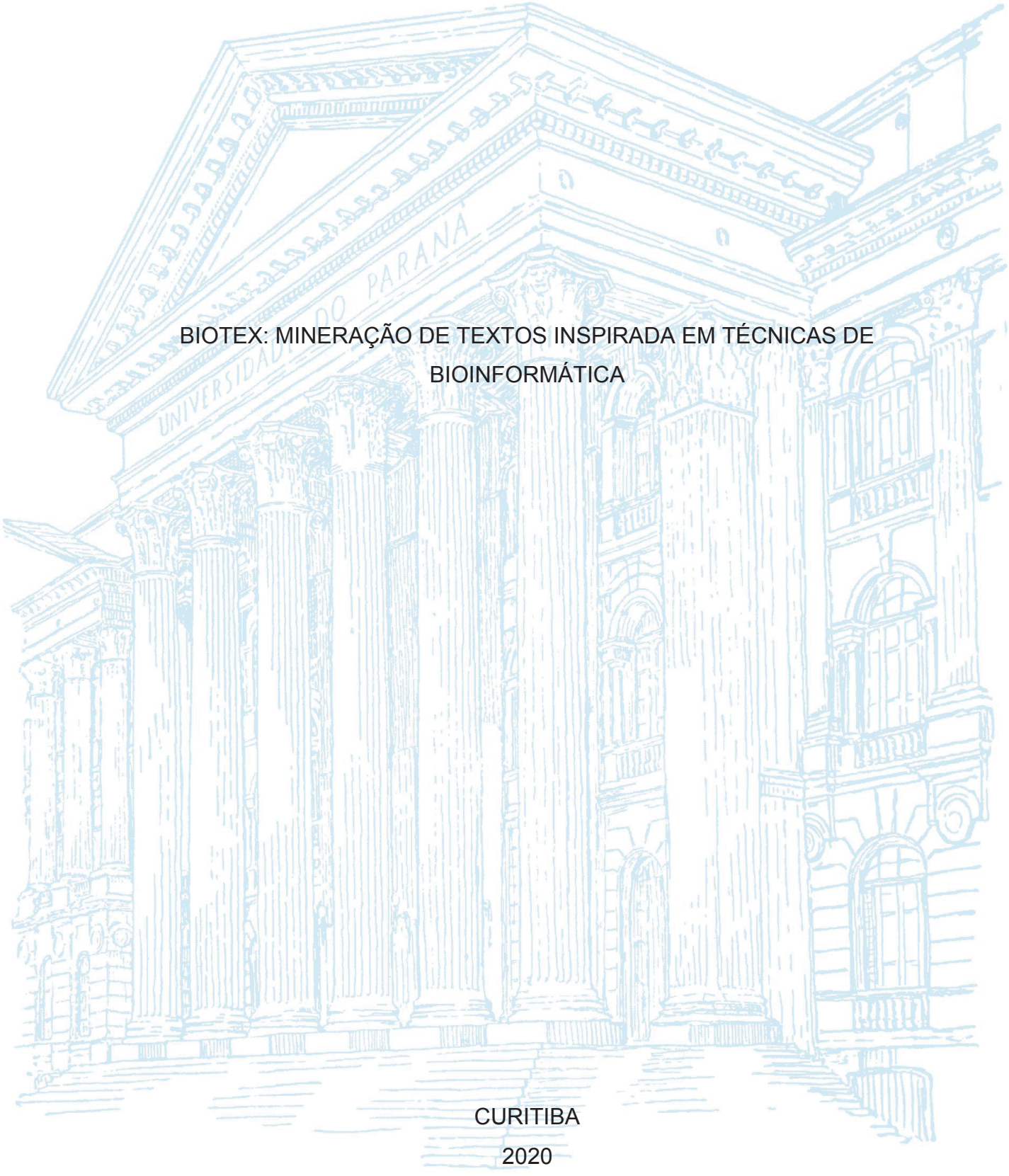
UNIVERSIDADE FEDERAL DO PARANÁ

DIOGO DE JESUS SOARES MACHADO



BIOTEX: MINERAÇÃO DE TEXTOS INSPIRADA EM TÉCNICAS DE BIOINFORMÁTICA

CURITIBA

2020

DIOGO DE JESUS SOARES MACHADO

BIOTEX: MINERAÇÃO DE TEXTOS INSPIRADA EM TÉCNICAS DE BIOINFORMÁTICA

Dissertação apresentada ao curso de Pós-Graduação em Bioinformática, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Bioinformática.

Orientador: Prof. Dr. Roberto Tadeu Raittz

CURITIBA

2020

Catalogação na publicação
Sistema de Bibliotecas UFPR
Biblioteca de Educação Profissional e Tecnológica

Machado, Diogo de Jesus Soares
BioTEX: mineração de textos inspirada em técnicas de Bioinformática / Diogo de Jesus Soares Machado. - Curitiba, 2020.
66 p.: il., tabs, grafs.

Orientador: Roberto Tadeu Raittz
Dissertação (Mestrado) – Universidade Federal do Paraná, Setor de Educação Profissional e Tecnológica, Curso de Pós-Graduação em Bioinformática.

1. Mineração de dados (Computação). 2. Mineração de textos. 3. Vetorização de textos. 4. Bioinformática. I. Raittz, Roberto Tadeu. II. Título. III. Universidade Federal do Paraná.

Elaboração: Angela Pereira de Farias Mengatto - CRB 9/1002

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em BIOINFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **DIOGO DE JESUS SOARES MACHADO** intitulada: **BioTEX: MINERAÇÃO DE TEXTOS INSPIRADA EM TÉCNICAS DE BIOINFORMÁTICA**, sob orientação do Prof. Dr. ROBERTO TADEU RAITTZ, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 19 de Junho de 2020.

Assinatura Eletrônica
29/06/2020 18:06:06.0
ROBERTO TADEU RAITTZ
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
17/09/2020 14:00:59.0
ROBERTO CARLOS DOS SANTOS PACHECO
Avaliador Externo (UNIVERSIDADE FEDERAL DE SANTA CATARINA)

Assinatura Eletrônica
29/06/2020 11:27:59.0
ROBERTO HIROCHI HERAI
Avaliador Externo (PONTIFICA UNIVERSIDADE CATÓLICA DO PARANÁ)

Aos meus pais,
ao meu irmão,
a toda minha família,
a todos os meus professores.

AGRADECIMENTOS

Agradeço aos meus pais, pelo suporte no decorrer de todos esses anos. Muitos exemplos de esforços para o desenvolvimento dos filhos. Tenho dúvidas se eu seria capaz de tudo isso.

Agradeço ao meu irmão pelo companheirismo e disposição sempre que necessário, bem como por ter influenciado muito na minha perspectiva do mundo, muito além do que meus olhos poderiam ver.

Agradeço aos meus professores no decorrer de toda minha formação até o momento, em especial ao meu orientador.

Agradeço aos artistas que criaram as músicas e os universos fictícios fantásticos que tanto admiro, seja através de jogos, animes, livros, filmes ou outros. Eles mantêm minha sanidade e ajudam muito na clareza na tomada de minhas decisões.

Agradeço a todos que tenham colaborado com meu trabalho em algum momento, seja diretamente ou indiretamente.

“O conhecimento é uma arma, Jon. Arme-se bem antes de partir para a batalha.”
(Frase do personagem fictício Meistre Aemon em “O Festim dos Corvos”, livro de George R. R. Martin)

RESUMO

A mineração de textos trata da obtenção de informação a partir do processamento de dados não estruturados, escritos em linguagem natural. A grande quantidade de conteúdo textual digitalizado disponível através da internet propiciou o interesse no desenvolvimento de técnicas envolvendo processamento de linguagem natural e aprendizado de máquina. Da mesma forma que ocorre com os textos também ocorre com dados de origem biológica. Os dados genômicos, proteômicos e transcriptômicos muitas vezes são disponibilizados na forma de arquivos FASTA, que são arquivos de texto com uma estrutura específica. Para trabalhar com esses dados foram desenvolvidas muitas ferramentas destinadas para bioinformática. A partir dessas observações, é válido supor a possibilidade de transformar textos escritos em linguagem natural para um formato baseado na representação de sequências biológicas, para propiciar a aplicação de ferramentas de bioinformática em estratégias de mineração de textos, ampliando o arsenal de recursos disponíveis para a área. Para possibilitar o avanço nessa abordagem, desenvolvemos um pacote em Python que chamamos de “BioTEX”, que oferece recursos para codificar textos para um formato baseado na representação de sequências biológicas, além de outros módulos para auxiliar no processo de mineração de textos através da estratégia proposta. Apresentamos um estudo de caso em que obtivemos artigos do PubMed e aplicamos o BioTEX para gerar um dendrograma de palavras, com o qual demonstramos indícios da ascensão da pandemia do SARS-CoV-2 apenas utilizando textos escritos até outubro de 2019, ou seja, identificamos indícios de eventos presentes em literatura passada.

Palavras-chave: Mineração de textos. Codificação de textos. Vetorização de textos. Bioinformática.

ABSTRACT

Text mining deals with obtaining information from the processing of unstructured data, written in natural language. The large amount of digitized textual content available over the internet has sparked interest in the development of techniques involving natural language processing and machine learning. As with texts, it also occurs with data of biological origin. Genomic, proteomic and transcriptomic data are often made available in the form of FASTA files, which are text files with a specific structure. To work with this data, many tools designed for Bioinformatics were developed. Based on these observations, it is valid to assume the possibility of transforming texts written in natural language into a format based on the representation of biological sequences, to provide the application of bioinformatics tools in text mining strategies, expanding the arsenal of resources available for the area. In order to advance this approach, we developed a Python package that we call “BioTEX”, which offers resources to encode texts into a format based on the representation of biological sequences, in addition to other modules to assist in the text mining process through the strategy proposed. We present a case study in which we obtained articles from PubMed and applied BioTEX to generate a word dendrogram, with which we demonstrate evidence of the rise of the SARS-CoV-2 pandemic only using texts written until October 2019, that is, we identified evidence of events present in past literature.

Keywords: Text mining. Text encoding. Text vectorization. Bioinformatics.

