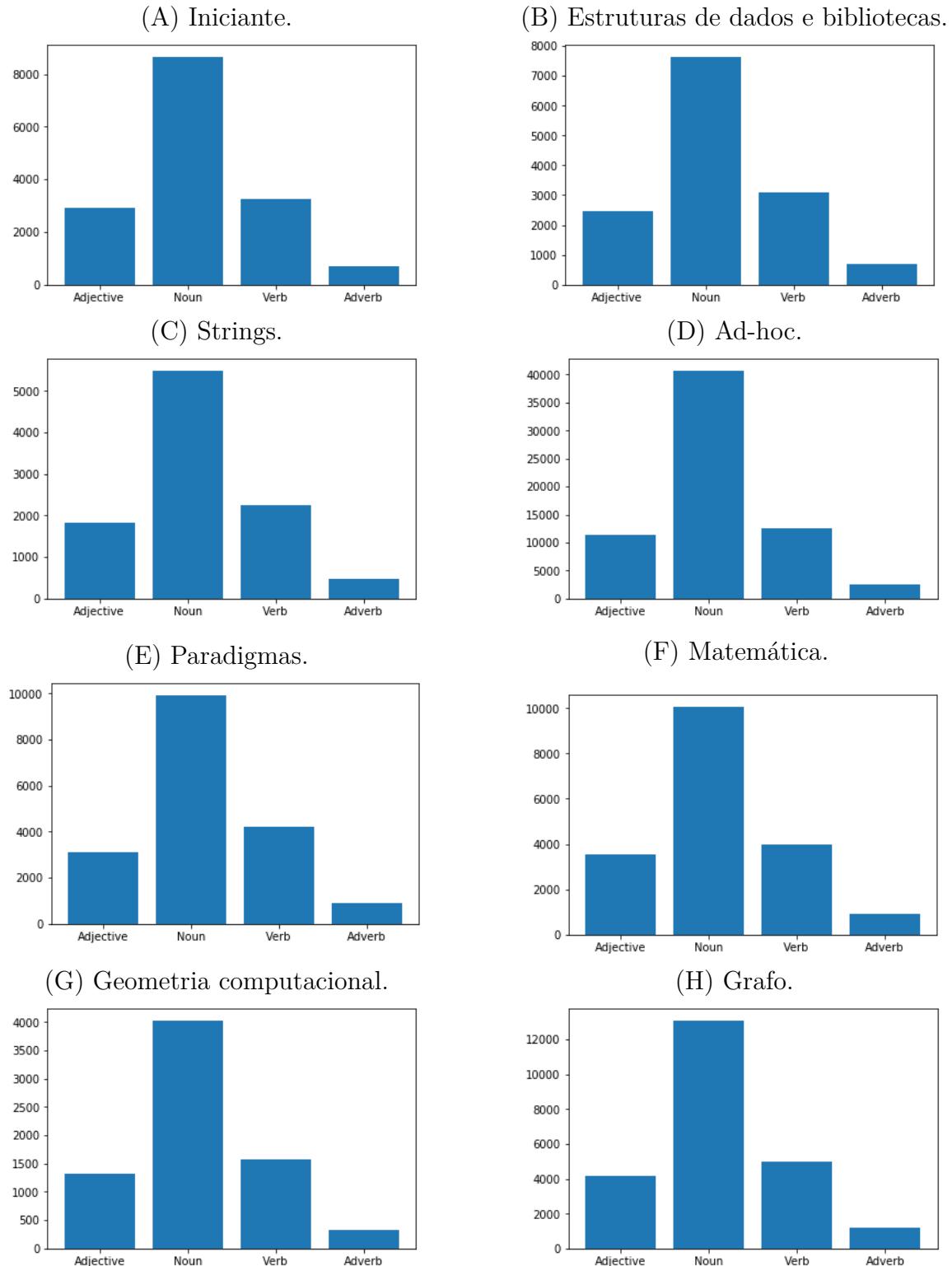
A primeira análise realizada sobre as descrições dos problemas foi com relação às classes gramaticais presentes em cada categoria de desafio. Para tal, foram gerados histogramas para cada uma delas, considerando apenas as quatro classes utilizadas pelo lematizador.

Como é possível observar na [Figura 13](#), as classes às quais pertencem a maioria e a minoria das palavras, independentemente da categoria foram ‘substantivo’ e ‘advérbio’, respectivamente. Quanto às outras duas (‘adjetivo’ e ‘verbo’), ambas tendem a aparecer com a mesma frequência, embora em todas as classes houvessem mais verbos que adjetivos.

Dentre as possíveis causas para a grande quantidade de substantivos, uma delas é o fato de que, no momento em que a “etiquetação” foi realizada, palavras que não possuíam classificação própria, foram tratadas como se fossem pertencentes à classe “substantivo”. Outra possível causa, que também está relacionada ao uso da classe “substantivo” como valor-padrão, está associada ao momento da conversão das classes gramaticais específicas do *Penn Treebank* (vide [Tabela 6](#)) para as quatro classes analisadas (antes de aplicar o lematizador), ou seja, certamente há palavras que não são substantivos mas que foram classificadas como tal.

Quanto ao tamanho do vocabulário, é possível notar que a categoria ‘ad-hoc’ é a que apresenta maior variabilidade de palavras (maior vocabulário), enquanto a ‘geometria computacional’ tem o menor vocabulário dentre todas elas. Isso pode ser explicado pelo fato de a maioria dos desafios estarem categorizados como pertencentes à ‘ad-hoc’, classe que, por sua própria natureza, apresenta maior diversidade de situações-problema, logo, podem ser utilizados termos de diversas naturezas para descrevê-las.

Figura 13 – Histograma de classes gramaticais de acordo com as categorias dos problemas.



Fonte: Produzido pelo autor.

## 5.2 Grafos de co-ocorrência

Para gerar os grafos, bastou utilizar a estrutura obtida na [subseção 4.2.2](#) como insumo para a biblioteca “*graphviz*”, que cria um arquivo com informações do grafo no formato “.dot”<sup>1</sup>, a fim de obter a visualização gráfica apenas dos problemas que possuíam maior similaridade semântica entre si (distância  $\leq 0.3$ , devido a limitações computacionais).

Para gerar a visualização das correlações entre problemas que apresentem ao menos 70% de similaridade semântica entre si ([Figura 39](#) e [Figura 40](#)), foi utilizada uma classe auxiliar chamada “Graph” (implementada manualmente), responsável por exportar outros arquivos de grafo com formato “.gdf” para que sejam renderizados no “Gephi”<sup>2</sup>, uma vez que o GraphViz tende a ficar muito lento ao trabalhar com grafos muito grandes, além de produzir imagens muito esparsas.

Diversos atributos do grafo foram personalizados, a fim de enriquecer a experiência visual de quem o observa e analisa. São eles:

- **Texto do vértice:** Apresenta o formato “< ID > [< CAT >]”, onde “< ID >” representa o nome/identificação do problema e “< CAT >” sua respectiva categoria;
- **Cor do vértice:** As cores dos vértices são dadas de acordo com o nível de dificuldade do problema, que varia de um a dez. A paleta de cores escolhida foi a divergente Azul-Vermelho (também chamada de “BuRd”)<sup>3</sup>, onde os problemas “mais fáceis” são representados pelos tons azuis, enquanto os mais difíceis pelos tons vermelhos;
- **Peso, espessura e texto da aresta:** Tanto o peso, quanto a espessura e o texto das arestas (relações) são dados pela similaridade semântica que dois problemas distintos (nós) apresentam entre si. Diferentemente do texto, que apresenta valores reais, os valores de peso e espessura das arestas variam de 1 a 6, onde 1 é para problemas menos similares entre si e 6 para problemas idênticos.

Após renderizar os grafos, para o primeiro caso (utilizando a matriz termo-documento padrão<sup>4</sup> e considerando apenas as descrições dos problemas), foram obtidas diversas componentes conexas, de forma que a maior delas pode ser observada na [Figura 14](#).

É notável que, além de possuírem níveis de dificuldade similares (todos estão em azul), todos os problemas presentes nesta componente, que também é completa, estão

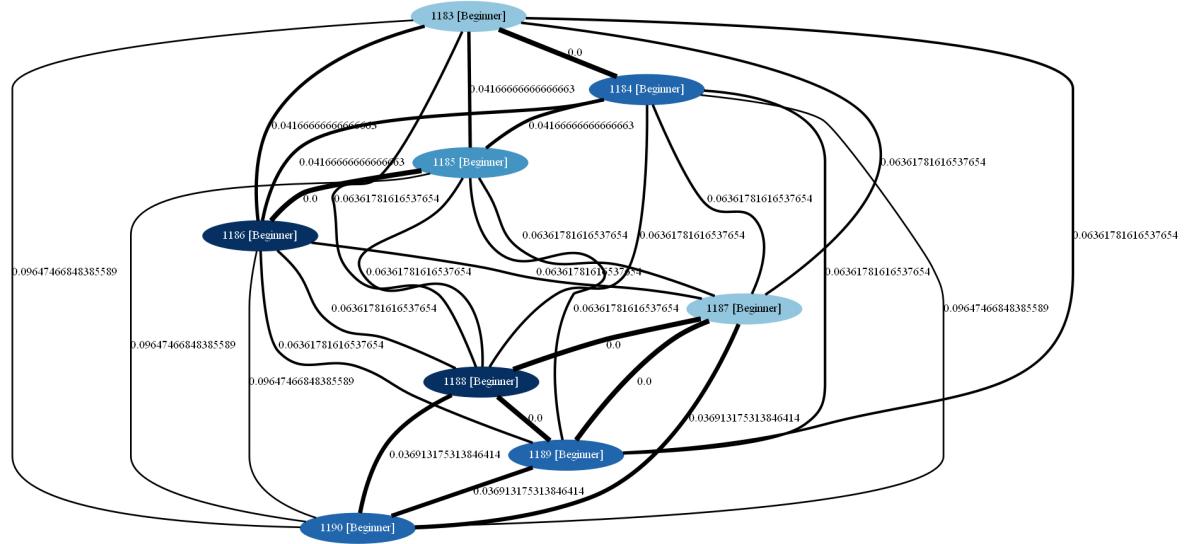
<sup>1</sup> Um software de visualização de grafos de código aberto ([ELLSON et al., 2003](#)).

<sup>2</sup> Uma ferramenta analítica utilizada para a criação e manipulação de grafos e redes ([BASTIAN et al., 2009](#)).

<sup>3</sup> A paleta em questão foi escolhida utilizando a ferramenta ColorBrewer, uma ferramenta que facilita o trabalho da escolha de esquemas de cores para cartografia ([HARROWER; BREWER, 2003](#)). O esquema Azul-Vermelho divergente foi escolhido por ser um dos que não causam problemas para pessoas daltônicas distinguirem as cores nele presentes.

<sup>4</sup> Como a maior componente conexa do grafo resultante ao utilizar a matriz TF-IDF foi muito similar ao representado na [Figura 14](#), optou-se por mover sua representação gráfica para o apêndice ([Figura 20](#)).

Figura 14 – Maior componente conexa do grafo de co-ocorrência (considerando apenas a descrição do problema).



Fonte: Produzido pelo autor.

classificados como ‘*beginner*’ (‘iniciante’), onde o motivo de tal coincidência será discutido a seguir.

Ao verificar o conteúdo das descrições, percebe-se que eles são, realmente, muito similares. Os problemas 1183 e 1184, por exemplo, diferem por uma única palavra, uma vez que o primeiro deles pede para calcular a somatória ou a média dos elementos acima da diagonal principal de uma matriz, enquanto o segundo pede para realizar a mesma operação, entretanto, com os elementos contidos abaixo da diagonal principal da matriz, conforme é possível observar na [Listagem 5.1](#) e na [Listagem 5.2](#). A similaridade entre estes problemas é tanta que a distância calculada entre eles foi de aproximadamente zero. Essa distância acabou por ser ainda mais reduzida devido ao processo de remoção de *stopwords*, que acabou por retirar a palavra ‘of’ do vocabulário analisado.

**Listagem 5.1 – Descrição do problema 1183 do URI. [Tonin e URI Online Judge \(2012a\)](#)**

Read an uppercase character that indicates an operation that will be performed in an array  $\mathbf{M}[12][12]$ . Then, calculate and print the sum or average considering only that numbers that are above the main diagonal of the array, like shown in the following figure (green area).