LISTA DE FIGURAS

FIGURA 1 - CORTE PARA IDENTIFICAÇÃO DE PALAVRAS SIGNIFICANTES.....22
FIGURA 2 - JANELA DE ENTIDADES DE TEXTO.....26
FIGURA 3 - ESQUEMATIZAÇÃO DO ALGORITMO SVM.....29
FIGURA 4 - COMPONENTES DO BIOTEX.....37
FIGURA 5 – DEMONSTRAÇÃO DE ALINHAMENTO DE TEXTOS CODIFICADOS41
FIGURA 6 - ESQUEMATIZAÇÃO DA CODIFICAÇÃO DO DNABITS.....42
FIGURA 7 - ALGORITMO HUFLUS.....44
FIGURA 8 - HEDGE DE CONCENTRAÇÃO.....47

LISTA DE GRÁFICOS

GRÁFICO 1 - EXEMPLO DE OBSERVAÇÕES BIDIMENSIONAIS 45
GRÁFICO 2 - RESULTADO DA CLUSTERIZAÇÃO HIERARQUICA 46
GRÁFICO 3 - ILUTRAÇÃO DO PROCESSO DE SUAVIZAÇÃO FUZZY 48
GRÁFICO 4 - RESULTADO DA CLUSTERIZAÇÃO PARA O EXEMPLO 49
GRÁFICO 5 - RECLASSIFICAÇÃO DE *CLUSTERS* DO HFCLUS.....50
GRÁFICO 6 - RAMO DA ÁRVORE DE PALAVRAS (DEFINIÇÃO DA DOENÇA) 54
GRÁFICO 7 - RAMO DA ÁRVORE DE PALAVRAS (POTENCIAL INFECCIOSO) ..55
GRÁFICO 8 - RAMO DA ÁRVORE DE PALAVRAS (FUTURO).....55
GRÁFICO 9 - RAMO DA ÁRVORE DE PALAVRAS (APRESENTAÇÃO) 56
GRÁFICO 10 - RAMO DA ÁRVORE DE PALAVRAS (VACINAS) 56

LISTA DE QUADROS

QUADRO 1 - COMPARAÇÃO ENTRE LEMATIZAÇÃO E STEMIZAÇÃO.....24
QUADRO 2 - DICIONÁRIO DO AMINOCODE.....39
QUADRO 3 - DEMONSTRAÇÃO DO AMINOCODE 40
QUADRO 4 - DICIONÁRIO DO DNABITS 42
QUADRO 5 - DEMONSTRAÇÃO DO DNABITS.....43

LISTA DE TABELAS

TABELA 1 - *CLUSTERS* PREDITOS PELO HUFLUS 53

LISTA DE ABREVIATURAS OU SIGLAS

COVID-19	*Coronavirus Disease 2019*
DM	*Data Mining* (Mineração de Dados)
HMM	*hidden markov model* (modelo oculto de Markov)
IDF	*inverse document frequency* (inverso da frequência nos documentos)
IE	*Information Extraction* (Extração de Informação)
IR	*Information Retrieval* (Recuperação de Informação)
KNN	*K-Nearest Neighbors* (K-ésimo vizinho mais próximo)
MLP	*Multilayer perceptron* (Perceptron multicamadas)
NLP	*Natural Language Processing* (Processamento de Linguagem Natural)
NLTK	*Natural Language Toolkit*
SARS	*severe acute respiratory syndrome* (síndrome respiratória aguda grave)
SARS-CoV-2	*severe acute respiratory syndrome coronavirus 2* (síndrome respiratória aguda grave de coronavírus 2)
SVM	*Support vector machine* (Máquina de vetores de suporte)
TF	*term frequency* (frequência de termo)

LISTA DE SÍMBOLOS

$\propto$ proporcional a

Π produtório

| condicional a

SUMÁRIO

1 INTRODUÇÃO 17
1.1 JUSTIFICATIVA 17
1.2 OBJETIVOS 18
1.2.1 Objetivo geral 18
1.2.2 Objetivos específicos 18
1.3 METODOLOGIA 18
2 REVISÃO DE LITERATURA 19
2.1 MINERAÇÃO DE TEXTOS 19
2.2 PRÉ-PROCESSAMENTO PARA MINERAÇÃO DE TEXTOS 20
2.2.1 Tokenização 21
2.2.2 Remoção de *Stop Words* 21
2.2.3 Stemização 23
2.2.4 Lematização 23
2.3 VETORIZAÇÃO DE TEXTOS 24
2.3.1 Vetorização pela contagem de termos 24
2.3.2 Vetorização pela ocorrência binária de termos 25
2.3.3 Vetorização pela definição do TF-IDF 25
2.3.4 Vetorização por janela deslizante 26
2.3.5 Vetorização por SWeeP 27
2.3.6 Outros possíveis atributos e vetorização composta 27
2.4 APRENDIZADO DE MÁQUINA EM MINERAÇÃO DE TEXTOS 28
2.4.1 Aprendizado Supervisionado 28
2.4.1.1 SVM 29
2.4.1.2 MLP 30
2.4.1.3 KNN 31
2.4.1.4 Naive Bayes 31
2.4.2 Aprendizado Não-supervisionado 31
2.4.2.1 Clusterização Hierárquica 32
2.4.2.2 Clusterização partitiva 33
2.4.3 Aprendizado Semi-supervisionado 33
2.4.4 Combinação de algoritmos 34
2.5 CONSIDERAÇÕES FINAIS DA REVISÃO 34

3 MATERIAL E MÉTODOS 36
3.1 TECNOLOGIAS UTILIZADAS E DESENVOLVIMENTO DOS ALGORITMOS ... 36
3.2 RECURSOS DESENVOLVIDOS 36
3.3 ESTUDO DE CASO 37
4 APRESENTAÇÃO DOS RESULTADOS 38
4.1.1 AMINOcode 38
4.1.2 DNAbits 42
4.1.3 Huflus 43
4.1.3.1 Predição dos centroides 44
4.1.3.2 Refinamento das bordas 49
4.1.4 Collsci 50
4.1.5 Recursos adicionais 51
4.2 ESTUDO DE CASO 52
4.2.1 Descrição do estudo de caso 52
4.2.2 Resultado do estudo de caso 53
5 CONCLUSÃO 57
5.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS 57
REFERÊNCIAS 59
ANEXO 1 – ALGORITMO PORTER STEMMER 64

1 INTRODUÇÃO

A computação e a internet tornaram a produção e disponibilização de materiais textuais extremamente eficientes, no entanto isso repercutiu em uma enorme quantidade de material, impossível para leitura e intepretação dadas as capacidades de um indivíduo (Berry; Castellanos, 2007).

Buscando remediar tal problemática, através de estratégias de mineração de textos, é possível o uso de abordagens computacionais para extração de conhecimento a partir de um corpus (conjunto de textos) (Qi et al., 2009).

Da mesma forma que ocorre com os textos, também ocorre com os dados biológicos. Para tratamentos da grande quantidade de informações que a evolução das técnicas laboratoriais trouxe, muitas ferramentas e estratégias computacionais foram desenvolvidas, conformando a bioinformática.

Dentro da bioinformática é muito comum a utilização de dados biológicos representados através de cadeias de caracteres, como exemplo os arquivos FASTA. Nos arquivos FASTA sequencias biológicas são representadas com um conjunto de letras (Ayyildiz; Piazza, 2019).

Sendo as sequências nos arquivos FASTA um conjunto de caracteres que representam uma informação, é possível considerar que não são muito diferentes de textos escritos em linguagem natural, apenas representam dados distintos.

Na perspectiva apresentada, é interessante o levantamento de uma estratégia que faz uso de textos escritos em linguagem natural, codificados em um formato baseado na representação dos arquivos FASTA. Desse modo, muitos métodos de bioinformática podem ser passíveis de aplicação em textos escritos em linguagem natural.

1.1 JUSTIFICATIVA

Muitas das ferramentas e estratégias de bioinformática utilizam algoritmos que têm como entrada dados em formato de texto, portanto é possível considerar a utilização de uma estratégia de codificação de textos escritos em linguagem natural para um formato baseado em dados biológicos. Através de tal recurso pode ser definida uma interseção entre bioinformática e mineração de textos, aumentando significativamente o arsenal de recursos da mineração de textos.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver ferramentas para dar suporte para o uso dos recursos de bioinformática em estratégias de mineração de textos.

1.2.2 Objetivos específicos

- Definir estratégias de codificação de texto para formado baseado em arquivos FASTA;
- Definir possíveis estratégias de mineração de textos utilizando o formato codificado;
- Executar a implementação dos recursos definidos;
- Executar um estudo de caso prático com o resultado.

1.3 METODOLOGIA

Prototipamos algoritmos computacionais para execução dos objetivos utilizando a linguagem de programação MATLAB. Após a prototipagem, os algoritmos foram reescritos na linguagem Python, para serem disponibilizados gratuitamente via internet. Também fizemos a demonstração de uso da ferramenta desenvolvida através de um estudo de caso.

2 REVISÃO DE LITERATURA

Para a compreensão adequada da metodologia e discussão deste trabalho, algumas diretrizes são necessárias. Em primeiro lugar uma definição clara e estado da arte sobre a mineração de textos, seguida pela resenha a respeito de tecnologias que são associadas à estratégia desenvolvida: processamento de linguagem natural, vetorização de textos e aprendizado de máquina em mineração de textos.

2.1 MINERAÇÃO DE TEXTOS

A essência da mineração de textos, segundo Qi et al. (2009), é a descoberta de informações previamente desconhecidas, tendo como entrada um conjunto de textos escritos em linguagem natural. Em sua revisão de literatura, os autores citam que essa característica diferencia a mineração de textos de tecnologias que são associadas a ela, mas, considerando as definições atuais, não são a mesma coisa:

a) Mineração de Dados (*Data Mining* - DM): descoberta de informações com base em dados estruturados. O foco é a entrada de dados já estruturados, diferente da mineração de textos, em que as entradas são textos não estruturados;
b) Recuperação de Informação (*Information Retrieval* - IR): é a recuperação de informações armazenadas, não inclui a obtenção de novas informações;
c) Extração de Informação (*Information Extraction* - IE): no contexto da manipulação de textos, consiste na obtenção de um conjunto de dados estruturados, a partir uma base de textos escritos em linguagem natural;
d) Processamento de Linguagem Natural (*Natural Language Processing* - NLP): área que estuda a automação no entendimento e geração de textos escritos em linguagem natural.

As terminologias citadas são referentes à obtenção e formatação dos dados, mas a mineração de textos também inclui o processamento inteligente via aprendizado de máquina (Berry; Castellanos, 2007). A mineração de textos efetivamente é uma combinação de um conjunto de técnicas.

2.2 PRÉ-PROCESSAMENTO PARA MINERAÇÃO DE TEXTOS

Antes da execução da mineração de textos propriamente dita, é comum a aplicação de métodos de pré-processamento, que tornam os textos mais viáveis para o processamento computacional, dado o algoritmo utilizado.

Os métodos de pré-processamento podem variar, mas alguns são frequentemente encontrados em trabalhos atuais (Anees et al., 2020 e Banik et al., 2019 e Bhalla, 2019):

a) Tokenização (*Tokenizetion*);
b) Remoção de *stop words* (*Stop Word Removal*);
c) Stemização (*Stemming*);
d) Lematização (*Lemmatization*).

Os métodos de pré-processamento podem ser classificados como recursos da IR, visto que são relacionados com a obtenção de dados existentes em uma base textual. Por sua vez, a IR pode ser considerada uma subdivisão da NLP, já que a recuperação de informação é usualmente vinculada com buscas textuais, apesar de cada vez mais a recuperação de dados ser aplicada em imagens, áudios ou vídeos também. No entanto, em aspectos práticos, a interação entre IR e NLP acaba sendo bastante limitada, devido os problemas e soluções gerais da IR não serem o foco da NLP (Manning; Schiitze, 1999).

Frequentemente os métodos de pré-processamento são encontrados em bibliotecas de linguagens de programação focadas em NLP, como é o caso do “Natural Language Toolkit” (NLTK), da linguagem Python, conforme pode ser verificado em sua documentação oficial, acessível pelo endereço <https://www.nltk.org/>. Alguns autores também colocam o pré-processamento de textos como parte do NLP em suas metodologias, conforme ocorre no artigo de Runeson et al. (2007).

2.2.1 Tokenização

Visando a simplificação de textos para um formato mais adequado para o processamento, pode ser necessária reestruturação do conteúdo em unidades menores.

A tokenização é o processo de obtenção de unidades mínimas com significado atribuído, chamadas de tokens (Manning; Schiitze, 1999). Um exemplo de processo de tokenização é a obtenção das palavras que compõe o corpus, sendo os tokens nesse caso as palavras.

2.2.2 Remoção de *Stop Words*

Stop Words são palavras funcionais que possuem importância semântica, mas que podem ser ignoradas na recuperação orientada por palavras, sem reduções significativas na acurácia. A remoção de *stop word*, portanto, é o processo de remoção dessas palavras (Manning; Schiitze, 1999).

A vantagem para a remoção de *stop words* pode ser explicada pela Lei de Zipf, apresentada pelo Linguista George Kingsley Zipf, no livro "*Human Behavior and the Principle of Least Effort*". A Lei de Zipf é determinada pela equação:

$$f \propto \frac{k}{r} \qquad (1)$$

, onde *r* é a posição de uma determinada palavra em uma lista, com todas as palavras que aparecem em um corpus ordenadas por frequência de ocorrência em ordem decrescente; *f* é a frequência da palavra na posição *r*; e *k* é uma constante específica para o conjunto de palavras. OBS.: O símbolo $\propto$ significa "proporcional", ou seja, representa que todos os pares possíveis entre *f* e *r*, são proporcionais. Em outras palavras, segundo a lei de Zipf, em um corpus escrito em linguagem natural a frequência das palavras reduz seguindo um padrão.

A Lei de Zipf não é absoluta, mas é uma caracterização que descreve de forma aproximada os fatos (Manning; Schiitze, 1999). Indícios da Lei de Zipf já foram identificados em vários idiomas através de trabalhos distintos, entre eles:

a) Chinês (Shtrikman, 1994);
b) Grego (Hatzigeorgiu et al., 2001);

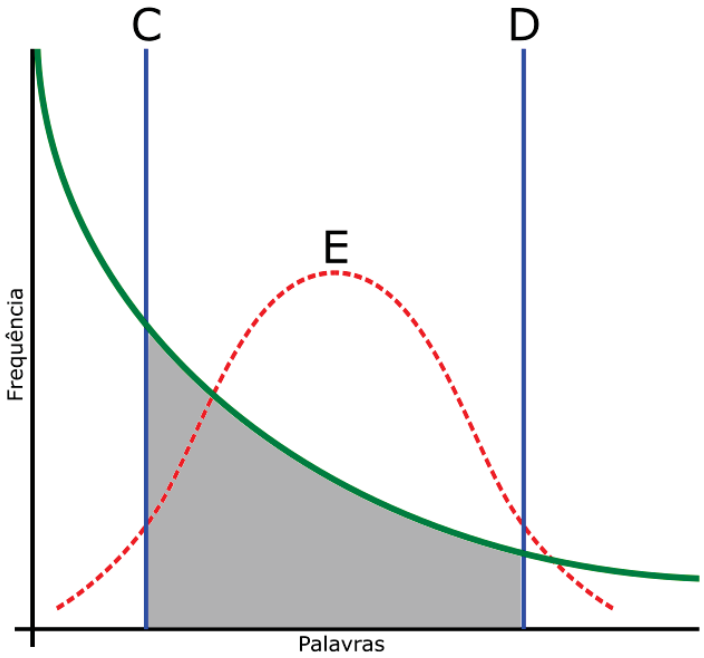
c) Inglês (Zipf, 1949);

d) Português (Lima; Silva, 1973).

Stop words tendem a estarem entre os primeiros *ranks* de uma lista ordenada por frequência. Com base na Lei de Zipf, é natural que ao eliminar poucas *stop words*, o conjunto total de palavras é bastante reduzido. A remoção de poucas dúzias de *stop words*, é suficiente para reduzir pela metade o tamanho do índice invertido[1] de um corpus (Manning; Schiitze, 1999).

Luhn (1958) também citou e utilizou em seu trabalho o corte de palavras com elevada ocorrência. Ele propôs a determinação de um ponto inferior e superior na lista de palavras ordenadas por frequência, estando na área entre as linhas de corte as palavras mais significantes. A FIGURA 1 representa a estratégia do autor. No eixo *x* são representadas as palavras, no eixo *y* as frequências. As linhas “C” e “D” esquematizam os cortes superior e inferior, que delimitam uma área central que representa as palavras mais significantes. Por fim, a curva “E” ilustra o grau de significância das palavras.

FIGURA 1 - CORTE PARA IDENTIFICAÇÃO DE PALAVRAS SIGNIFICANTES



FONTE: Adaptado de Luhn (1958).

[1] O índice invertido é a estrutura de dados que relaciona todas as palavras que ocorrem em corpus com os documentos que as contêm, junto com as respectivas frequências em cada documento.

A remoção de *stop words* tem fundamento e pode ser relevante, mas nem sempre é aplicada, visto que em alguns casos é desejado não perder informações nas frases em que elas ocorrem (Manning; Schiitze, 1999).

2.2.3 Stemização

É chamado de stemização o processo de reduzir as palavras de um corpus para um radical, agrupando flexões verbais e derivações de palavras em um mesmo tronco (*stem*). O objetivo da stemização é agrupar palavras que apresentam o mesmo significado central.

O conceito da stemização pode ser aplicado através de diferentes algoritmos, mas um muito utilizado na Língua Inglesa atualmente é o “Porter Stemmer”, descrito por Porter (1980). O “Porter Stemmer” é extremamente popular, sendo que Willett (2006) descreve a técnica como abordagem padrão para fusão de palavras.

No ANEXO 1 é esquematizado o algoritmo “Porter Stemmer” original, publicado por Porter (1980), no entanto o autor disponibilizou alguns anos depois um *website* com adaptações do algoritmo para outras línguas, incluindo a Língua Portuguesa. O endereço paro o *website* em questão é <http://snowball.tartarus.org/>, (Porter, 2001).

Porter (2001) cita que o processo de stemização pode apresentar alguns problemas, entre eles:

a) “over-stemming”: ocorre quando palavras que não são flexões verbais ou derivações de uma mesma palavra são caracterizadas com um mesmo tronco;
b) “under-stemming” ocorre quando palavras que são flexões ou derivações de uma mesma palavra não são caracterizadas pelo mesmo tronco;
c) “miss-stemming”: ocorre quando é feita a remoção de letras que deveriam fazer parte do tronco da palavra.

2.2.4 Lematização

A lematização tem um objetivo semelhante à stemização, mas apresenta um conceito distinto. Enquanto o objetivo da stemização é o agrupamento de palavras com o mesmo radical em um único tronco, o objetivo da lematização é a obtenção do

“lema” das palavras, ou seja, a palavra base que derivou as diversas flexões de acordo com o significado no dicionário, seguindo uma análise morfológica (Anees et al., 2020). O QUADRO 1 demonstra a comparação entre stemização e lematização, nele é possível notar que na stemização ocorre um corte na palavra original, enquanto na lematização ocorre a obtenção da forma não flexionada.

QUADRO 1 - COMPARAÇÃO ENTRE LEMATIZAÇÃO E STEMIZAÇÃO

Palavra	**Lematização**	**Stemização**
was	be	wa
studies	study	studi
studying	study	study

FONTE: Anees et al. (2020).

2.3 VETORIZAÇÃO DE TEXTOS

A vetorização de textos é a obtenção de um conjunto de valores numéricos que representem dados textuais. Várias estratégias podem ser adotadas para isso, seguem algumas encontradas em artigos científicos:

a) Frequência de termos;
b) Ocorrência de termos;
c) TF-IDF;
d) Janela deslizante.

A vetorização torna os textos mais susceptíveis ao processamento matemático, o que facilita análises estatísticas e aprendizado de máquina através dos algoritmos atuais.

2.3.1 Vetorização pela contagem de termos

A vetorização por contagem de palavras consiste na listagem de todas as palavras de um corpus, seguida pela contagem dessas palavras em cada um dos documentos, mesmo que igual a zero. Cada palavra em cada documento tem uma quantidade, conformando uma matriz.

O uso dessa estratégia é feito por exemplo no trabalho de Khaleghi et al. (2019), onde fazem a construção de uma matriz com a frequência de termos em textos de um corpus relacionado a cirurgias.

2.3.2 Vetorização pela ocorrência binária de termos

Semelhante à vetorização pela contagem de palavras, a ocorrência binária apenas varia no fato que não retorna a quantidade de vezes que cada palavra ocorre, apenas se ocorre ou não. Ou seja, os vetores são populados apenas com os números 0 e 1, indicando respectivamente que a palavra não ocorre ou ocorre no documento.

Alkalbani et al. (2016), por exemplo, obtiveram bons resultado ao utilizar vetorização por ocorrência binária na classificação de sentimentos em textos de opiniões, com SVM (do inglês *support vector machine*, algoritmo descrito no tópico 2.4.1.1).

2.3.3 Vetorização pela definição do TF-IDF

A TF-IDF é uma métrica para medição da significância de palavras em um texto, dado o corpus no qual está inserido. A métrica IDF (do inglês *inverse document frequency*, que significa frequência inversa de documentos), definida por Jones (1972), foi combinada com a TF (do inglês *term frequency*, que significa frequência de termo), conformando o que é conhecido hoje como TF-IDF, uma métrica extremamente importante e robusta (Robertson, 2004).

Palavras que ocorrem em vários documentos dentro de um corpus, devem receber um peso menor do que as que ocorrem em poucos documentos. Quanto mais exclusiva a palavra, mais representativa ela é (Jones, 1972).

A fórmula mais comum para a IDF é:

$$idf(t_i) = log \frac{N}{n_i} \qquad (2)$$

, onde $idf(t_i)$ é o IDF para o termo t_i; N é o número total de documentos no corpus; e n_i é o número de documentos que contém o termo t_i.

Baseada na combinação da IDF com a TF, a métrica TF-IDF é obtida por:

$$tf\text{-}idf(t_{i,d}) = tf_{i,d} \times idf(t_i) \quad (3)$$

, onde $tf\text{-}idf(t_{i,d})$ é a métrica TF-IDF para o termo *i* no documento *d*; e $tf_{i,d}$ é a frequência do termo *i* no documento *d*.

Cada termo em cada documento tem um valor de TF-IDF, conformando uma matriz.

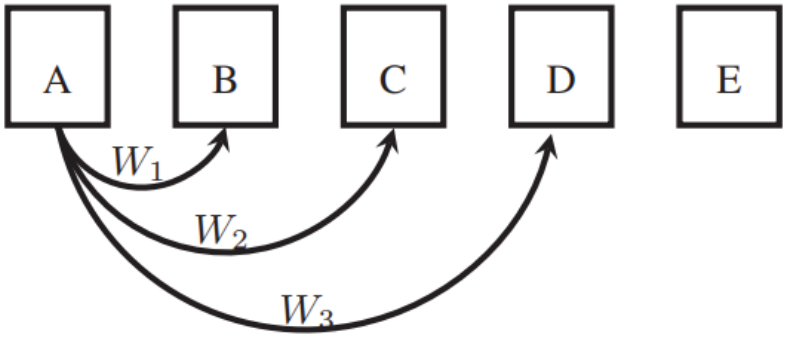
Exemplificando, Simon et al. (2019), no trabalho em que descrevem a ferramenta BioReader, citam a utilização de TF-IDF para vetorização de um corpus de *abstracts* de artigos científicos.

2.3.4 Vetorização por janela deslizante

Na vetorização por janela deslizante os termos não são tratados individualmente, mas sim em intervalos (janelas).

Fantozzi et al. (2018) definiram um exemplo de método em que é definida uma janela deslizante de entidades para percorrer um texto. As entidades podem ser as palavras ou uma coleção de itens. Cada coocorrência de entidades dentro de uma mesma janela é contabilizada, no entanto um peso é atribuído de acordo com a distância entre as entidades. A FIGURA 2 esquematiza a estratégia, onde a coocorrência (A, B) tem peso W_1, (A, C) tem peso W_2 e (A, D) tem peso W_3. No exemplo a janela deslizante é definida com o tamanho 4, portanto a coocorrência (A, E) não é verdadeira, não sendo contabilizada.

FIGURA 2 - JANELA DE ENTIDADES DE TEXTO



FONTE: Fantozzi et al. (2018).

As presenças de coocorrência de entidades em um documento podem ser utilizadas como atributos de um vetor. No caso da estratégia de Fantozzi et al. (2018) cada atributo é ponderado pelo peso, no entanto isso não é essencialmente obrigatório.

2.3.5 Vetorização por SWeeP

Descrito por De Pierri et al. (2020), o SWeeP é um algoritmo de vetorização originalmente dedicado para vetorização de sequência biológicas em formato FASTA, no entanto, conforme descrito na introdução, arquivos FASTA são arquivos de texto em um formato específico dedicados para representação de uma categoria de dados. No artigo é citada a utilização da ferramenta para vetorização de sequências de aminoácidos, portanto é aplicável em *strings* com os 20 caracteres que são utilizados na representação de aminoácidos.