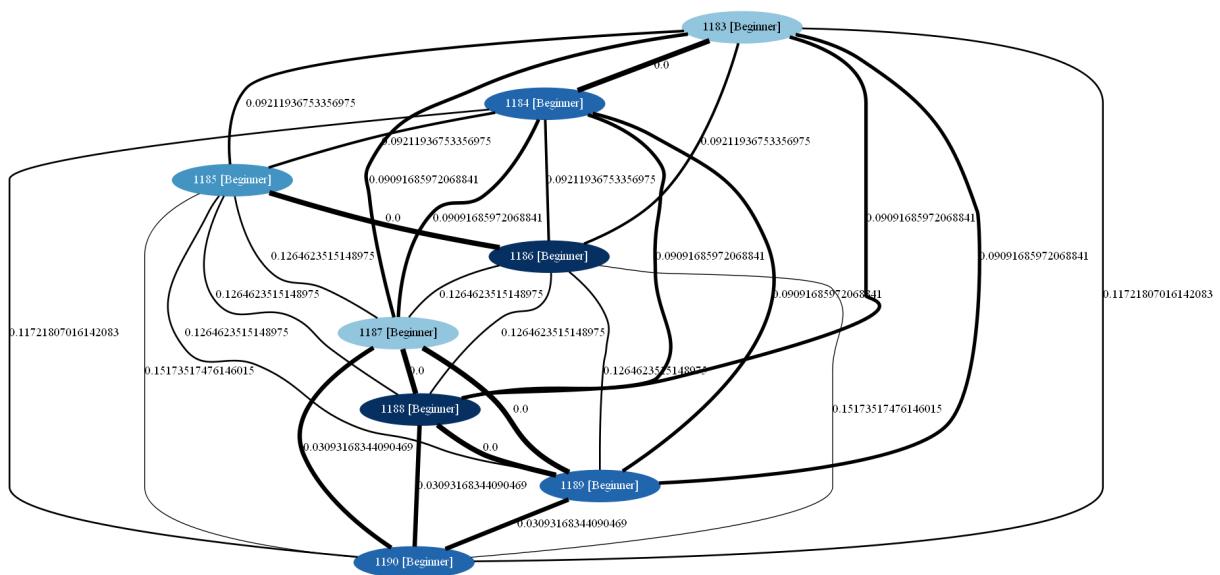
**Listagem 5.2 – Descrição do problema 1184 do URI. [Tonin e URI Online Judge \(2012b\)](#)**

Read an uppercase character that indicates an operation that will be performed in an array  $\mathbf{M}[12][12]$ . Then, calculate and print the sum or average considering only that numbers that are below of the main diagonal of the array, like shown in the following figure (green area).

Ainda analisando a [Figura 14](#), ao verificar as descrições de cada um dos problemas indicados nos vértices, confirma-se que a proximidade entre eles é inegável, uma vez que todos consistem em efetuar operações sobre diagonais de matrizes, além do fato de apresentarem vocabulários extremamente similares ou idênticos (após o preprocessamento).

Já no segundo grafo gerado, que também considera as descrições dos dados de entrada e saída dos desafios, embora tenha mantido a maior componente conexa, representada na [Figura 15](#), é possível notar algumas diferenças, como é o caso dos problemas 1188 e 1189, cuja distância entre eles cresceu de 0 a aproximadamente 0,005, isso devido ao fato de, na descrição dos dados de entrada do desafio 1188, haver a informação de que a matriz deve possuir números do tipo “*double*”.

[Figura 15](#) – Maior componente conexa do segundo grafo de co-ocorrência (considerando a descrição do problema, dos dados de entrada e de saída).



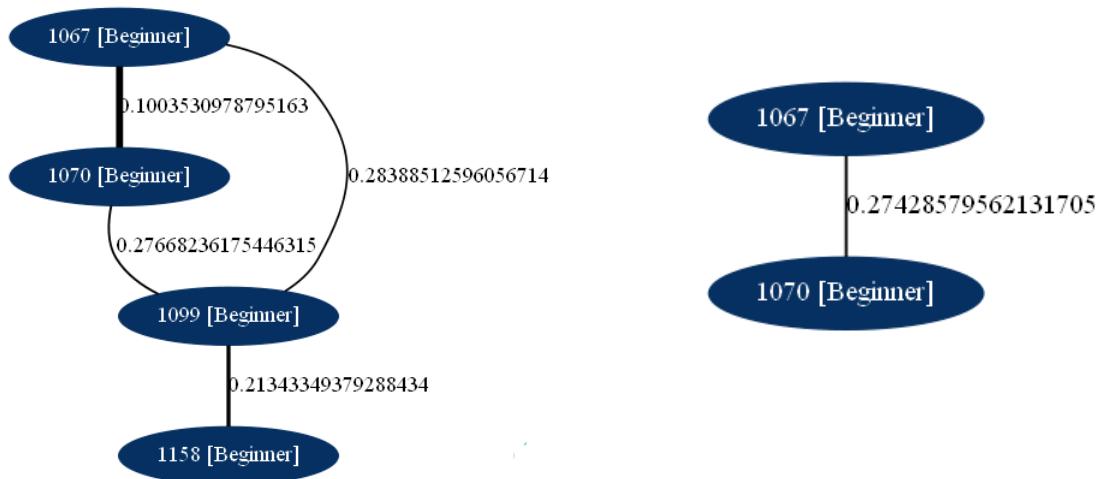
Fonte: Produzido pelo autor.

Ao analisar outras componentes conexas em ambos os grafos renderizados, é possível notar que, algumas vezes, considerar a descrição de dados de entrada e de saída fez com que a distância de similaridade entre os problemas ficasse muito maior, como é o caso apresentado na [Figura 16](#). Com os dados apresentados nestas componentes, é possível perceber que a distância de similaridade aumentou não só entre os problemas 1067-1099 e 1070-1099, mas entre os problemas 1067 e 1070 também, onde passou de cerca de 0.1003, para aproximadamente 0.2743, ou seja, quase o triplo da distância inicial.

De fato, embora o texto de descrição da entrada seja relativamente similar, não é possível dizer o mesmo do texto de descrição da saída, conforme é possível observar na [Listagem 5.3](#) e [Listagem 5.4](#).

Figura 16 – Comparaçāo de componentes conexas do grafo.

(A) Sem considerar entrada e saída. (B) Considerando entradas e saídas.



Exemplo de desafios cujos valores de similaridade foram altamente afetados ao considerar as descrições dos dados de entrada e de saída. Fonte: Produzido pelo autor.

Listagem 5.3 – Descrição de entrada e saída do problema 1067. [Tonin e URI Online Judge \(2012c\)](#)

**Input**

The input will be an integer value.

**Output**

Print all odd values between 1 and  $x$ , including  $x$  if it is the case.

Listagem 5.4 – Descrição de entrada e saída do problema 1070. [Tonin e URI Online Judge \(2012d\)](#)

**Input**

The input will be a positive integer value.

**Output**

The output will be a sequence of six odd numbers.

Pelo fato de os textos serem relativamente pequenos, qualquer diferença no vocabulário causa um grande impacto no cálculo de similaridade entre eles, principalmente pela natureza da métrica utilizada (distância do cosseno).

### 5.3 Distribuição espacial e busca por correlações

Por fim, foi realizada uma análise de inspeção visual a fim de buscar padrões de distribuição espacial que pudessem indicar a existência de alguma correlação entre a similaridade semântica dos textos com os demais atributos presentes nos desafios.

Conforme mencionado na [subseção 4.2.3](#), foi necessário aplicar técnicas de redução de dimensionalidade a fim de que a distribuição espacial dos problemas (com relação às descrições) pudessem ser representadas graficamente.

Dado que os atributos adicionais dos problemas extraídos foram: ‘categoria’, ‘nível de dificuldade’, ‘possui imagens na descrição’ (booleano), ‘submissões’, ‘submissões corretas’ e ‘taxa de resolução’ ( $\frac{\text{submissões corretas}}{\text{submissões}}$ ), é possível afirmar que apenas três deles são realmente passíveis de análise de aglomeração: ‘categoria’, ‘nível de dificuldade’ e ‘taxa de resolução’.

Tanto o atributo ‘submissões’, quanto o ‘submissões corretas’ não foram analisados, pois, por si só, eles tendem a apresentar valores arbitrários e ilimitados, além de não serem passíveis de normalização. Porém, eles são essenciais para compor o ‘taxa de resolução’, um valor normalizado e que traz uma informação estatística potencialmente útil para análise. Já o atributo ‘possui imagens na descrição’ não foi utilizado, pois, parte dos desafios no qual ele possui o valor-verdade definido como ‘verdadeiro’ foram removidos no preprocessamento, uma vez que a descrição estava contida na própria imagem.

Para cada um dos três atributos visualizados foram utilizadas três diferentes técnicas de redução de dimensionalidade para ambas as versões de matrizes termo-documento geradas: PCA, MDS e SVD.

### 5.3.1 Categoria

O primeiro atributo a ser testado foi a categoria dos desafios, de forma que os resultados obtidos podem ser observados na [Figura 17](#)<sup>5</sup>. Nela é possível notar que, aparentemente, não há qualquer indício de possíveis agrupamentos claros por parte das categorias, embora seja possível fazer duas observações iniciais: a primeira é que, para a matriz termo-documento, a aplicação do SVD não melhorou tanto a visualização gráfica quanto o esperado, quando comparada à aplicação do PCA<sup>6</sup>; a segunda é o comportamento da aplicação das técnicas de redução de dimensionalidade sobre a matriz TF-IDF, que apresenta melhor dispersão dos dados em um ambiente bidimensional.

Embora não haja indício de grandes agrupamentos, conforme mencionado anteriormente, de certa forma, a categoria “iniciante” (“beginner”) acaba destacando-se das demais em alguns casos, conforme é possível observar nos itens B, C e F da [Figura 17](#). É evidente que em algumas regiões existem apenas problemas pertencentes a ela, como no caso dos itens C e F, onde, enquanto no item C ela acaba apresentando um aglomerado notável e consistente na região central da figura, no item F, desafios que acabaram por ser representados na parte superior do gráfico (cujo valor do eixo y são maiores que 0.1) também são, em sua maioria, pertencentes a esta categoria. Desta forma, é possível afirmar

<sup>5</sup> Para melhor visualização dos gráficos, eles também foram adicionados ao apêndice deste trabalho ([Figura 21](#), [Figura 22](#), [Figura 23](#), [Figura 24](#), [Figura 25](#) e [Figura 26](#)) .

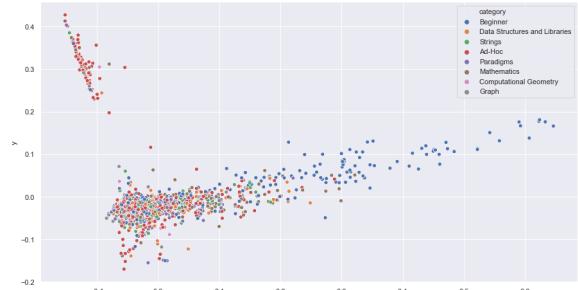
<sup>6</sup> O SVD tende a apresentar melhores resultados quando a matriz de entrada é esparsa.

Figura 17 – Visualização da distribuição espacial dos problemas por categoria.

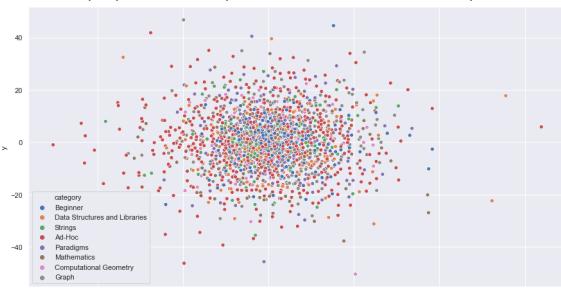
(A) PCA (termo-documento).



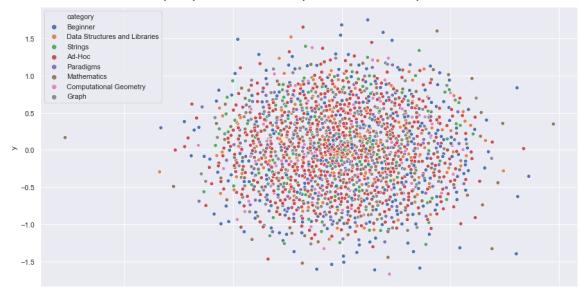
(B) PCA (TF-IDF).



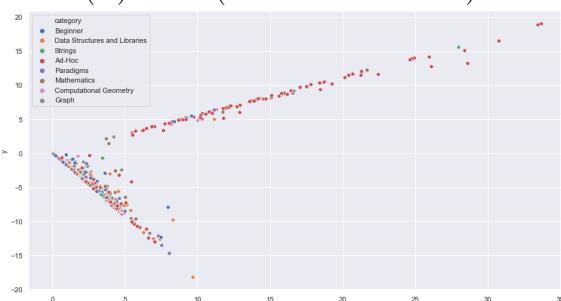
(C) MDS (termo-documento).



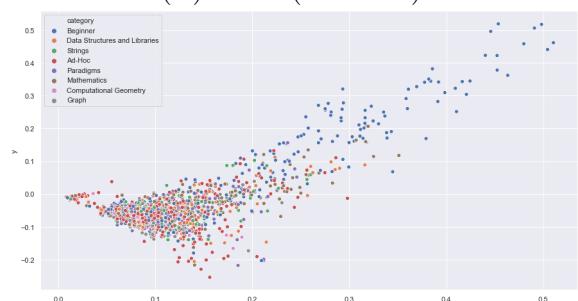
(D) MDS (TF-IDF).



(E) SVD (termo-documento).



(F) SVD (TF-IDF).



Fonte: Produzido pelo autor.

que há um certo grau de similaridade semântica entre parte dos problemas categorizados como “iniciante” (fato que acaba por ser reforçado pelas componentes dos grafos analisados na [seção 5.2](#)).