Tal como o método descrito no tópico anterior, o SWeeP também é baseado em janela deslizante. Nas configurações padrões, o SWeeP utiliza uma janela deslizante com cinco caracteres, ignora o caractere central e marca uma matriz de acordo com a coocorrência da combinação de caracteres laterais.

2.3.6 Outros possíveis atributos e vetorização composta

As estratégias de vetorização podem ir muito além das citadas, pode inclusive existir combinações de mais de uma estratégia para compor uma matriz. O trabalho de Mitsumori et al. (2005) é um bom exemplo disso. Os autores propuseram uma estratégia para identificar termos com significado biológico dentro de um corpus, utilizando SVM (*Support vector machine* - Máquina de vetores de suporte), um algoritmo de aprendizado de máquina supervisionado (mais sobre na seção 2.4). Cada linha da matriz criada no trabalho representava um termo (token) no corpus e as colunas continham os seguintes atributos:

a) a característica ortográfica do token (digito, letra maiúscula, letra minúscula, misto entre maiúscula e minúscula, vírgula, letra grega etc.);
b) prefixos do token quando possível (com uma, duas e três letras);
c) sufixos do token quando possível (com uma, duas e três letras);
d) classe gramatical do token quando possível;

e) binário representando a presença ou não do token em um dicionário de proteínas e genes, considerando do token sozinho, e a presença dele em conjunto com um ou dois tokens posteriores a ele no dicionário de proteínas. Este foi um atributo externo aos textos de entrada, que os autores utilizaram para avaliar a eficiência e necessidade de utilização. Chegaram à conclusão que foi positivo, no entanto pouco significativo.

2.4 APRENDIZADO DE MÁQUINA EM MINERAÇÃO DE TEXTOS

Podem ser aplicadas em documentos textuais as três abordagens diferentes de aprendizado de máquina: supervisionado, não-supervisionado e semi-supervisionado (Baharudin et al., 2010).

2.4.1 Aprendizado Supervisionado

O aprendizado supervisionado é referente à classificação dentro de categorias pré-definidas, o que é de grande interesse na mineração de textos, considerando a grande quantidade de material textual disponível via internet (Baharudin et al., 2010). Com o aprendizado de máquina supervisionado, a partir de um conjunto de observações[2] cuja classificação é conhecida, é feito um treinamento que torna possível a identificação das classes de observações previamente desconhecidas.

Algumas das estratégias nessa categoria que já foram utilizados em mineração de textos são: SVM (*support vector machine*), redes neurais, NB (Nayve Bayes), KNN (*K-nearest neighbor*), árvore de decisões, LSA (*Latent Semantic Analysis*), algoritmo Rocchio's, correlação *fuzzy* e algoritmo genético (Baharudin et al., 2010).

Redes neurais são muito utilizada na classificação de observações e a MLP (*multi-layer perceptron*) é a arquitetura de rede mais popular (Shafie et al., 2012), mas Baharudin et al. (2010) citam que os algoritmos mais apropriados para classificação e documentos, considerando uma revisão de literatura que fizeram, são: SVM, NB e

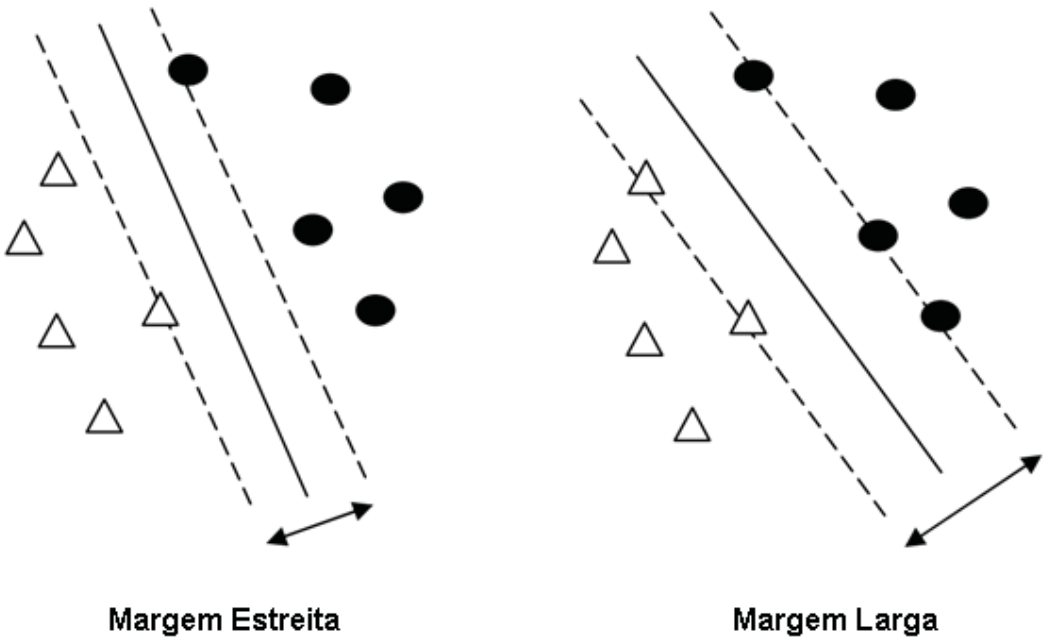
[2] Observações são um conjunto de atributos (coordenada) que representam uma unidade de alguma entidade.

KNN. Considerando essas afirmações, seguem detalhamentos para esses quatro algoritmos.

2.4.1.1 SVM

Usado por exemplo no trabalho de Mitsumori et al. (2005), o SVM (do inglês *support vector machine*, que significa máquina de vetores de suporte) visa identificar o hiperplano (ou linha, no caso de observações com duas dimensões) que melhor divide as observações com a máxima distância possível, para isso ele cria hiperplanos de suporte para identificação da posição do hiperplano ótimo, por isso o nome do algoritmo. A FIGURA 3 esquematiza o funcionamento do algoritmo. Para tratar observações que não podem ser separados por um hiperplano, o SVM possibilita a utilização de funções kernel, que, neste contexto, são funções matemáticas que podem ser aplicadas nos vetores para reposicioná-los de forma que passem a ser separáveis.

FIGURA 3 - ESQUEMATIZAÇÃO DO ALGORITMO SVM



FONTE: Adaptado de Mitsumori et al. (2005).

LEGENDA: Os círculos e os triângulos representam dados de diferentes classes, as linhas tracejadas representam os vetores de suporte e a linha sólida representa o hiperplano com maior distância em relação aos vetores de suporte. Comparando as duas representações na figura, a imagem à direita demonstra um hiperplano que divide as classes com uma maior distância, portanto seria a selecionada como resultado.

Uma sessão de treinamento do SVM pode classificar apenas entre duas classes, no entanto existem artifícios para contornar esse problema utilizando mais de uma sessão do SVM. Mitsumori et al. (2005) utilizaram três classes em seu trabalho, uma classificação que chamaram de “BIO”. O objetivo foi classificar tokens em artigos entre: tokens significativos biologicamente (representados pela letra “B”), tokens que complementam o sentido de “B” (representados pela letra “I”) e tokens fora de qualquer contexto biológico (representados pela letra “O”). Para aplicar o SVM nessa configuração os autores optaram pela realização de três sessões de treinamento (“B” vs. “I”, “B” vs. “O” e “I” vs. “O”), portanto três hiperplanos foram formados. Na classificação, a classe ideal de cada token foi definida a partir da classe que tinha o valor máximo nas três funções do hiperplano.

2.4.1.2 MLP

Perceptron é um algoritmo que faz uma alusão ao funcionamento dos neurônios biológicos. Quando existe apenas uma camada de neurônio é utilizado a denominação “perceptron”, já quando existe mais de uma camada é utilizada a denominação MLP (Ghosh et al., 2012), do inglês *multi-layer perceptron*, que significa perceptron multicamadas.

Os atributos entram na rede MLP e, conforme eles passam pelos neurônios artificiais através dos axônios, pesos são atribuídos. Nos neurônios são aplicadas funções de ativação, que transformam o(s) valor(es) na camada anterior. Existe mais de uma opção de função de ativação, sendo que a escolha afeta significativamente o treinamento. Para o treinamento da MLP um dos algoritmos mais populares é o *backpropagation* (retropropagação). No algoritmo, primeiramente são determinados pesos aleatórios para cada axônio, mas em cada passagem das observações de treinamento pela rede, após calcular o erro na classificação, a correção dos pesos ocorre regressando dos últimos neurônios até os primeiros. O processo é repetido até alcançar algum critério de parada (Shafie et al., 2012).

Exemplificando a utilização de MLP em mineração de textos, Yan e Wong (2020) desenvolveram uma estratégia para identificação de descrições de eventos biomédicos em literatura científica.

2.4.1.3 KNN

O KNN (do inglês *K-nearest neighbor*, que significa K-ésimo vizinho mais próximo) é um algoritmo para classificação de observações com classes desconhecidas, através da identificação da proximidade com as coordenadas de observações com classes conhecidas. Para tanto é determinado um valor para *K*, que representa o número de vizinhos da observação a ser classificada que serão avaliados. A classe que predominar nos vizinhos é utilizada como classe para a observação com classificação previamente desconhecida (Baharudin et al., 2010).

2.4.1.4 Naive Bayes

O Naive Bayes é baseado no teorema de Bayes, sendo um algoritmo probabilístico. A probabilidade de uma observação com classe previamente desconhecida estar inclusa em uma determinada classe (C_j), é determinada pelo produtório das probabilidades de cada atributo dentro da classe questão ($P(A_i|C_j)$), considerando o conjunto de treinamento composto por observações com classes conhecidas, junto com a probabilidade da classe ocorrer no conjunto de treinamento ($P(C_j)$). A classificação para uma observação (v_{NB}) então é dada pela identificação do máximo (*arg max*) entre as probabilidades dela estar dentro de cada classe. Em representação matemática (Zhang; Li, 2007):

$$v_{NB} = \underset{C_j \in C}{arg\ max}\, P(C_j) \prod_i P(A_i|C_j) \qquad (4)$$

Zhang e Li (2007) apresentaram em seu trabalho a utilização do Naive Bayes para classificação de *e-mails* como *spam* ou não *spam*.

2.4.2 Aprendizado Não-supervisionado

Agrupamento ou clusterização (*clustering*) é o processo de agrupar uma população de observações em conjuntos com padrões semelhantes. Os algoritmos de agrupamento são adotados para executar classificação não-supervisionada de observações. O objetivo é que os elementos dentro de um grupo (*cluster*) sejam mais

semelhantes entre si do que se comparados com elementos fora do *cluster* (Jain et al., 1999).

Em corpus com grandes quantidades de textos, o agrupamento de elementos com características semelhantes é um recurso muito relevante. Por se tratar de uma estratégia não-supervisionada, a clusterização é uma boa alternativa para primeira etapa de mineração de textos, sendo muitas vezes usado como estratégia de pré-processamento. Uma aplicação importante da identificação de *clusters* é a definição dos tópicos (tema central) de documentos (Antony; Wagh, 2017).

É possível classificar algoritmos de agrupamento em duas categorias: hierárquicos e partitivos (Steinbach et al., 2000).

2.4.2.1 Clusterização Hierárquica

A clusterização hierárquica consiste no agrupamento par a par de observações semelhantes. Uma camada superior inclui todos os ramos, os níveis intermediários combinam pares de *clusters* da camada seguinte, enquanto a última camada agrupa os pares de observações. Os resultados de clusterizações hierárquicas podem ser representados graficamente na forma de dendrogramas (Steinbach et al., 2000).

Os métodos de clusterização hierárquica podem ser classificados em aglomerativos ou divisivos. A clusterização hierárquica aglomerativa começa com todas as observações separadas, agrupando par a par no decorrer dos passos. Já na clusterização hierárquica divisiva as observações começam todas no mesmo grupo, sendo divididas no decorrer dos passos (Steinbach et al., 2000).

Exemplificando, Aich et al. (2017) utilizaram clusterização hierárquica para analisar textos no *abstract* de artigos relacionados à Doença de Parkinson. Os autores clusterizaram os termos nos textos, utilizando como atributos as frequências em cada *abstract*.

Khaleghi et al. (2019) também utilizaram clusterização hierárquica, no caso para auxiliar na identificação de erros ortográficos e abreviações em anotações sobre cirurgias de múltiplos hospitais. Palavras com o mesmo sentido tendem a apresentam uma configuração de frequências semelhante, portanto a distância entre elas na clusterização tende a ser menor. Os autores justificaram a escolha da clusterização

hierárquica pela não necessidade de determinação de um número de *clusters* como entrada.

2.4.2.2 Clusterização partitiva

Diferente da clusterização hierárquica, a clusterização partitiva divide as observações em um nível único. Frequentemente o número de *clusters* é chamado de *K*. O K-means é um exemplo de algoritmo de clusterização partitiva que pode ser utilizado na clusterização de textos (Steinbach et al., 2000).

O K-means consiste no agrupamento das observações em *K clusters*, de acordo com a proximidade com coordenadas (centroides) que representam os *clusters*. Steinbach et al. (2000) citam que etapas básicas do K-means são:

a) Selecionar *K* coordenadas para serem os centroides iniciais, com *K* sendo um valor de entrada no algoritmo;
b) Atribuir o centroide aos pontos mais próximos;
c) Definir novos centroides, obtendo a média das coordenadas de observações classificadas dentro dos *clusters*;
d) Repetir os passos “b” e “c” até não haver mais alterações na posição dos centroides.

Aich et al. (2017), citados como exemplo no tópico sobre clusterização hierárquica, também utilizaram em seu trabalho o K-means. Da mesma forma que no método hierárquico, clusterizaram os termos nos textos, utilizando como atributos as frequências em cada *abstract*.

2.4.3 Aprendizado Semi-supervisionado

Algoritmos de aprendizado semi-supervisionados são intermediários entre as abordagens supervisionadas e não-supervisionadas, sendo que geralmente o que ocorre é a extensão de estratégias de uma das categorias com elementos da outra. Um exemplo de aprendizado semi-supervisionado é clusterizar com restrições de agrupamento, seguindo regras de elementos que devem ou não estarem no mesmo *cluster* ou restringindo o tamanho dos *clusters* (Goldberg, 2009).

2.4.4 Combinação de algoritmos

K. Dalal e A. Zaveri (2011) fizeram uma revisão sobre as técnicas de aprendizado de máquina aplicadas na classificação de textos, concluindo que que para os melhores resultados devem ser avaliadas a combinação de técnicas de aprendizado de máquina.

Simon et al. (2019), por exemplo, implementaram o “Bioreader”, um classificador de artigos que recebe duas listas de PMIDs (identificadores de artigos do banco de dados PubMed), uma referente a artigos sobre um determinado tema de interesse e outra referente a artigos de temas desconhecidos. O Bioreader obtém os *abstracts* (resumos) de cada artigo, executa treinamentos utilizando a lista de artigos com tema conhecido e classifica os de tema desconhecido como pertencentes ou não ao mesmo tema dos utilizados no treinamento. Para esse processo, os autores utilizaram dez algoritmos de classificação diferentes no treinamento, sempre selecionando o melhor em cada caso através de validação cruzada do conjunto de treinamento.

2.5 CONSIDERAÇÕES FINAIS DA REVISÃO

Através da revisão apresentamos várias possibilidade de técnicas, refletindo que não existe uma abordagem padronizada para mineração de textos. Diferentes cientistas de dados utilizam critérios diferentes para selecionar as estratégias a serem utilizadas, sendo que a busca de novas técnicas é desejável para expandir ainda mais as possibilidades. As ferramentas de bioinformática podem funcionar como recurso para esse fim, ampliando as ferramentas disponíveis para a mineração de textos.

Entre as possibilidades para textos codificados em formato baseado em sequências biológicas estão:

a) Vetorização utilizando o SWeeP, método descrito por De Pierri et al. (2020). Uma vez vetorizados, é possível submeter os textos em métodos de aprendizado de máquina facilmente, além de construir dendrogramas baseados no cálculo de distância entre os vetores;
b) Alinhamentos utilizando ferramentas como o Clustal Omega, descrito por Sievers e Higgins (2018);

c) Identificação e visualização do consenso a partir de um alinhamento, utilizando formas gráficas como as logos de sequências, descritas por Schneider e Stephens (1990);
d) Representação e reconhecimento de padrões utilizando HMM (do inglês *hidden Markov model*, que significa modelo oculto de Markov), conforme método explicado por Eddy (2004).

3 MATERIAL E MÉTODOS

3.1 TECNOLOGIAS UTILIZADAS E DESENVOLVIMENTO DOS ALGORITMOS

No desenvolvimento do projeto utilizamos a linguagem de programação MATLAB para modelagem dos algoritmos, mas para distribuição desenvolvemos um pacote com a linguagem Python.

Na implementação em Python utilizamos uma série de pacotes consolidados:

a) Pandas (McKinney, 2010) e NumPy (Oliphant, 2006): manipulação de vetores e matrizes;
b) scikit-learn (Pedregosa et al., 2011) e SciPy (Virtanen et al., 2020): execução de algoritmos de aprendizado de máquina e tarefas matemáticas;
c) Matplotlib (Hunter, 2007): plotagem de gráficos;
d) Biopython (Cock et al., 2009): manipulações de sequências em formato FASTA.

Para vetorização dos textos codificados em forma de sequência utilizamos uma implementação própria em Python do SWeeP (descrito no tópico 2.3.5).

3.2 RECURSOS DESENVOLVIDOS

Denominamos o pacote que desenvolvemos de “BioTEX”, o qual disponibilizamos no repositório PyPI. Eventualmente poderemos atualizar os módulos do BioTEX, então os recursos disponíveis no pacote poderão ser modificados. Alterações no BioTEX que venhamos a fazer, incluindo todos os padrões de atributos de entrada citados no decorrer desta dissertação, serão descritas na página do BioTEX no *site* PyPI (<https://pypi.org/project/biotex/>).

O primeiro recurso que desenvolvemos para o BioTEX foi o AMINOcode, um algoritmo para codificação de texto em um formato baseado na representação de aminoácidos. Além do AMINOcode, desenvolvemos para o BioTEX o DNAbits, que também é um algoritmo para codificação de textos, mas para formato baseado na representação de nucleotídeos.

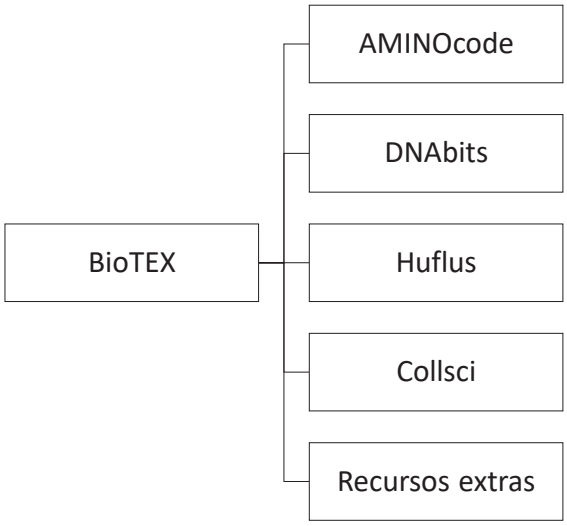
Também incluímos no BioTEX o Huflus (*Hierarquical clUstering - Fuzzy clUStering*), um clusterizador que faz uso de modelo hierárquico (descrito no tópico

2.4.2.1) e tomada de decisão a partir de critério difuso, conforme utilizado no trabalho de Guizelini et al. (2016).

Outro módulo em destaque no BioTEX é o Collsci, o núcleo de mineração de textos, que engloba as funcionalidades dos algoritmos citados anteriormente para utilização prática. O nome “Collsci” é inspirado na expressão “COLLective conSCIence” (consciência coletiva), mas também faz referência ao termo “COLLective SCIence” (ciência coletiva). Nosso objetivo com esse nome foi fazer uma alusão ao conhecimento humano coletivo.

A FIGURA 4 ilustra os componentes do BioTEX.

FIGURA 4 - COMPONENTES DO BIOTEX



FONTE: O autor (2020).

3.3 ESTUDO DE CASO

Além de desenvolvermos o BioTEX para auxiliar na execução de mineração de textos com ferramentas de Bioinformática, também executamos um estudo de caso, demonstrando a utilização dos recursos desenvolvidos.

Demonstramos a aplicação da biblioteca desenvolvida com a análise de um conjunto de títulos de artigos, que foram obtidos a partir de uma busca no PubMed.

4 APRESENTAÇÃO DOS RESULTADOS

Neste tópico serão abordadas as explicações sobre os recursos implementados para atender a proposta do trabalho, além da apresentação de um estudo de caso para exemplificar o uso.

4.1.1 AMINOcode

O AMINOcode é um algoritmo desenvolvido para codificação de textos escritos em linguagem natural para um formato baseado na representação de aminoácidos em arquivos FASTA.

A codificação do AMINOcode consiste na substituição de todos os caracteres do alfabeto romano, além de alguns caracteres adicionais utilizados na linguagem natural, apenas por caracteres que são utilizados na representação de sequências de aminoácidos em formato FASTA, ou seja, 20 caracteres.

Definimos uma codificação mínima e alguns detalhamentos opcionais para o AMINOcode. Na codificação mínima, que é a opção padrão, a sequência resultante do processo tem o menor tamanho possível, que não são recuperáveis na decodificação. Os detalhamentos fazem com que o tamanho da sequência aumente, mas mantenha a informação de alguns caracteres adicionais. O QUADRO 2 esquematiza o dicionário de substituição que padronizamos.

O QUADRO 3 exibe um exemplo de codificação com as versões mínimas e expandida do AMINOcode, além das respectivas decodificações. A frase utilizada é o título de um artigo utilizado no estudo de caso descrito no tópico 4.2. A frase foi codificada duas vezes, gerando duas sequências, uma com a versão mínima e outra com os detalhamentos, em seguida ambas foram decodificadas.

QUADRO 2 - DICIONÁRIO DO AMINOCODE

Caractere	Aminoácido na versão expandida	Aminoácido na versão mínima
a	YA	
b	E	
c	C	
d	D	
e	YE	
f	F	
g	G	
h	H	
i	YI	
j	I	
k	K	
l	L	
m	M	
n	N	
o	YQ	
p	P	
q	Q	
r	R	
s	S	
t	T	
u	YV	
v	V	
x	W	
z	A	
w	YW	
y	YY	
0	YDA	YD
1	YDQ	
2	YDT	
3	YDH	
4	YDF	
5	YDI	
6	YDS	
7	YDE	
8	YDG	
9	YDN	
.	YPE	YP
,	YPC	
;	YPS	
!	YPW	
?	YPQ	
:	YPT	
espaço	YS	
outros	YK	

FONTE: O autor (2020).

QUADRO 3 - DEMONSTRAÇÃO DO AMINOCODE

Texto original	**Global virus outbreaks: Interferons as 1st responders.**
Texto codificado (versão mínima)	GLYQEYALYSVYIRYVSYSYQYVTERYEYAKSYPYSYINTYERFYE RYQNSYSYASYSYDSTYSRYESPYQNDYERSYP
Texto codificado (versão expandida)	GLYQEYALYSVYIRYVSYSYQYVTERYEYAKSYPTYSYINTYERFYE RYQNSYSYASYSYDQSTYSRYESPYQNDYERSYPE
Texto decodificado (versão mínima)	global virus outbreaks. interferons as 9st responders.
Texto decodificado (versão expandida)	global virus outbreaks: interferons as 1st responders.