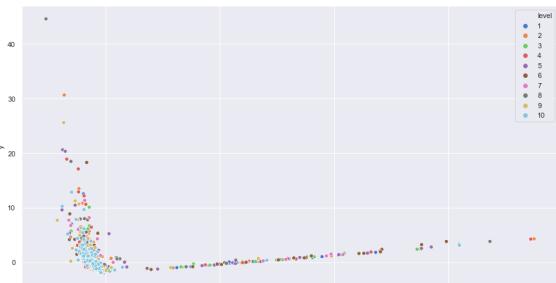
Outro fator relevante que pode ser claramente observado é a variância na distribuição de problemas categorizados como “*Ad-Hoc*”, que estão presentes em, praticamente, toda a área do gráfico que contenha outros problemas. Trata-se de um comportamento, no mínimo, interessante, uma vez que alguns desafios categorizados como “*Ad-Hoc*” apresentam pertencer à alguma das demais categorias, podendo justificar, talvez, a ausência de outros agrupamentos nos gráficos gerados, além de explicar o fato de, junto à categoria “iniciante”, ser a que mais apresenta desafios categorizados como tal.

### 5.3.2 Nível de dificuldade

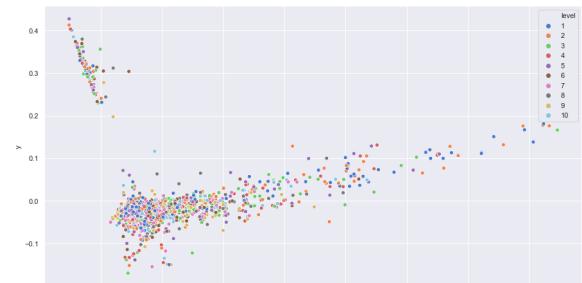
O segundo atributo a ser analisado foi o nível de dificuldade dos desafios, cuja dispersão pode ser observada na Figura 18<sup>7</sup>.

Figura 18 – Visualização da distribuição espacial dos problemas por nível de dificuldade.

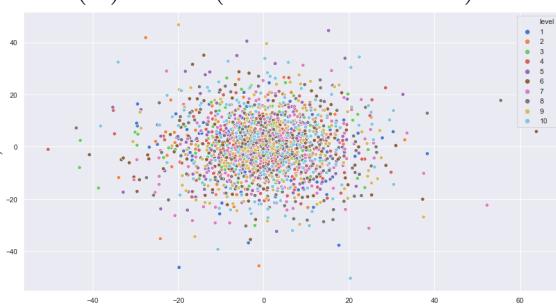
(A) PCA (termo-documento).



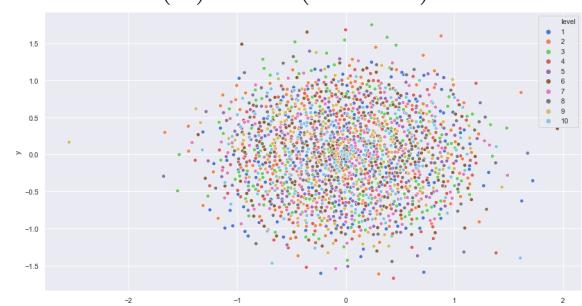
(B) PCA (TF-IDF).



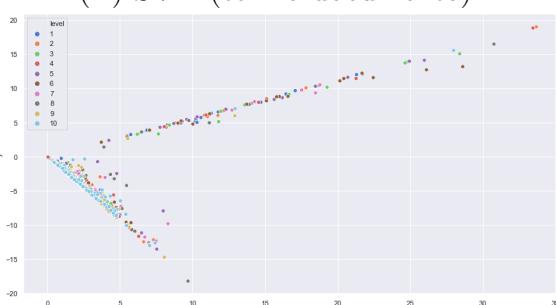
(C) MDS (termo-documento).



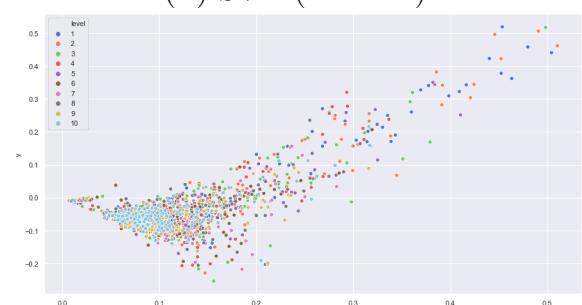
(D) MDS (TF-IDF).



(E) SVD (termo-documento).



(F) SVD (TF-IDF).



Fonte: Produzido pelo autor.

A primeira característica observável, aparentemente relevante, é que há um agrupamento de problemas de nível 10 presente nos itens A, E e F. Porém, diferentemente do esperado, estes problemas acabaram agrupados não necessariamente por possuírem vocabulário similar entre si, mas por serem diferentes de todos os outros, ou seja, cada desafio acaba apresentando uma situação distinta e, consequentemente possui um vocabulário particular. Essa característica acaba sendo reforçada, inclusive, nos grafos de co-ocorrência, onde é possível observar que desafios mais complicados tendem a apresentar menor similaridade entre si, ao analisar seus enunciados.

<sup>7</sup> Para melhor visualização dos gráficos, eles também foram adicionados ao apêndice deste trabalho (Figura 27, Figura 28, Figura 29, Figura 30, Figura 31 e Figura 32).

O primeiro agrupamento observado pode ter sido causado pela forma com que os algoritmos executam a redução de dimensionalidade. O PCA, por exemplo, busca manter dimensões em que os dados apresentem maior variância. Já o SVD busca manter as componentes com valores mais representativos. Em ambos os casos, dimensões que apresentem pouca variância nos valores dos dados podem ser, consequentemente, eliminadas. Desta forma, desafios que não apresentam vocabulário ligeiramente semelhante aos demais, como é o caso de problemas mais difíceis, podem acabar tendo suas dimensões mal representadas nas outras resultantes e, como consequência, serem espacialmente agrupados pela característica de possuírem vocabulários extremamente particulares.

Problemas mais fáceis (níveis do 1 ao 3), por outro lado, não apresentam qualquer agrupamento, uma vez que, assim como problemas classificados como “*Ad-Hoc*”, estão dispersos por toda a área plotada (vide [subseção 5.3.1](#)), podendo, inclusive, haver alguma correlação entre esses dois atributos <sup>8</sup>. Isso não significa, necessariamente, que não haja forte similaridade semântica entre alguns deles <sup>9</sup>.

### 5.3.3 Taxa de resolução

Dentre os atributos disponíveis para serem analisados, o último deles foi a taxa de resolução, cuja dispersão está representada na [Figura 19](#) <sup>10</sup>. Este atributo foi escolhido pela possibilidade de estar, de certa forma, relacionado ao nível de dificuldade já analisado na [subseção 5.3.2](#).

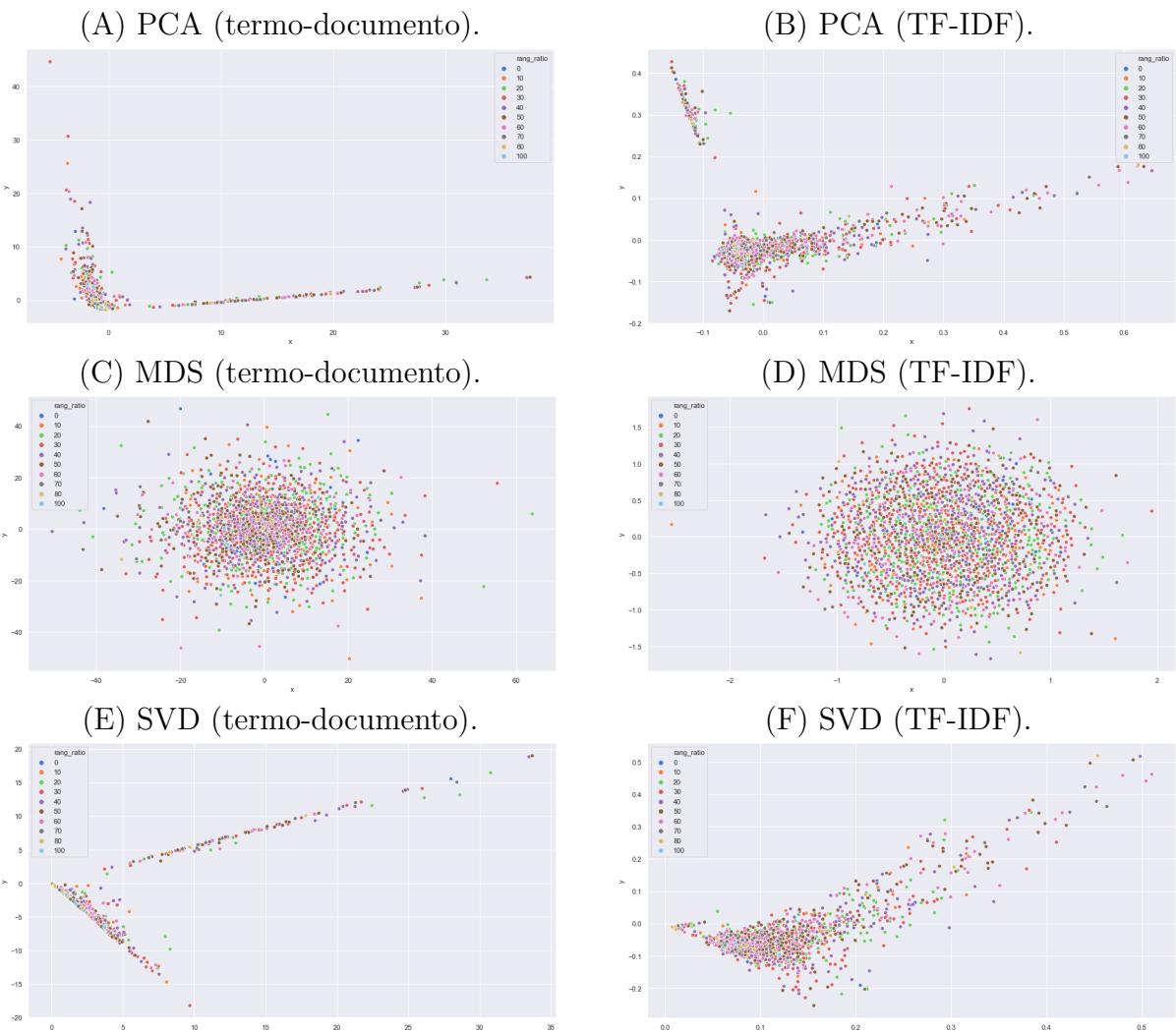
Diferentemente dos demais atributos analisados (e da expectativa de haver alguma correlação com nível), a taxa de resolução não apresenta aglomerações claras, em qualquer faixa de valor possível, embora seja notável que a maioria dos desafios apresentam taxa de resolução entre 60 e 69%, podendo ser confundida com um aglomerado. Porém, ao observar os gráficos como um todo, é possível notar que ela está dispersa por toda a área plotada.

<sup>8</sup> Análise não realizada, pois, foge ao escopo do trabalho.

<sup>9</sup> Neste caso, a melhor estrutura visual para este tipo de análise, são os grafos de co-ocorrência obtidos na [seção 5.2](#).

<sup>10</sup> Para melhor visualização dos gráficos, eles também foram adicionados ao apêndice deste trabalho ([Figura 33](#), [Figura 34](#), [Figura 35](#), [Figura 36](#), [Figura 37](#) e [Figura 38](#)).

Figura 19 – Visualização da distribuição espacial dos problemas por taxa de resolução



Fonte: Produzido pelo autor.

# Considerações finais

## Conclusões

Conforme as análises de resultados apresentadas no [Capítulo 5](#), é possível afirmar que, de fato, ao aplicar técnicas já conhecidas de PLN em um certo corpus, neste caso, o conjunto de desafios do URI, não só é possível traçar relações de similaridade semântica entre os textos por meio de uma rede de co-ocorrência, como também entre outras propriedades da entidade (problema/desafio) de forma visual, como realizado na [seção 5.3](#).

As redes de co-ocorrência geradas, representadas parcialmente pelos grafos disponíveis no apêndice ([Figura 39](#) e [Figura 40](#)), permitiram concluir que, além de o método utilizado para traçar correlações apresentarem bons resultados (conforme discutido na análise da maior componente conexa na [seção 5.2](#)), os problemas que apresentam repertório de palavras similares em suas descrições (boa similaridade semântica entre si), tendem a apresentar níveis de dificuldade mais próximos.

Ainda com relação às redes de co-ocorrência, fica evidente que as descrições dos conteúdos de entrada e de saída dos desafios também apresentam grande importância ao elaborar o modelo analítico, uma vez que mesmo que o enunciado de dois problemas distintos sejam similares, o objetivo deles pode ser totalmente diferente, de forma que esta diferença possa estar expressa somente nessas outras duas descrições. Nesse sentido, o modelo elaborado mostrou-se capaz de detectá-las, mesmo que sejam sutis.

Os gráficos de dispersão espacial obtidos a partir das matrizes termo-documento e TF-IDF, acabaram por evidenciar que é possível traçar correlações entre problemas semanticamente similares e as demais propriedades existentes, além de reforçar a ideia de que problemas mais difíceis tendem a apresentar vocabulários próprios, conforme é possível observar nos grafos de co-ocorrência.

Conforme descrito na metodologia ([subseção 4.2.1](#)), foram criadas duas matrizes distintas: uma que considera a contagem da frequência bruta dos termos e outra que normaliza a frequência e penaliza termos comuns (TF-IDF). Estas matrizes, embora tenham apresentado resultados próximos durante a geração do grafo de co-ocorrência, possuem características que podem tornar alguma delas mais relevante dependendo da aplicação final. A primeira matriz, por exemplo, apresentou melhores resultados na identificação de desafios similares para certos problemas, enquanto a dispersão espacial da TF-IDF facilitou a identificação de agrupamentos em alguns casos (característica muito útil para modelos que fazem uso de técnicas de aprendizagem de máquina).

## Limitações

Dentre as limitações das análises realizadas, a principal delas é a referente ao vocabulário, descrita na [seção 5.1](#), onde foram observadas as classes gramaticais das palavras antes de serem transformadas pelo lematizador, que foi realizada forma superficial para que o escopo proposto não fosse extrapolado. Portanto, podem ser realizados outros trabalhos cujo foco seja a análise de vocabulário.

## Trabalhos futuros

Além da possibilidade de realizar trabalhos com foco na análise de vocabulário e utilizar a matriz TF-IDF para modelos de aprendizagem de máquina, como proposto anteriormente, também é possível utilizar a estrutura obtida na [subseção 4.2.2](#) para elaborar um sistema de recomendação de enunciados relacionados, através da similaridade semântica existente entre eles ou da combinação dela com os demais atributos presentes na entidade, sistema este que pode ser útil para a elaboração de listas de exercícios, atividades, provas ou ainda para a própria plataforma *online* do URI.

Além da similaridade entre documentos, também seria interessante realizar um aprofundamento quanto à análise dos significados que cada palavra pode ter em problemas de maratona de programação, uma vez que podem variar de acordo com o problema, categoria (como no caso da palavra ‘banco’ na categoria ‘banco de dados’ e nos demais) ou até mesmo com o nível de dificuldade.

## Referências

- ACHANANUPARP, P. et al. Utilizing sentence similarity and question type similarity to response to similar questions in knowledge-sharing community. In: *Proceedings of QAWeb 2008 Workshop, Beijing, China*. [S.l.: s.n.], 2008. v. 214. Citado na página 14.
- BARBERÁN, A. et al. Using network analysis to explore co-occurrence patterns in soil microbial communities. *The ISME Journal*, v. 6, n. 2, p. 343–351, Feb 2012. ISSN 1751-7370. Disponível em: <<https://doi.org/10.1038/ismej.2011.119>>. Citado na página 32.
- BASTIAN, M. et al. Gephi: an open source software for exploring and manipulating networks. *Icwsrm*, San Jose, California, v. 8, n. 2009, p. 361–362, 2009. Citado na página 49.
- BISHOP, D. *Text Analytics – Document Term Matrix*. 2017. Website online. Disponível em: <<https://www.darrinbishop.com/blog/2017/10/text-analytics-document-term-matrix/>>. Acesso em: 20 nov. 2019. Citado na página 23.
- BORG, P. J. F. G. I. *Modern Multidimensional Scaling: Theory and Applications*. 2nd. ed. [S.l.]: Springer, 2005. (Springer Series in Statistics). ISBN 9780387251509,0387251502. Citado na página 30.
- BROWNLEE, J. *A Gentle Introduction to the Bag-of-Words Model*. 2017. Website online. Disponível em: <<https://machinelearningmastery.com/gentle-introduction-bag-words-model/>>. Acesso em: 20 nov. 2019. Citado na página 23.
- CHONG, M.; SPECIA, L.; MITKOV, R. Using natural language processing for automatic detection of plagiarism. In: *Proceedings of the 4th International Plagiarism Conference (IPC-2010)*. [S.l.: s.n.], 2010. Citado na página 12.
- CHRUPAŁA, G. Simple data-driven context-sensitive lemmatization. 2006. Citado na página 19.
- ELLSON, J. et al. Graphviz and dynagraph – static and dynamic graph drawing tools. In: *GRAPH DRAWING SOFTWARE*. [S.l.]: Springer-Verlag, 2003. p. 127–148. Citado na página 49.
- FACER, C. *Text Analysis: Hooking up Your Term Document Matrix to Custom R Code*. 2017. Website online. Disponível em: <<https://www.displayr.com/text-analysis-hooking-up-your-term-document-matrix-to-custom-r-code/>>. Acesso em: 21 nov. 2019. Citado na página 23.
- FELLBAUM, C. *WordNet: An Electronic Lexical Database*. [S.l.]: Bradford Books, 1998. Citado na página 41.
- FOUNDATION, A. S. *TFIDFSimilarity (Lucene 5.1.0 API)*. 2017. Website online. Disponível em: <[https://lucene.apache.org/core/5\\_1\\_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html](https://lucene.apache.org/core/5_1_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html)>. Acesso em: 12 jul. 2019. Citado na página 26.

- GODAYAL, D.; MALHOTRA, S. *An introduction to part-of-speech tagging and the Hidden Markov Model*. 2018. Website online. Disponível em: <<https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/>>. Acesso em: 20 nov. 2019. Citado na página 23.
- GUPTA, D.; VANI, K.; SINGH, C. K. Using natural language processing techniques and fuzzy-semantic similarity for automatic external plagiarism detection. In: IEEE. *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.], 2014. p. 2694–2699. Citado na página 12.
- HARMAN, D.; CANDELA, G. Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *Journal of the American Society for Information Science*, v. 41, n. 8, p. 581–589, 1990. Disponível em: <<https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-4571%28199012%2941%3A8%3C581%3A%3AAID-ASI4%3E3.0.CO%3B2-U>>. Citado na página 21.
- HARROWER, M.; BREWER, C. A. Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, Taylor & Francis, v. 40, n. 1, p. 27–37, 2003. Disponível em: <<https://www.tandfonline.com/doi/abs/10.1179/000870403235002042>>. Citado na página 49.
- HEDLEY, J. *jsoup Java HTML Parser, with best of DOM, CSS, and jquery*. 2019. Disponível em: <<https://jsoup.org/>>. Acesso em: 25 nov. 2019. Citado na página 33.
- HEIDENREICH, H. *Stemming? Lemmatization? What?* 2018. Disponível em: <<https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8>>. Acesso em: 11 nov. 2019. Citado 2 vezes nas páginas 21 e 22.
- JAMES, G. et al. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 112. Citado na página 29.
- JAYAKODI, K.; BANDARA, M.; MEEDENIYA, D. An automatic classifier for exam questions with wordnet and cosine similarity. In: IEEE. *2016 Moratuwa Engineering Research Conference (MERCon)*. [S.l.], 2016. p. 12–17. Citado na página 17.
- JURAFSKY, D.; MARTIN, J. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. [S.l.: s.n.], 2008. v. 2. Citado na página 19.
- KUBEK, M.; UNGER, H.; DUSIK, J. Correlating words - approaches and applications. In: . [S.l.: s.n.], 2015. Citado na página 32.
- Lexical Computing. *POS tags*. 2019. Website online. Disponível em: <<https://www.sketchengine.eu/pos-tags/>>. Acesso em: 19 nov. 2019. Citado na página 23.
- LOVINS, J. B. Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, v. 11, n. 1-2, p. 22–31, 1968. Citado na página 21.
- MANNING, C. D.; RAGHAVAN, P.; SCHÜTZ, H. The term vocabulary and postings lists. In: \_\_\_\_\_. *Introduction to Information Retrieval*. [S.l.]: Cambridge University Press, 2008. Citado 5 vezes nas páginas 19, 25, 26, 30 e 31.

- MCKINNEY, W. *pandas.DataFrame - pandas 0.25.3 documentation*. 2019. Disponível em: <<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>>. Acesso em: 26 nov. 2019. Citado na página 37.
- MCKINNEY, W. et al. Data structures for statistical computing in python. In: AUSTIN, TX. *Proceedings of the 9th Python in Science Conference*. [S.l.], 2010. v. 445, p. 51–56. Citado na página 37.
- MENA-CHALCO, J. P. *Normalização de texto: Palavras e stopwords*. 2019. Disponível em: <<http://professor.ufabc.edu.br/~jesus.mena/courses/pln-2q-2019/PLN-aula03.pdf>>. Acesso em: 23 ago. 2019. Citado na página 19.
- MEURERS, D. et al. Integrating parallel analysis modules to evaluate the meaning of answers to reading comprehension questions. *International Journal of Continuing Engineering Education and Life-Long Learning*, Inderscience Publishers Ltd., PO Box 735 Olney Bucks MK 46 5 WB United Kingdom, v. 21, n. 4, p. 355–369, 2011. Citado na página 12.
- MOHLER, M.; BUNESCU, R.; MIHALCEA, R. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (HLT '11), p. 752–762. ISBN 978-1-932432-87-9. Disponível em: <<http://dl.acm.org/citation.cfm?id=2002472.2002568>>. Citado na página 17.
- MOREIRA, V. P. *RSLP Stemmer (Removedor de Sufixos da Lingua Portuguesa)*. 2010. Disponível em: <<http://www.inf.ufrgs.br/~viviane/rslp/index.htm>>. Acesso em: 16 nov. 2019. Citado na página 22.
- NLTK Project. *nltk.stem package*. 2019. Disponível em: <<https://www.nltk.org/api/nltk.stem.html#module-nltk.stem.wordnet>>. Acesso em: 27 nov. 2019. Citado na página 40.
- NLTK Project. *nltk.tag package*. 2019. Disponível em: <<https://www.nltk.org/api/nltk.tag.html#module-nltk.tag>>. Acesso em: 26 nov. 2019. Citado na página 40.
- NLTK Project. *nltk.tokenize package*. 2019. Disponível em: <<https://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize>>. Acesso em: 26 nov. 2019. Citado na página 38.
- OLIVEIRA, G. F. de. *XXIV Maratona de Programação*. 2019. Website online. Disponível em: <<http://maratona.ime.usp.br/sobre19.html>>. Acesso em: 06 nov. 2019. Citado na página 12.
- PADMANABHAN, A. *Lemmatization*. 2019. Website online. Disponível em: <<https://devopedia.org/lemmatization>>. Acesso em: 28 fev. 2020. Citado na página 19.
- PAUL, M.; JAMAL, S. An improved srl based plagiarism detection technique using sentence ranking. *Procedia Computer Science*, Elsevier, v. 46, p. 223–230, 2015. Citado na página 12.
- PEDREGOSA, F. et al. *Decomposing signals in components (matrix factorization problems)*: Truncated singular value decomposition and latent semantic analysis. 2020. Disponível em: <<https://scikit-learn.org/stable/modules/decomposition.html#lsa>>. Citado na página 46.

- PEDREGOSA, F. et al. *Manifold learning: Multi-dimensional Scaling (MDS)*. 2020. Disponível em: <<https://scikit-learn.org/stable/modules/manifold.html#multidimensional-scaling>>. Citado 2 vezes nas páginas 30 e 45.
- PORTER, M. F. An algorithm for suffix stripping. *Program*, MCB UP Ltd, v. 14, n. 3, p. 130–137, 1980. Citado 2 vezes nas páginas 21 e 22.
- PORTER, M. F. *Porter Stemming Algorithm*. 2006. Website online. Disponível em: <<https://tartarus.org/martin/PorterStemmer/>>. Acesso em: 27 fev. 2020. Citado na página 21.
- PRABHAKARAN, S. *Cosine Similarity – Understanding the math and how it works (with python codes)*. 2019. Website online. Disponível em: <<https://www.machinelearningplus.com/nlp/cosine-similarity/>>. Acesso em: 19 nov. 2019. Citado na página 28.
- RACHIELE, G. *Tokenization and Parts of Speech(POS) Tagging in Python's NLTK library*. 2018. Disponível em: <<https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk-library-2d30f70af13b>>. Acesso em: 20 nov. 2019. Citado na página 23.
- Refsnes Data. *Python String isalpha() Method*. 2019. Disponível em: <[https://www.w3schools.com/python/ref\\_string\\_isalpha.asp](https://www.w3schools.com/python/ref_string_isalpha.asp)>. Acesso em: 26 nov. 2019. Citado na página 39.
- SAKTI, E.; FAUZI, M. Comparative analysis of string similarity and corpus-based similarity for automatic essay scoring system on e-learning gamification. In: . [S.l.: s.n.], 2016. Citado 2 vezes nas páginas 15 e 16.
- SANTORINI, B. *Part-of-speech tagging guidelines for the Penn Treebank Project*. [S.l.]: University of Pennsylvania, School of Engineering and Applied Science . . . , 1990. Citado na página 40.
- SARKAR, S. et al. Nlp algorithm based question and answering system. In: *Seventh International Conference on Computational Intelligence, Modeling and Simulation*. [S.l.: s.n.], 2015. Citado na página 15.
- SIEG, A. *Text Similarities : Estimate the degree of similarity between two texts*. 2018. Website online. Disponível em: <<https://medium.com/@adriensieg/text-similarities-da019229c894>>. Acesso em: 20 nov. 2019. Citado na página 28.
- Stanford NLP Group. *Stanford Log-linear Part-Of-Speech Tagger*. 2019. Website online. Disponível em: <<https://nlp.stanford.edu/software/tagger.shtml>>. Acesso em: 19 nov. 2019. Citado na página 23.
- TONIN, N.; URI Online Judge. *1183 - Above the Main Diagonal*. 2012. Disponível em: <<https://www.urionlinejudge.com.br/judge/en/problems/view/1183>>. Acesso em: 09 dez. 2019. Citado na página 50.
- TONIN, N.; URI Online Judge. *1184 - Above the Main Diagonal*. 2012. Disponível em: <<https://www.urionlinejudge.com.br/judge/en/problems/view/1184>>. Acesso em: 09 dez. 2019. Citado na página 50.