FONTE: O autor (2020).

Na FIGURA 5 apresentamos um alinhamento feito com textos codificados utilizando o AMINOcode. Utilizamos os títulos dos mesmos artigos obtidos para o estudo de caso tratado no tópico 4.2. Os títulos foram codificados com AMINOcode, vetorizados com SWeeP e clusterizados com Huflus (algoritmo explicado no tópico 4.1.3). A figura é composta pelos representantes de um dos *clusters*, alinhados através do programa Clustal Omega, descrito por Sievers et al. (2011). No alinhamento é possível identificar as palavras “*severe*”, “*acute*”, “*respiratory*” e “*syndrome*”, demonstrando que é possível reconhecer padrões coerentes a partir de um alinhamento de textos codificados. Se os textos de entrada fossem maiores, *abstracts* por exemplo, talvez fosse necessário particionar os textos por frases (ou outra divisão) para alinhamentos mais coerentes. Disponibilizamos o arquivo com o alinhamento utilizado na geração da FIGURA 5, além do arquivo FASTA com os títulos codificados, no endereço: <https://drive.google.com/drive/folders/1DTbuGZDsTCRF-iP8GHH7Wof6Gek_3c5a>.

FIGURA 5 – DEMONSTRAÇÃO DE ALINHAMENTO DE TEXTOS CODIFICADOS

FONTE: O autor (2020).

LEGENDA: Recorte de alinhamento de títulos de artigos codificados com o AMINOcode. A última linha, destacada com contorno vermelho, representa a decodificação do consenso para um trecho do alinhamento. A figura foi gerada utilizando o programa Jalview (disponível em <https://www.jalview.org/>).

4.1.2 DNAbits

O DNAbits é baseado na codificação dos textos primeiro para formato de bytes, que em seguida é convertido para a representação de nucleotídeos.

A primeira etapa é a codificação de *string* para bits seguindo a ASCII (do inglês *American Standard Code for Information Interchange*, que significa Código Padrão Americano para o Intercâmbio de Informação), portanto cada caractere é substituído por um byte (8 bits). Após a obtenção dos bytes em sequência, os bits são dois a dois substituídos pelas letras utilizadas na representação de nucleotídeos, seguindo a regra que definimos e esquematizamos no QUADRO 4.

QUADRO 4 - DICIONÁRIO DO DNABITS

bits	**nucleotídeo**
00	A
10	C
01	G
11	T

FONTE: O autor (2020).

A FIGURA 6 esquematiza as etapas do processo de codificação do DNAbits, utilizando a palavra “Texto” como exemplo.

FIGURA 6 - ESQUEMATIZAÇÃO DA CODIFICAÇÃO DO DNABITS

FONTE: O autor (2020).

A codificação do método DNAbits pode ser decodificada mantendo todos os caracteres, mas o tamanho da sequência gerada por ele é maior se comparada ao AMINOcode.

O QUADRO 5 exibe um exemplo de codificação e decodificação com o DNAbits. A frase utilizada é o título de um artigo utilizado no estudo de caso descrito no tópico 4.2, o mesmo utilizada no QUADRO 3.

QUADRO 5 - DEMONSTRAÇÃO DO DNABITS

Texto original	**Global virus outbreaks: Interferons as 1st responders.**
Texto codificado	TCACATGCTTGCGAGCCAGCATGCAAGAGCTCCGGCGATCCCTCTATCAAGA TTGCCCTCACTCGAGCGATCCCGCCAGCTGGCTATCGGTAAAGACGACGTG CACTCCCGCGATCGCGCCCGCGATCTTGCGTGCTATCAAGACAGCTATCAA GACATATATCACTCAAGAGATCCCGCTATCAATCTTGCGTGCACGCCCGCGA TCTATCGTGA
Texto decodificado	Global virus outbreaks: Interferons as 1st responders.

FONTE: O autor (2020).

4.1.3 Huflus

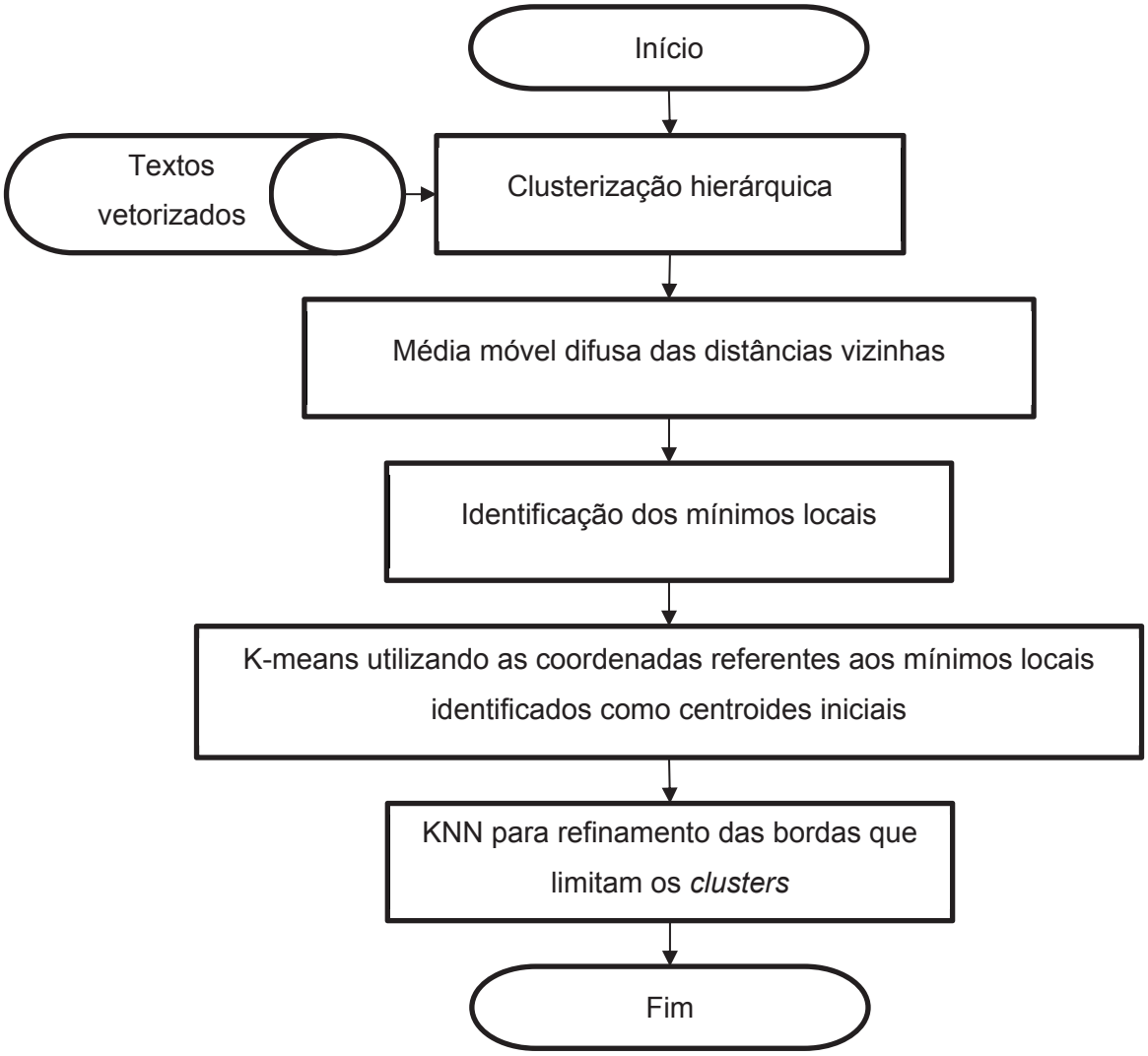
O algoritmo do Huflus (**H**ierarquical cl**U**stering - **F**uzzy c**LUS**tering) pode ser dividido em duas partes:

a) Predição dos centroides;

b) Refinamento das bordas que limitam os *clusters*.

Nos subtópicos seguintes as partes são tratadas com mais detalhes, mas a FIGURA 7 resume em forma de fluxograma os passos do Huflus por completo.

FIGURA 7 - ALGORITMO HUFLUS

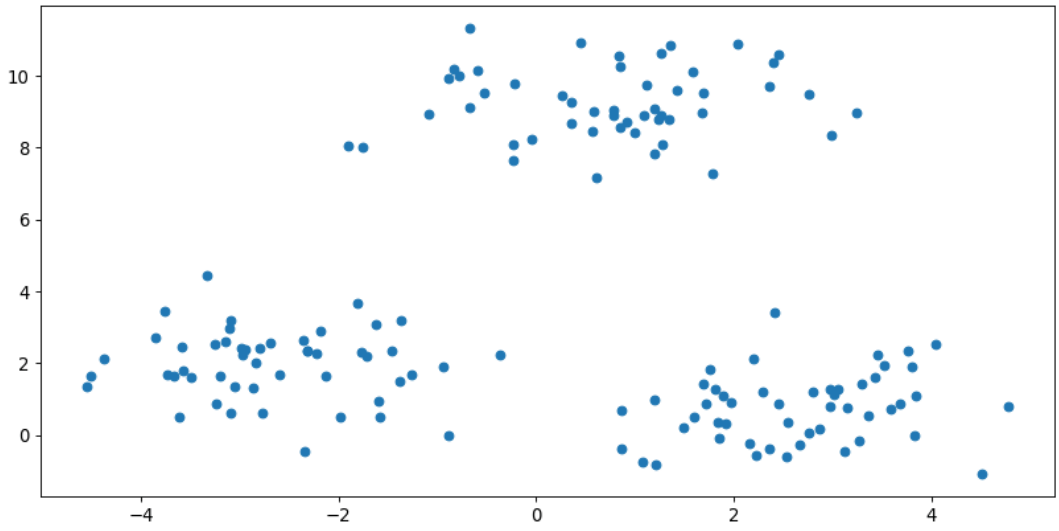


FONTE: O autor (2020).

4.1.3.1 Predição dos centroides

Para descrever o algoritmo optamos por utilizar uma abordagem explicativa através de exemplo. O GRÁFICO 1 representa as 150 observações que foram utilizados no passo-a-passo, as quais foram geradas de forma semialeatória, para que fossem conformados três grupos visualmente distinguíveis.

GRÁFICO 1 - EXEMPLO DE OBSERVAÇÕES BIDIMENSIONAIS



FONTE: O autor (2020).

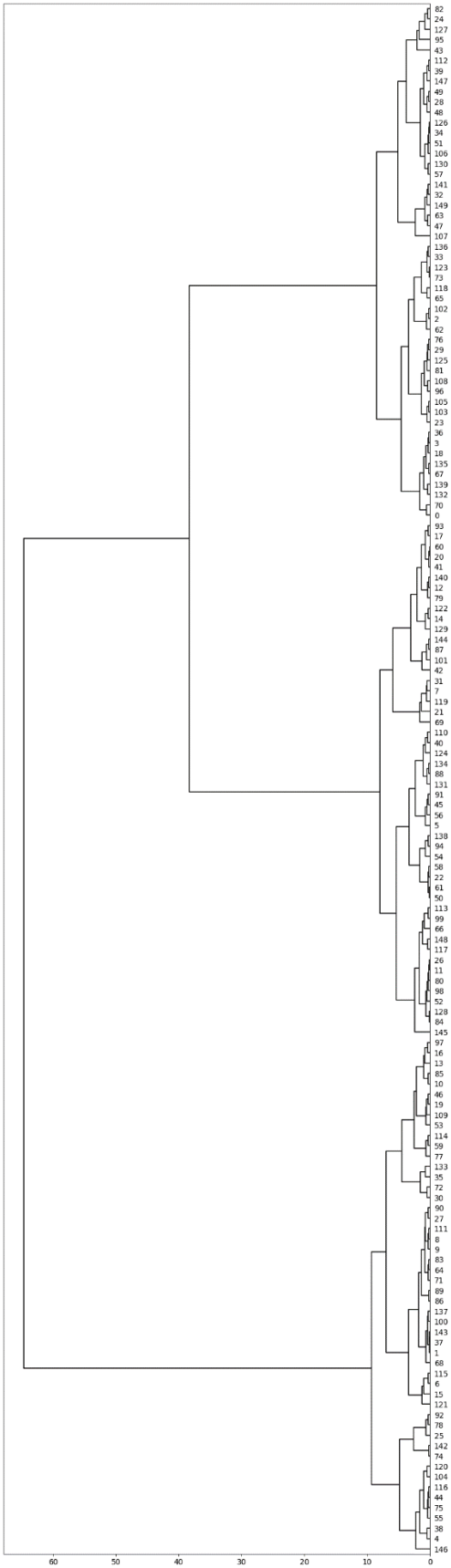
LEGENDA: Plotagem das 150 observações bidimensionais utilizadas para exemplificar o funcionamento da predição dos centroides do Huflus.