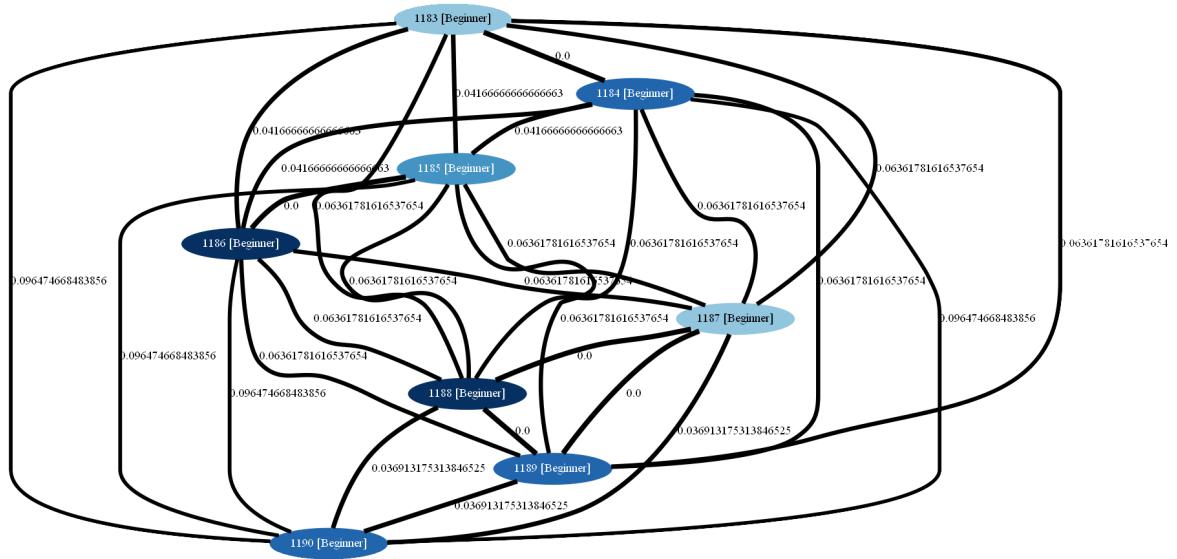
TONIN, N.; URI Online Judge. *Odd Numbers*. 2012. Disponível em: <<https://www.urionlinejudge.com.br/judge/en/problems/view/1067>>. Acesso em: 04 dez. 2020. Citado na página 52.

TONIN, N.; URI Online Judge. *Six Odd Numbers*. 2012. Disponível em: <<https://www.urionlinejudge.com.br/judge/en/problems/view/1070>>. Acesso em: 04 dez. 2020. Citado na página 52.

# Apêndices

# APÊNDICE A – Componentes principais de grafos de co-ocorrência

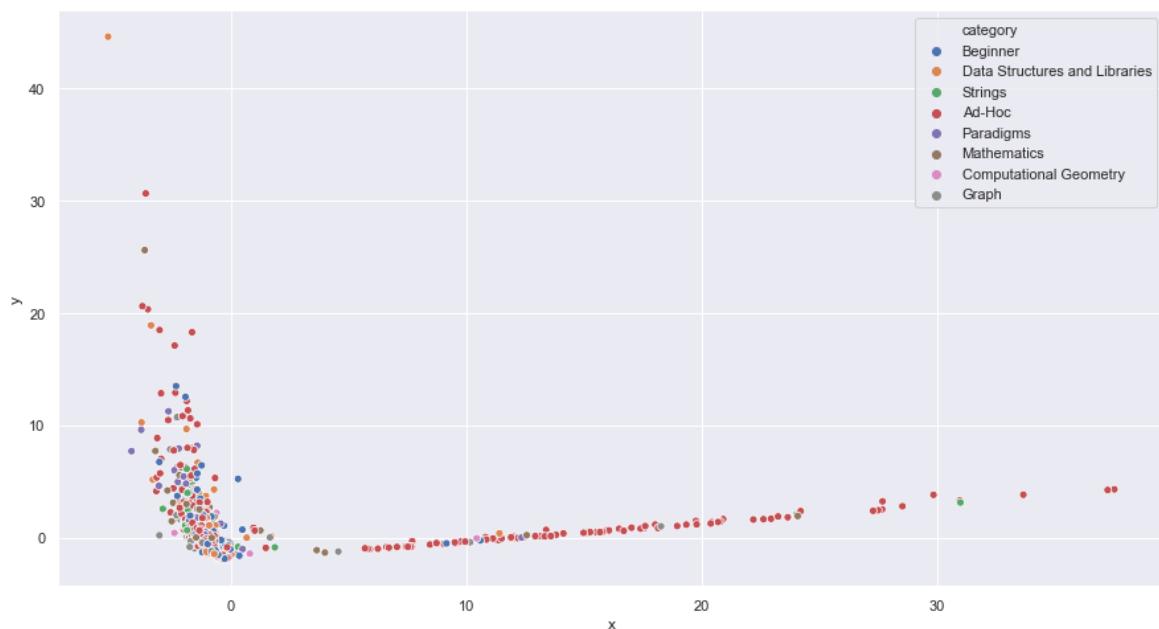
Figura 20 – Maior componente conexa do grafo de co-ocorrência (utilizando a matriz TF-IDF e considerando apenas a descrição do problema).



Fonte: Produzido pelo autor.

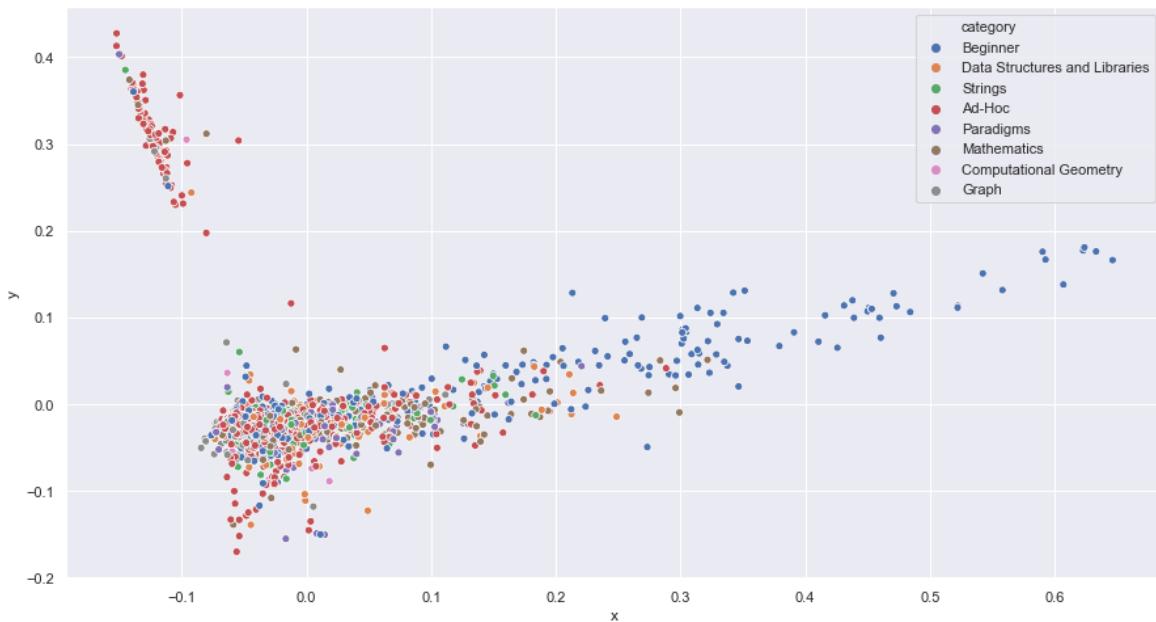
## APÊNDICE B – Gráficos de dispersão dos problemas analisados (distribuição espacial)

Figura 21 – Distribuição espacial dos problemas por categoria (PCA em matriz termo-documento).



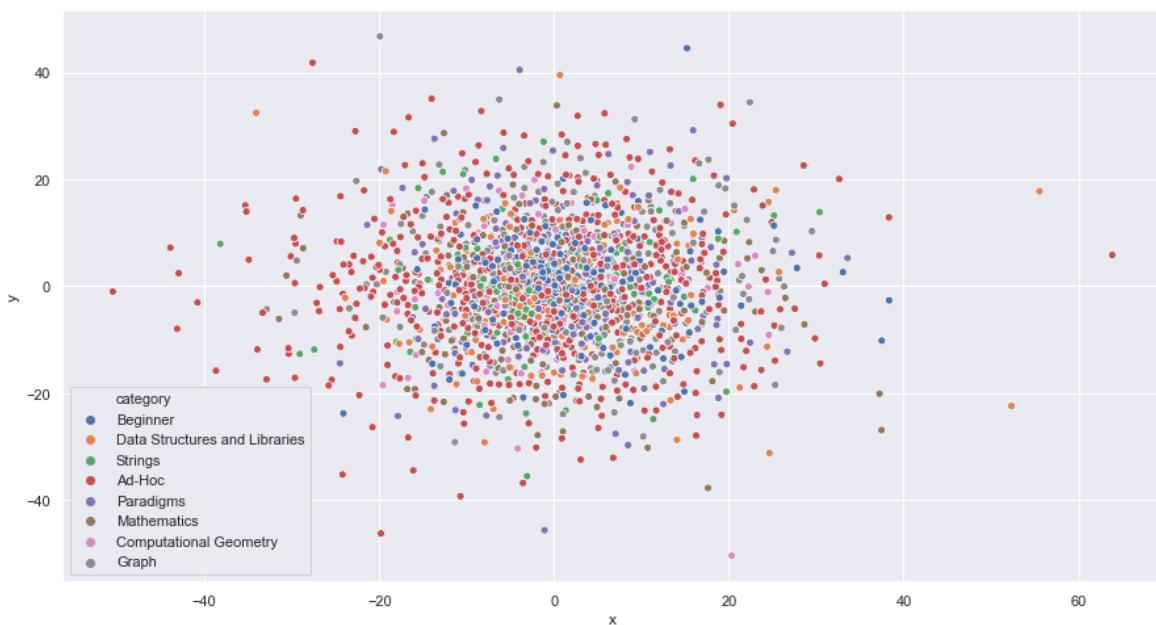
Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do PCA (redução para 2 dimensões). Item A da Figura 17. Fonte: Produzido pelo autor.

Figura 22 – Distribuição espacial dos problemas por categoria (PCA em matriz TF-IDF).



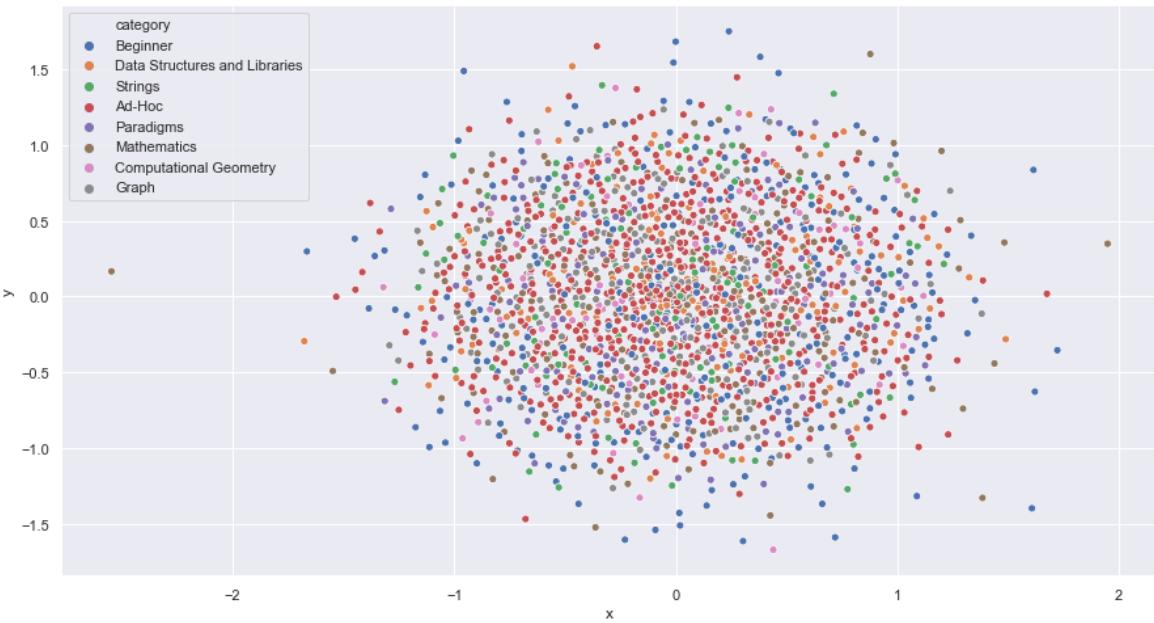
Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do PCA (redução para 2 dimensões). Item B da [Figura 17](#). Fonte: Produzido pelo autor.