A predição dos centroides começa com a execução de uma clusterização hierárquica (descrita no tópico 2.4.2.1), seguida pela determinação dos pares de observações vizinhas. O GRÁFICO 2 esquematiza o resultado da clusterização para o exemplo em forma de dendrograma.

O passo seguinte é a definição da lista de distâncias euclidianas entre as observações vizinhas. No caso do exemplo, conforme pode ser verificado no GRÁFICO 2, as cinco primeiras distâncias são entre os seguintes pares de observações:

a) 82 e 24;
b) 24 e 127;
c) 127 e 95;
d) 95 e 43;
e) 43 e 112.

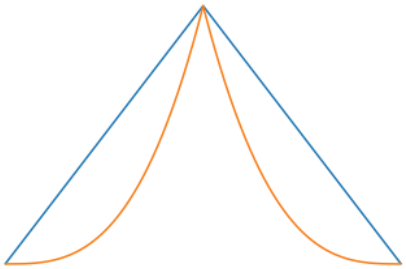
GRÁFICO 2 - RESULTADO DA CLUSTERIZAÇÃO HIERARQUICA



FONTE: O autor (2020).

Com a lista de distâncias é executada uma suavização através de uma média difusa, conforme feito também no trabalho de Guizelini et al. (2016). Em outras palavras, é executada uma média ponderada de cada elemento da lista com seus vizinhos, sendo o peso reduzido de acordo com o distanciamento do elemento central. Como padrão definimos a utilização de dois vizinhos de cada lado para a média ponderada, além de um *hedge* de concentração igual a 3. A aplicação do *hedge* de concentração consiste na potenciação de pesos lineares com um expoente específico, o que faz os pesos mais internos serem evidenciados em relação aos mais externos. Graficamente, com o *hedge* de concentração os pesos passam a apresentar uma conformação côncava, conforme demonstrado na A FIGURA 8.

FIGURA 8 - HEDGE DE CONCENTRAÇÃO



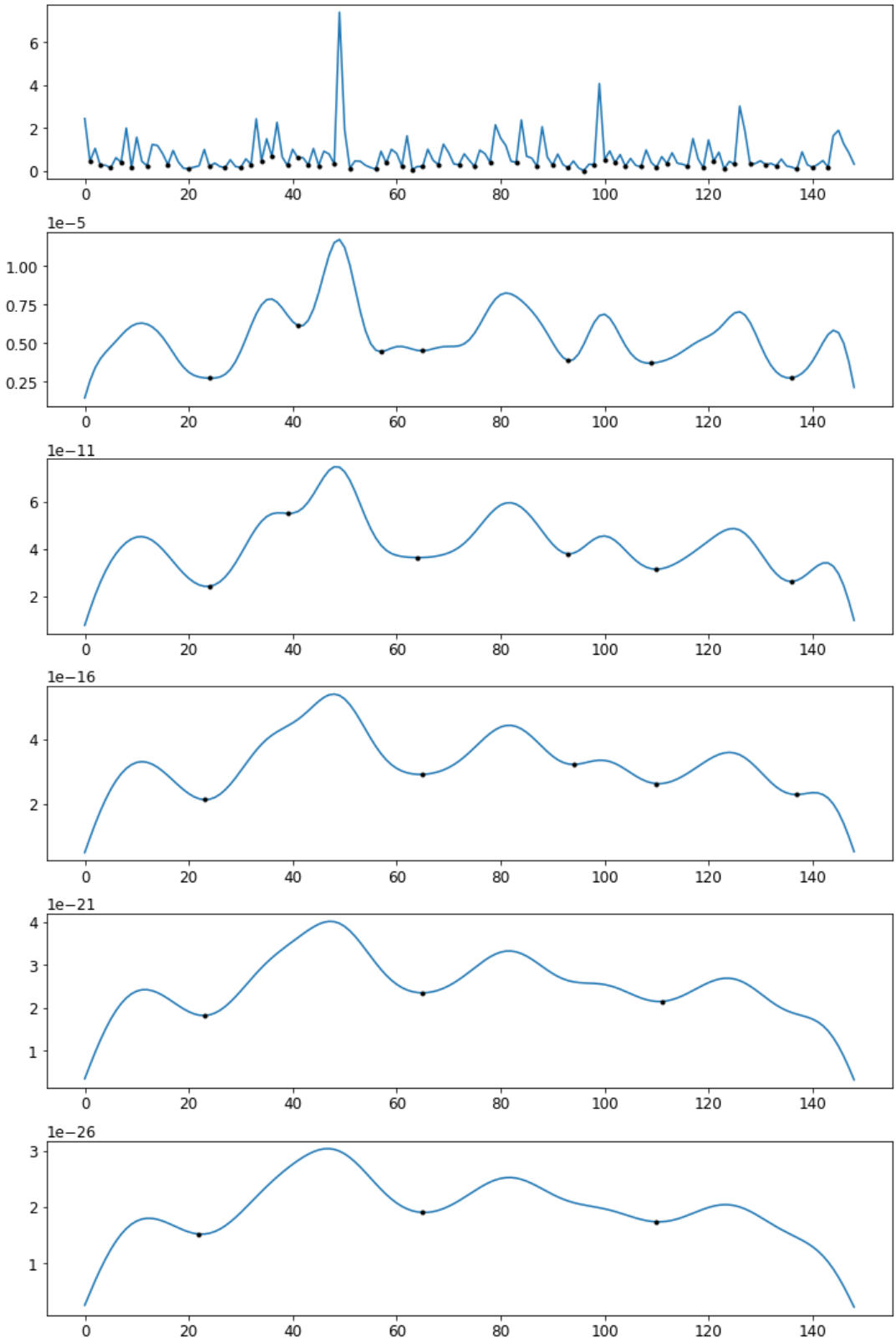
FONTE: O autor (2020).

LEGENDA: As linhas em azul e em laranja representam respectivamente os pesos sem e com hedge de concentração.

O processo de suavização é executado em *loop* (100 por padrão). No GRÁFICO 3 demonstramos o comportamento das distâncias no decorrer do processo de suavização.

No GRÁFICO 3 também são destacados os mínimos locais, que são alterados conforme decorre a suavização. Para seguimento do processo, é utilizada a lista de distâncias após todos os *loops* de suavização. Os mínimos locais indicam uma inversão na tendência das distâncias, portanto definimos a identificação e utilização das coordenas que geraram essas inversões como centroides iniciais para os clusters. Em consequência também é predito o número de clusters, que é igual ao número de centroides, que por sua vez é igual ao número de mínimos locais. No exemplo, três clusters foram encontrados.

GRÁFICO 3 - ILUTRAÇÃO DO PROCESSO DE SUAVIZAÇÃO FUZZY

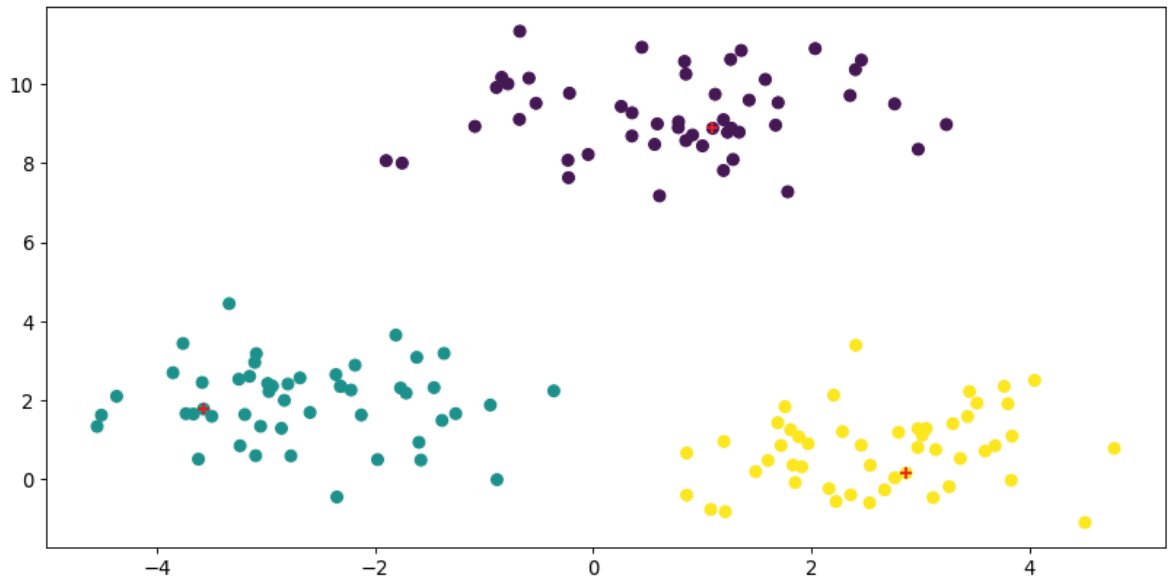


FONTE: O autor (2020).

LEGENDA: A linha no topo ilustra um vetor de distâncias original, enquanto a linhas abaixo representam o decorrer de quatro processos de suavização, de 20 em 20. Os pontos pretos destacam os mínimos locais.

Por fim é executado um K-means tradicional (descrito no tópico 2.4.2.2), utilizando as coordenadas identificadas pelo processo anterior como centroides iniciais, ao invés de coordenadas aleatórias. O GRÁFICO 4 representa o resultado para o exemplo apresentado no GRÁFICO 1, onde as diferentes cores representam os diferentes *clusters* determinados pelo método.

GRÁFICO 4 - RESULTADO DA CLUSTERIZAÇÃO PARA O EXEMPLO



FONTE: O autor (2020).

LEGENDA: As diferentes cores nos pontos representam diferentes *clusters*. As cruzes vermelhas representam os centroides.