Figura 23 – Distribuição espacial dos problemas por categoria (MDS em matriz termo-documento).



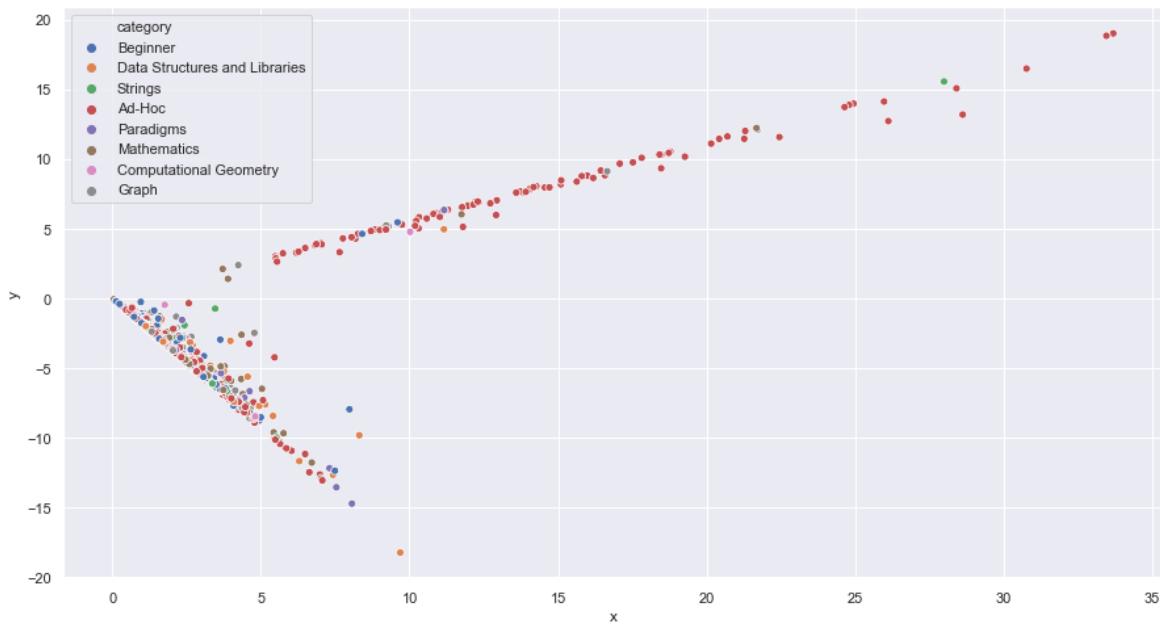
Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do MDS (redução para 2 dimensões). Item C da [Figura 17](#). Fonte: Produzido pelo autor.

Figura 24 – Distribuição espacial dos problemas por categoria (MDS em matriz TF-IDF).



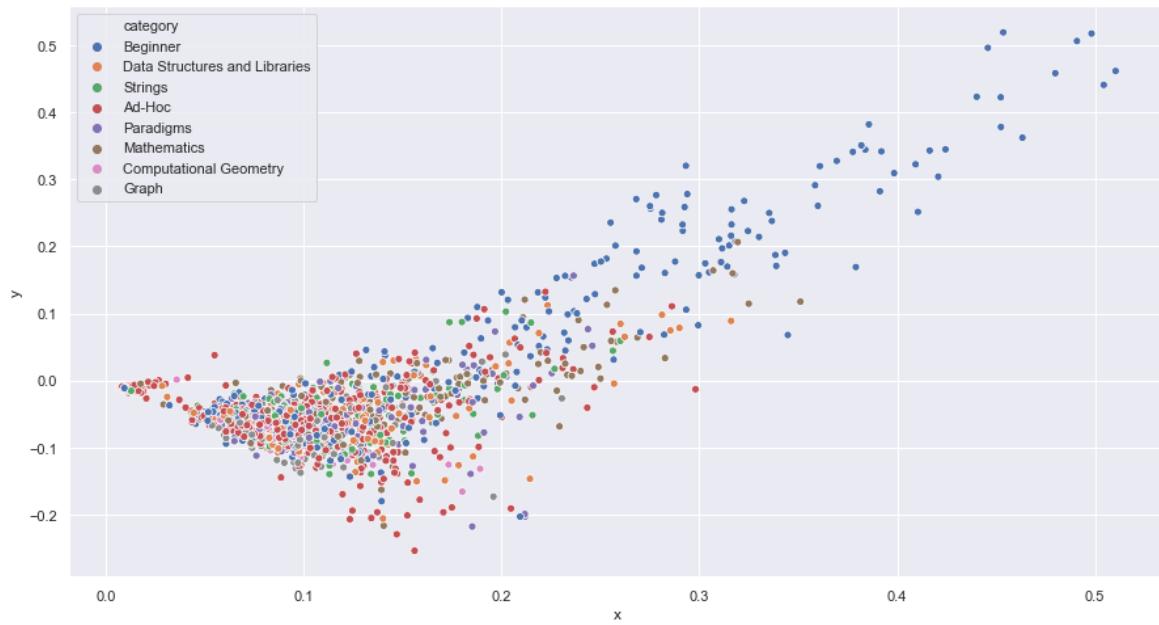
Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do MDS (redução para 2 dimensões). Item D da [Figura 17](#). Fonte: Produzido pelo autor.

Figura 25 – Distribuição espacial dos problemas por categoria (SVD em matriz termo-documento).



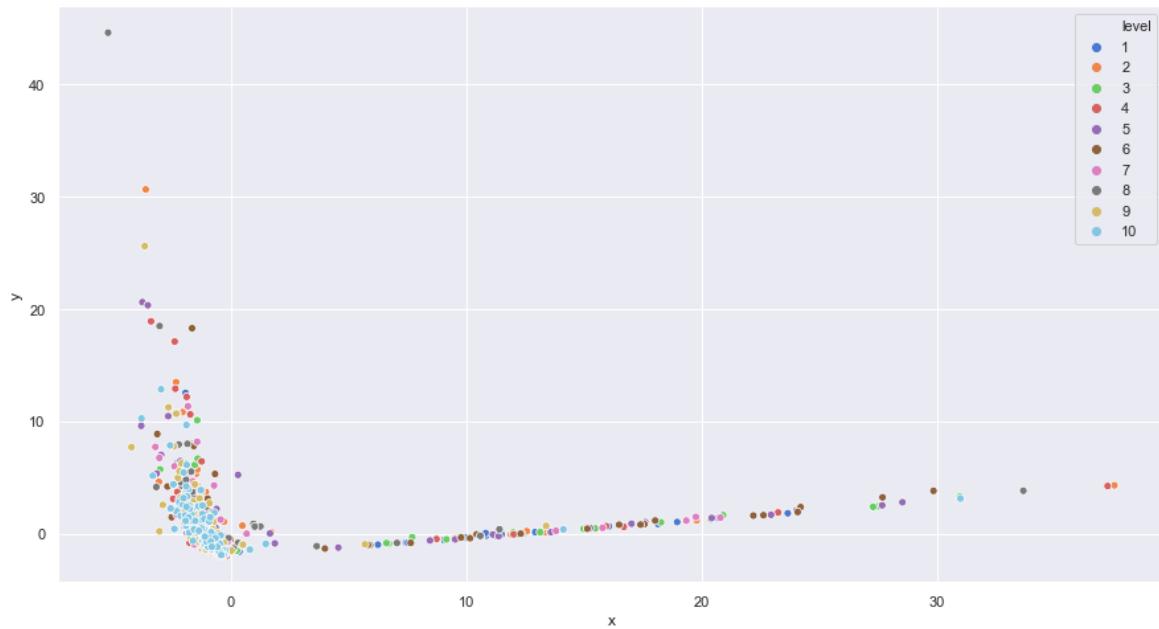
Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do SVD (redução para 2 dimensões). Item E da [Figura 17](#). Fonte: Produzido pelo autor.

Figura 26 – Distribuição espacial dos problemas por categoria (SVD em matriz TF-IDF).



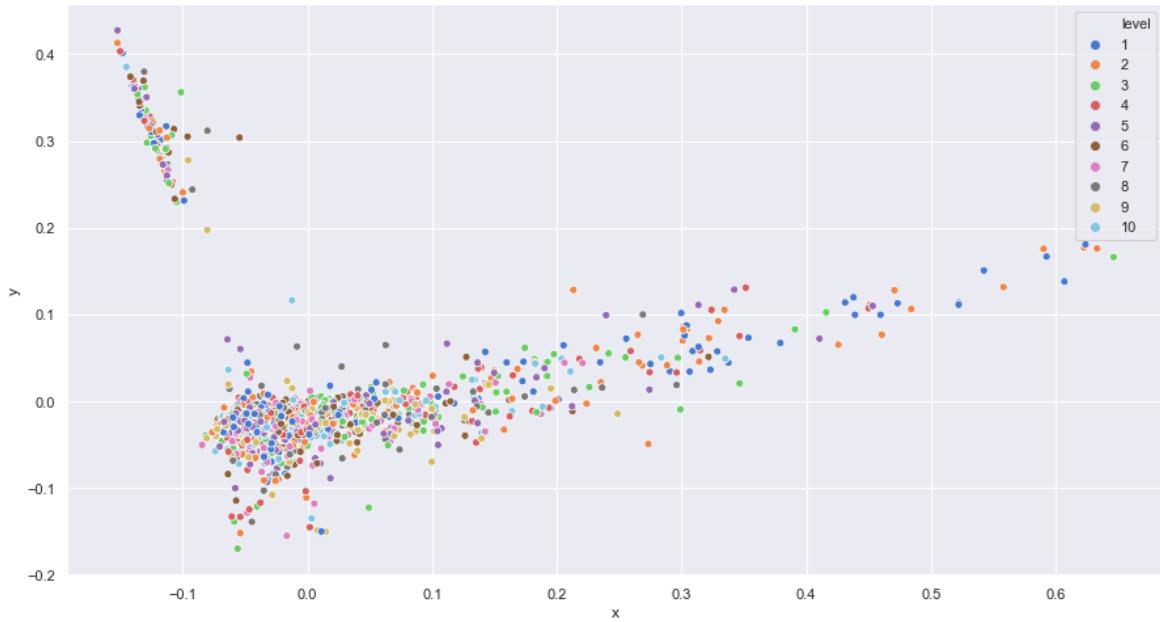
Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do SVD (redução para 2 dimensões). Item F da [Figura 17](#). Fonte: Produzido pelo autor.

Figura 27 – Distribuição espacial dos problemas por nível de dificuldade (PCA em matriz termo-documento).



Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do PCA (redução para 2 dimensões). Item A da [Figura 18](#). Fonte: Produzido pelo autor.

Figura 28 – Distribuição espacial dos problemas por nível de dificuldade (PCA em matriz TF-IDF).



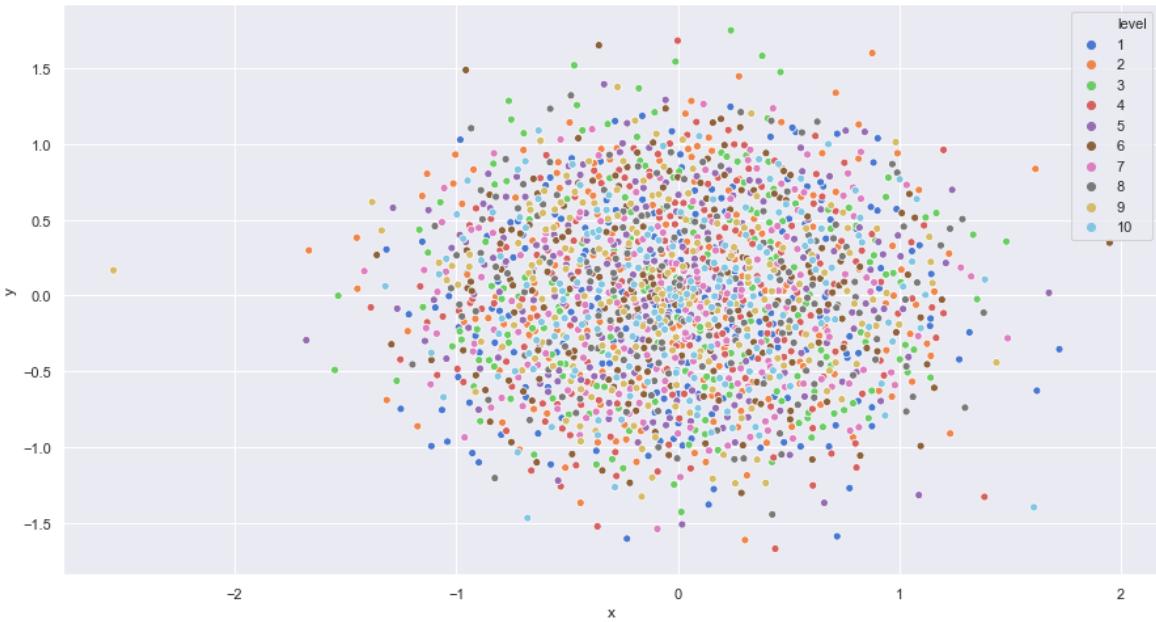
Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do PCA (redução para 2 dimensões). Item B da [Figura 18](#). Fonte: Produzido pelo autor.

Figura 29 – Distribuição espacial dos problemas por nível de dificuldade (MDS em matriz termo-documento).



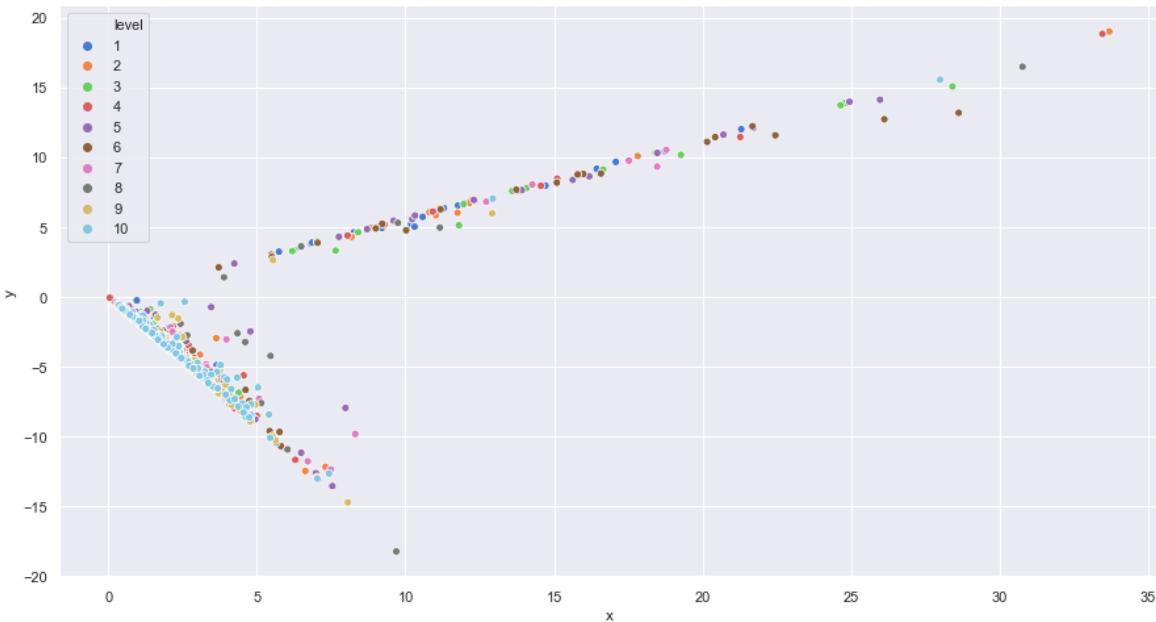
Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do MDS (redução para 2 dimensões). Item C da [Figura 18](#). Fonte: Produzido pelo autor.

Figura 30 – Distribuição espacial dos problemas por nível de dificuldade (MDS em matriz TF-IDF).



Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do MDS (redução para 2 dimensões). Item D da [Figura 18](#). Fonte: Produzido pelo autor.

Figura 31 – Distribuição espacial dos problemas por nível de dificuldade (SVD em matriz termo-documento).



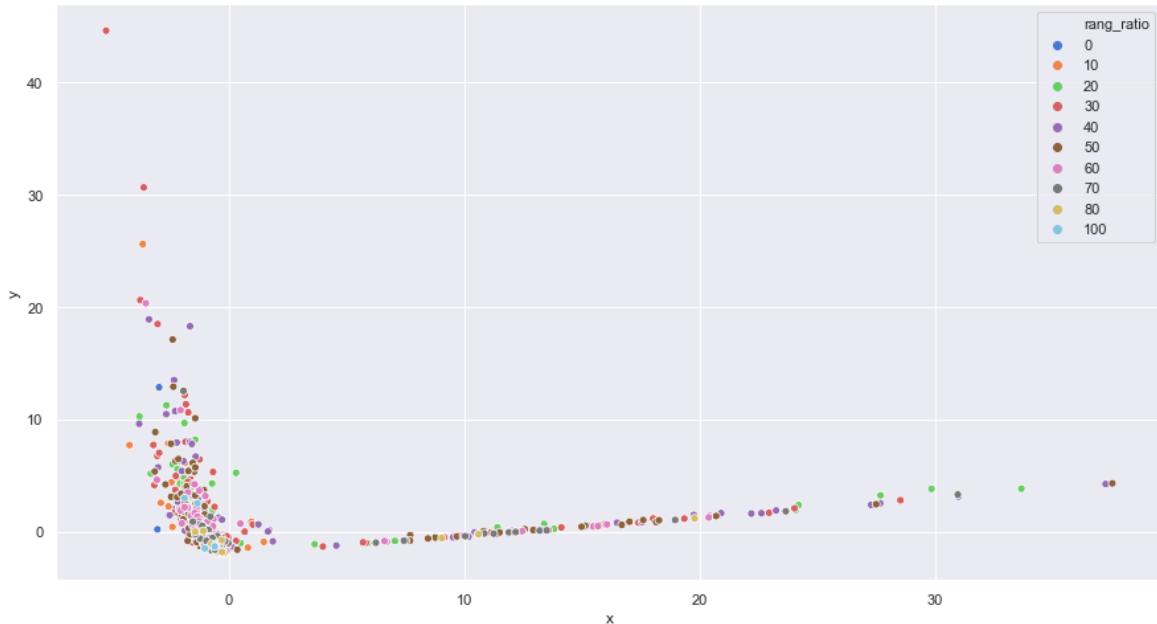
Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do SVD (redução para 2 dimensões). Item E da [Figura 18](#). Fonte: Produzido pelo autor.

Figura 32 – Distribuição espacial dos problemas por nível de dificuldade (SVD em matriz TF-IDF).



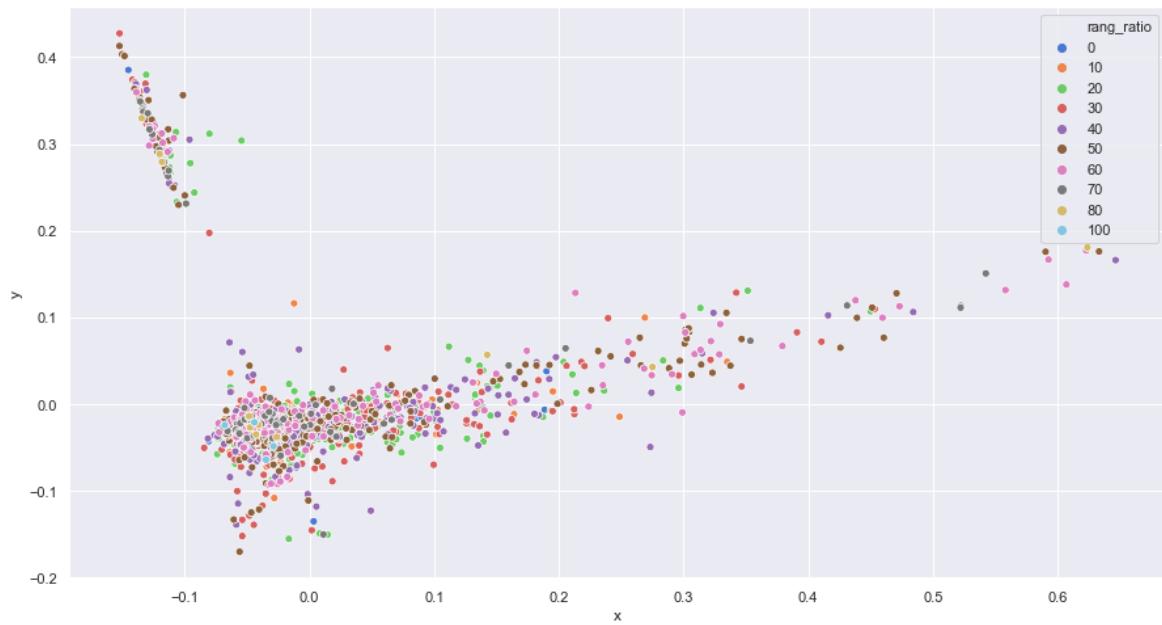
Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do SVD (redução para 2 dimensões). Item F da [Figura 18](#). Fonte: Produzido pelo autor.

Figura 33 – Distribuição espacial dos problemas por taxa de resolução (PCA em matriz termo-documento).



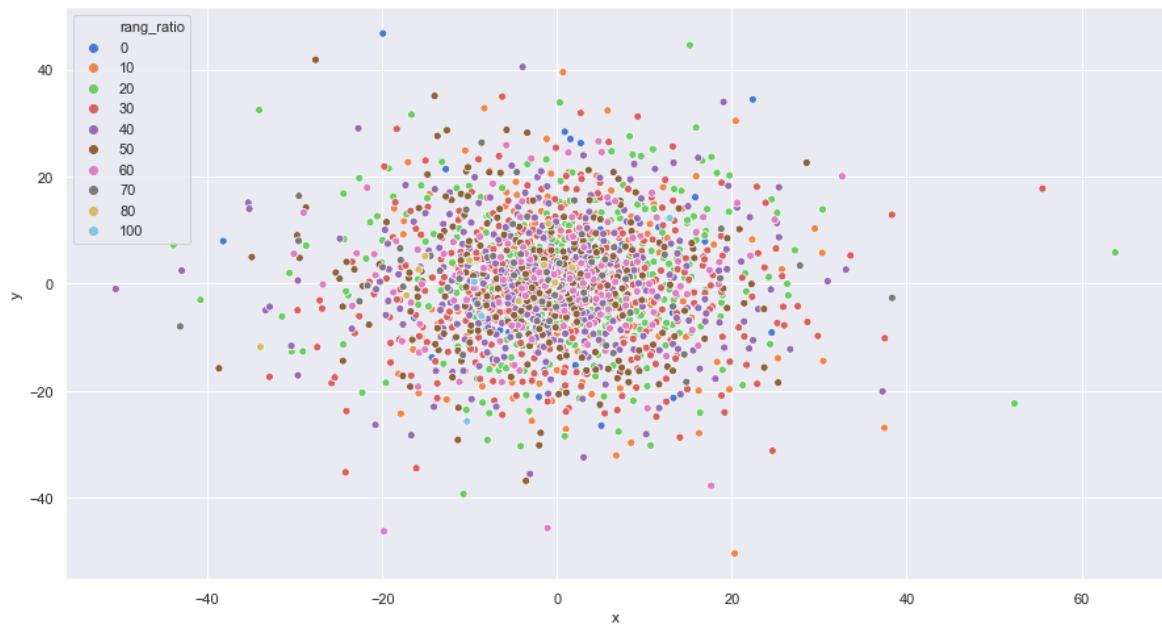
Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do PCA (redução para 2 dimensões). Item A da [Figura 19](#). Fonte: Produzido pelo autor.

Figura 34 – Distribuição espacial dos problemas por taxa de resolução (PCA em matriz TF-IDF).



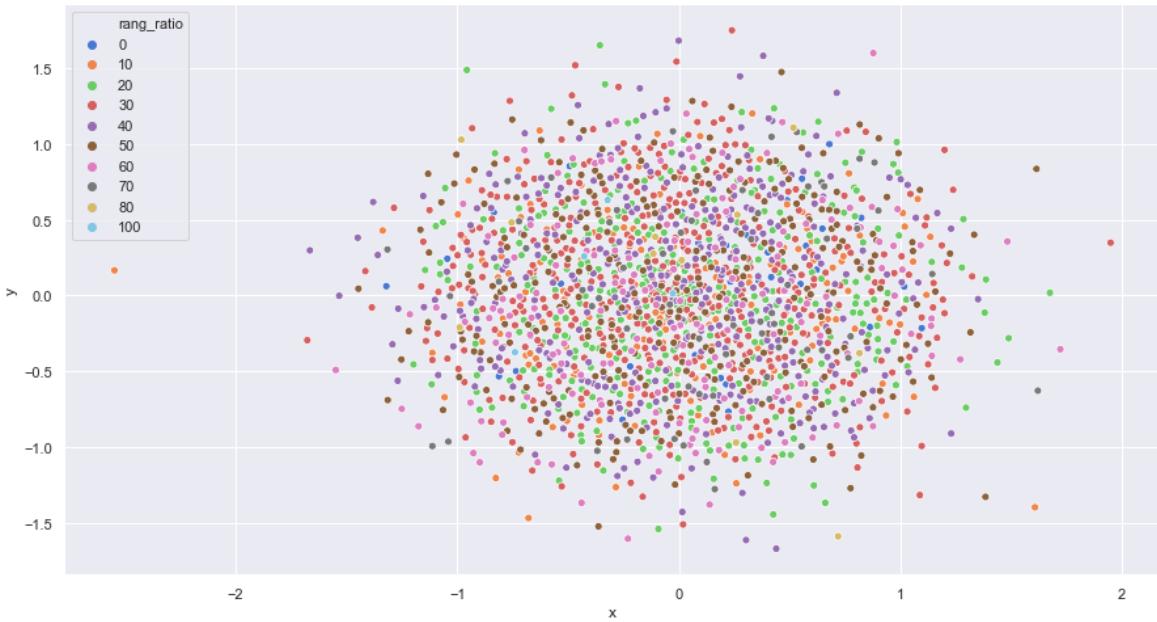
Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do PCA (redução para 2 dimensões). Item B da [Figura 19](#). Fonte: Produzido pelo autor.

Figura 35 – Distribuição espacial dos problemas por taxa de resolução (MDS em matriz termo-documento).



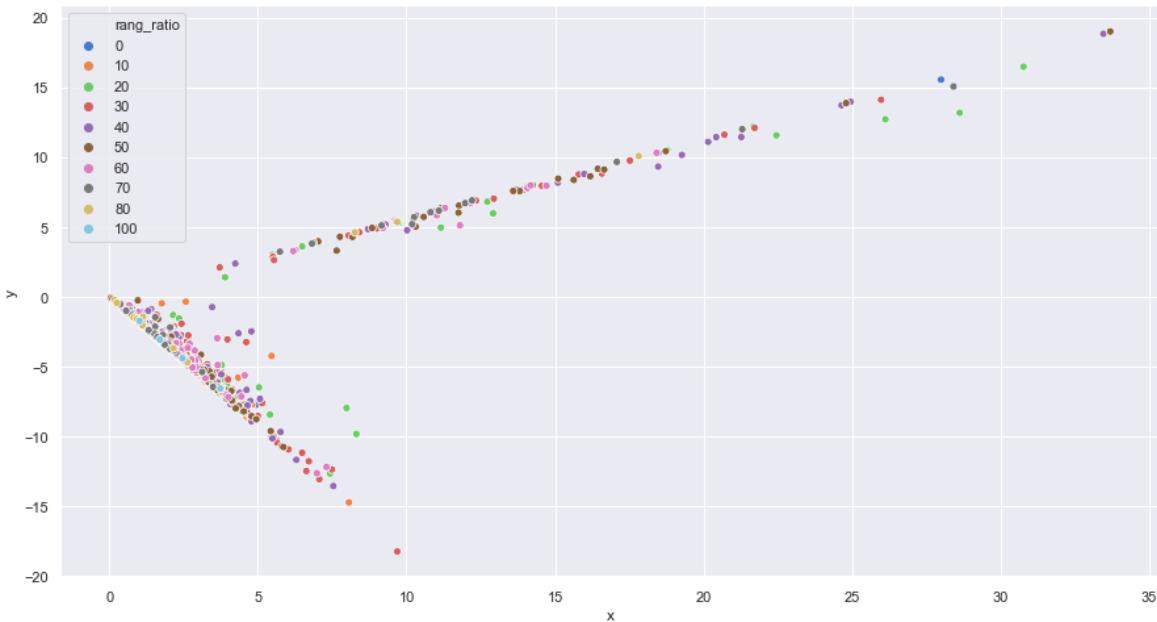
Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do MDS (redução para 2 dimensões). Item C da [Figura 19](#). Fonte: Produzido pelo autor.

Figura 36 – Distribuição espacial dos problemas por taxa de resolução (MDS em matriz TF-IDF).



Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do MDS (redução para 2 dimensões). Item D da [Figura 19](#). Fonte: Produzido pelo autor.

Figura 37 – Distribuição espacial dos problemas por taxa de resolução (SVD em matriz termo-documento).



Distribuição espacial dos problemas com base na matriz termo-documento após aplicação do SVD (redução para 2 dimensões). Item E da [Figura 19](#). Fonte: Produzido pelo autor.

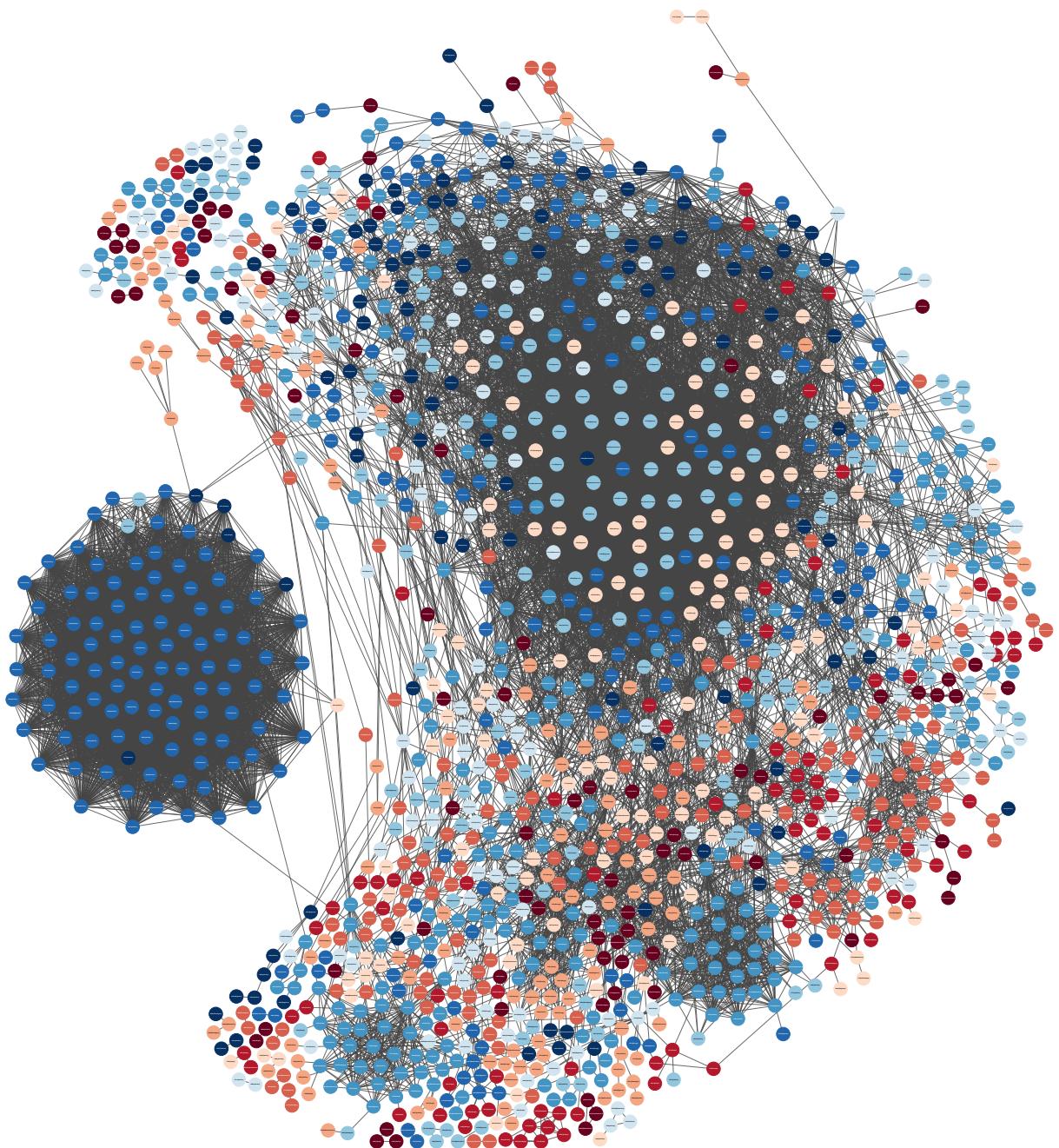
Figura 38 – Distribuição espacial dos problemas por taxa de resolução (SVD em matriz TF-IDF).



Distribuição espacial dos problemas com base na matriz TF-IDF após aplicação do SVD (redução para 2 dimensões). Item F da [Figura 19](#). Fonte: Produzido pelo autor.

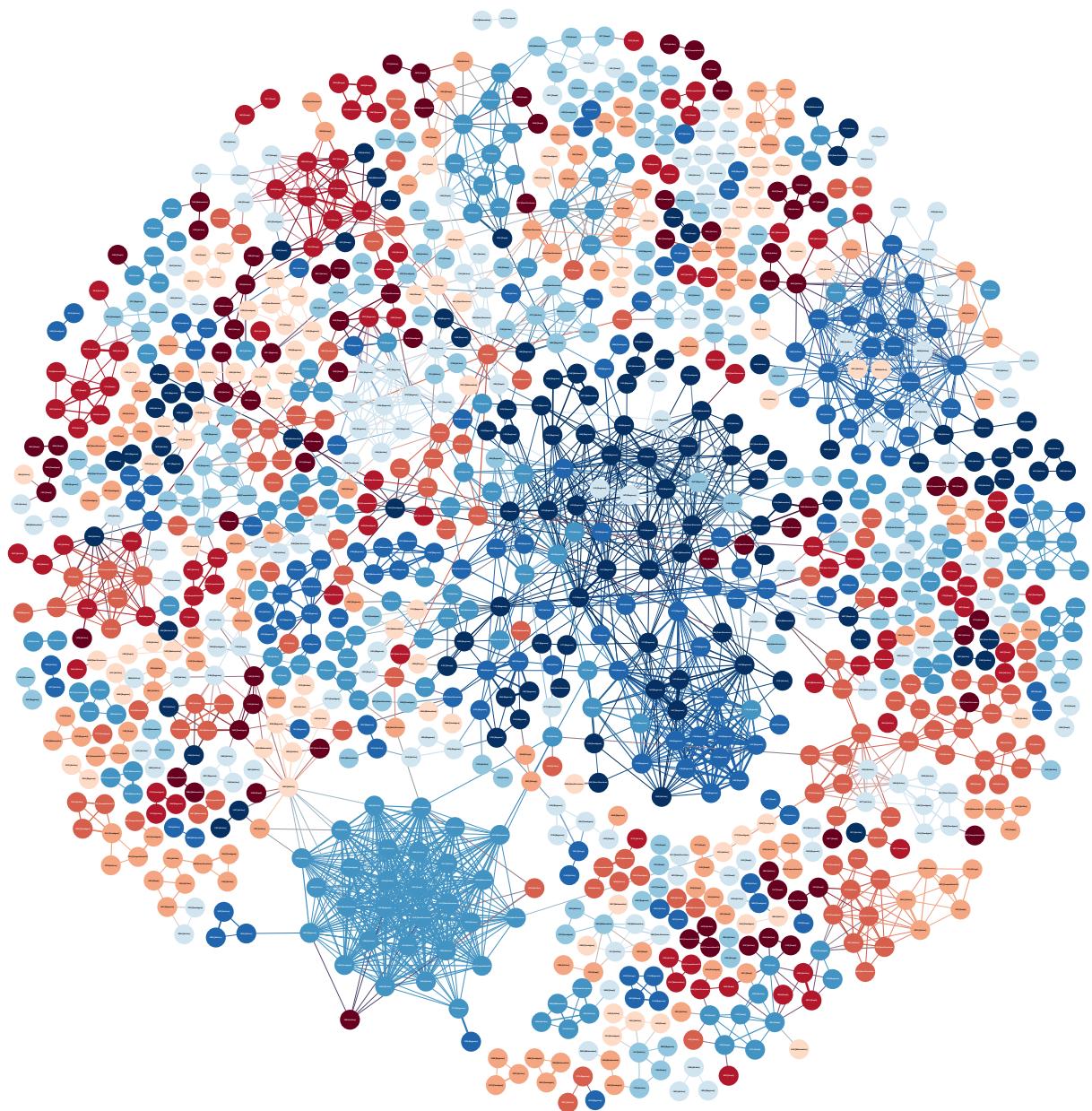
## APÊNDICE C – Grafos de co-ocorrência

Figura 39 – Grafo de co-ocorrência (matriz termo-documento).



Grafo gerado a partir do cálculo da distância do cosseno utilizando a matriz termo-documento, considerando apenas a descrição do problema. A fim de prover uma melhor visualização gráfica, este grafo conta apenas com desafios que apresentem ao menos 70% de similaridade. Fonte: Produzido pelo autor.

Figura 40 – Grafo de co-ocorrência (matriz TF-IDF).



Grafo gerado a partir do cálculo da distância do cosseno utilizando a matriz TF-IDF, considerando apenas a descrição do problema. A fim de prover uma melhor visualização gráfica, este grafo conta apenas com desafios que apresentem ao menos 70% de similaridade. Fonte: Produzido pelo autor.

# Índice

Atributos dos problemas disponíveis no *DataFrame*, 38

Classes gramaticais do *Penn Treebank*, 24

Estrutura de elemento HTML de dados de descrição, entrada e saída de problemas do URI, 33

Estrutura de elemento HTML de estatísticas de problemas do URI, 34

Formato do problema extraído em JSON, 35

Histogramas para cada categoria dos problemas, gerados com base nas classificações gramaticais das palavras presentes em suas descrições, 48

Visualização da distribuição espacial dos problemas por categoria, 54

Visualização da distribuição espacial dos problemas por nível de dificuldade, 55

Visualização da distribuição espacial dos problemas por taxa de resolução., 57