4.1.3.2 Refinamento das bordas

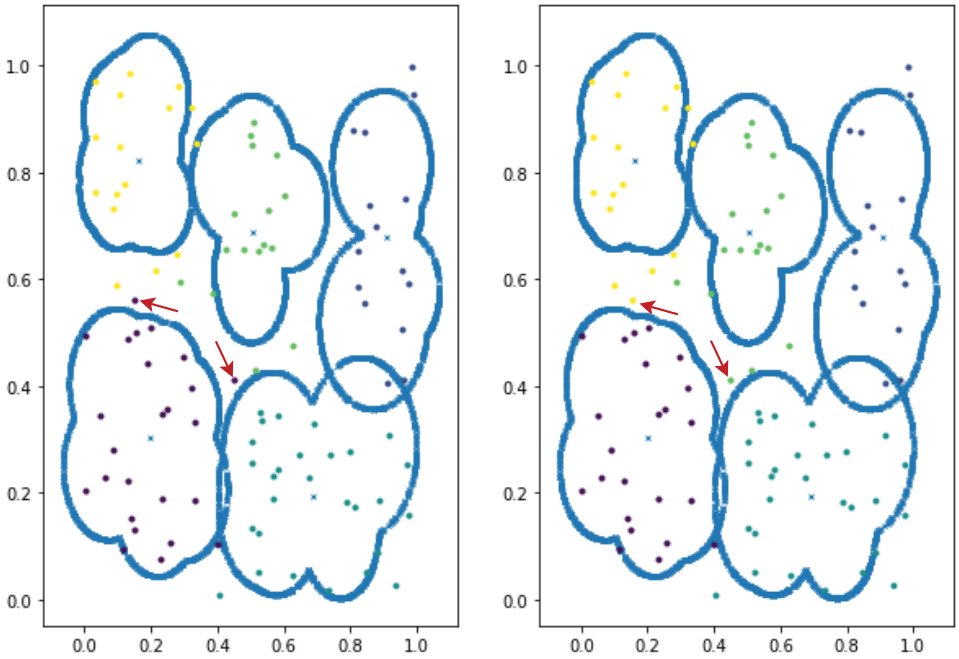
Após a predição dos centroides é executado o refinamento das bordas, ou seja, das coordenadas que limitam o espaço referente a cada *cluster*. O refinamento é feito considerando a vizinhança das observações. Esse processo é executado utilizando o algoritmo KNN (descrito no tópico 2.4.1.3).

Um conjunto de versores gerados aleatoriamente é projetado nos atributos de cada *cluster* determinado pelo K-means. As coordenadas dos versores projetados e os centroides são utilizados para o treinamento do KNN, com as classes sendo os *clusters* correspondentes. O número de versores aleatórios gerados por padrão é 1000, assim como o número de vizinhos utilizados na classificação também é 1000.

Após o treinamento do KNN, cada observações é reclassificada. O GRÁFICO 5 ilustra os versores projetados e a reclassificação das observações. O exemplo na

figura é referente ao processo para observações com duas dimensões, para facilidader a exibição gráfica, mas matematicamente o processo também pode ser feito para um maior número de dimensões, visto que também é possível obter a distâncias entre coordenadas com mais que duas dimensões.

GRÁFICO 5 - RECLASSIFICAÇÃO DE *CLUSTERS* DO HFCLUS



FONTE: O autor (2020).

LEGENDA: Os contornos em azul indicam os versores projetados. Os pontos indicam as observações, cada cor representando um *cluster*. As flechas vermelhas indicam as observações que apresentaram alteração na classificação.

4.1.4 Collsci

O Collsci é a integração dos outros algoritmos citados anteriormente, para utilização em abordagens de mineração de textos. Foi programado através de abordagem orientada a objeto, sendo bastante versátil. A classe principal do Collsci é chamada “Brain”, que pode instanciar um objeto cuja chamada requer um corpus como entrada, que pode ser manipulado para diferentes estratégias. Os principais métodos que desenvolvemos e disponibilizamos até o momento da apresentação do presente documento são:

a) **corpus2fasta**: executa a codificação do corpus para um formato baseado na representação de sequências biológica;

b) **fasta2vect**: executa o SWeeP para vetorizar as sequências obtidas pelo métodos anterior;
c) **vect2clus**: executa a clusterização a partir da matriz gerado pelo SWeeP. O método de clusterização padrão é o HFclus;
d) **corpus2ppcorpus**: executa o pré-processamento dos textos. Os pré-processamentos utilizados por padrão são: tokenização (explicação no tópico 2.2.1), remoção de pontuações, remoção de palavras com 2 ou menos caracteres, remoção de palavras com 20 ou mais caracteres, remoção de caracteres não alfanuméricos e remoção de *stop words* (explicação no tópico 2.2.2). Esses pré-processamentos apenas foram aplicados para utilização do método seguinte, eles não são utilizados por padrão em nenhum dos métodos citados anteriormente;
e) **word2vect**: vetoriza as palavras considerando a representatividade delas dentro dos *clusters* de textos resultantes de uma determinada clusterização. Cada *cluster* é tratado como uma unidade de texto (documento), gerando um corpus com uma nova organização, que é utilizado para criação de uma matriz TF-IDF (explicação no tópico 2.3.3), onde cada linha representa uma palavra;
f) **wordvect2tree**: a matriz gerada pelo método "word2vect" é utilizada para executar uma clusterização hierarquica das palavras, cujo resultado é transformado em formato de dendrograma ("árvore de palavras" ou "*word tree*"). As palavras são colocadas em um ranque, para possibilitar a filtragem de um número específico de palavras na árvore. As dez palavras mais significativas ("*top words*") são colocadas em caixa alta e o número no *rank* é exibido na direita de cada palavra. A métrica para o a ranqueamento é a ordenação da norma dos vetores que representam as palavras, normalizadas entre o intervalo de 0 a 1.

4.1.5 Recursos adicionais

Incluímos no BioTEX também algumas funções facilitadoras para eventuais trabalhos, entre elas:

a) **biotex.searchtools.pubsearch**: recebe uma *string* de busca e retorna os resultados a partir do PubMed;

b) **biotex.fastatools.clustalo**: utiliza uma instalação local do programa de alinhamento de sequências Clustal Omega, descrito por Sievers et al. (2011), para executar o alinhamento múltiplo dentro do Python;
c) **biotex.collsci.tree2newick**: converte uma variável do tipo “scipy.cluster.hierarchy.ClusterNode” da biblioteca SciPy para uma *string* no formato newick, que pode ser utilizado em uma série de programas de visualização de dendrogramas;
d) **biotex.fastatools.dendroscope**: recebe uma *string* com conteúdo no formato newick, cria um arquivo temporário e abre o arquivo utilizando uma instalação do Dendroscope 3. O Desdroscope 3 é um *software* para visualização de dendrogramas, descrito por Huson e Scornavacca (2012);
e) **biotex.fastatools.getcons**: a partir de um conjunto de sequências de entrada executa a obtenção da sequência consenso;
f) **biotex.fastatools.fasta2vect**: implementação em python do SWeeP, cuja distribuição original é descrita por De Pierri et al. (2020).

4.2 ESTUDO DE CASO

4.2.1 Descrição do estudo de caso

Decorremos o estudo de caso com um exemplo envolvendo o coronavírus, devido sua relevância no período em que o presente trabalho foi desenvolvido.

Realizamos no dia 12 de abril de 2020, utilizando o biopython, uma pesquisa no PubMed com a seguinte *string* de busca: “sars-cov[Title/Abstract] and zoonotic[Title/Abstract] AND ("0001/1/1"[Date - Publication] : "2019/10/31"[Date - Publication])”. Nosso objetivo foi identificar indícios da COVID-19 (do inglês *Coronavirus Disease 2019*, que significa Doença do Coronavírus 2019) antes da ascensão da doença.

Segundo Shereen et al. (2020), o SARS-CoV-2 (do inglês *severe acute respiratory syndrome coronavirus 2*, que significa síndrome respiratória aguda grave de coronavírus 2), vírus causador da COVID-19, se originou em um mercado de frutos do mar na cidade chinesa Wuhan, no qual também eram comercializados morcegos, cobras, cães-guaxinim, civetas de palmeiras e outros animais. A doença foi

identificada em dezembro de 2019. Ainda não é confirmado, mas há indícios de que o SARS-CoV-2 surgiu a partir de transmissão zoonótica vinda de morcegos.

Utilizamos a palavra "*zoonotic*" na *string* de busca para identificação de indícios de evolução zoonótica da doença. Também incluímos a limitação de data de publicação até o final de outubro de 2019, que é antes do período que a COVID-19 foi identificada na China (Shereen et al., 2020), com uma margem de segurança como medida de garantia para que os artigos retornados fossem de fato anteriores aos acontecimentos.

4.2.2 Resultado do estudo de caso

Através da busca, 205 resultados foram retornados, dos quais extraímos os *abstracts* e aplicamos o Collsci do BioTEX, executando os algoritmos AMINOcode, SWeeP e Huflus, todos com os parâmetros padrões.

Cinco *clusters* foram preditos pelo Huflus, mas um deles é composto apenas por um elemento vazio, referente a um artigo sem *abstract*, portanto os *clusters* com conteúdo são quatro. A TABELA 1 indica o número de elementos dentro de cada *cluster*. Disponibilizamos os dados de todos os artigos utilizados no processo, bem como o número de *cluster* indexado para cada um deles, no seguinte endereço: <https://docs.google.com/spreadsheets/d/1u4vXhvmoM2RnTQjyi0DJR_mC2nC1tDg07xdIHf_SM9Q>.

TABELA 1 - *CLUSTERS* PREDITOS PELO HUFLUS

cluster	**Número de elementos**
0	1
1	44
2	1
3	60
4	99

FONTE: O autor (2020).

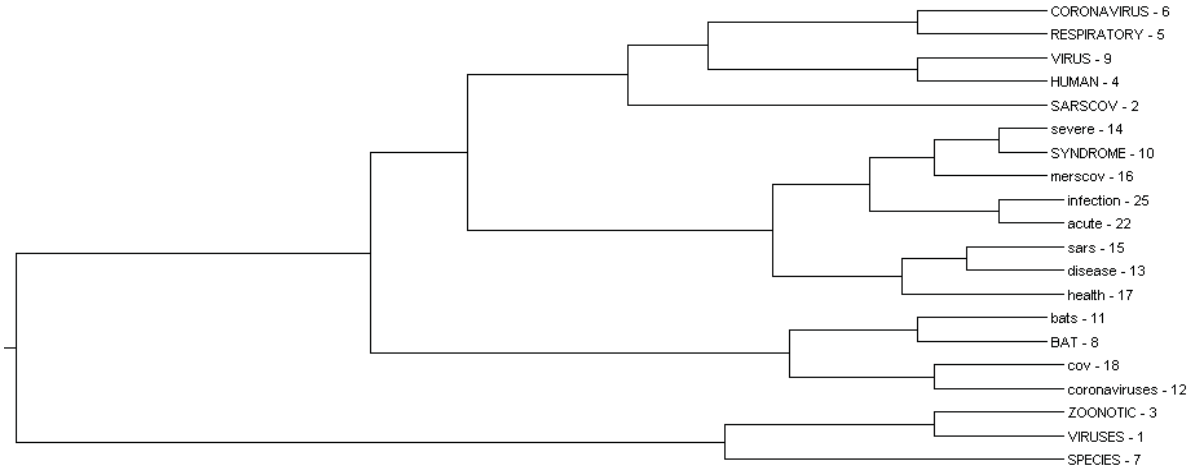
Executamos também a construção da árvore de palavras. Definimos duzentas palavras na construção da árvore, o que fez com que fosse inviável inserir uma imagem inteira dela em uma página de tamanho A4, portanto selecionamos partes para decorrer discussões, no entanto disponibilizamos a árvore completa para

download no endereço <https://drive.google.com/drive/folders/1uoBi6h7Rd0ltPI8-kLCmirswskf4Avo3>.

O GRÁFICO 6 exibe o ramo que descreve a doença, com palavras como: “coronavirus” (“coronavírus”), “syndrome” (“síndrome”), “vírus” (“vírus”), “repiratory”, (respiratório), “infection” (“infecção”) etc. Vale notar que as 10 palavras mais significativas aparecem nesse ramo, indicando que elas são muito relevantes para os artigos em um contexto geral. As palavras “human” (“humano”) e “species” (“espécies”) apareceram próximas, provavelmente sugerindo indícios de evolução da doença por meio de animais que não humanos. Também aparecem em destaque as palavras “bat” (“morcego”) e “bats” (“morcegos”), indicando que mesmo antes da COVID-19, já havia destaque para os estudos com morcegos como hospedeiros do coronavírus.

Para comprovar a evidência apontada, buscamos em meios aos artigos utilizados um que colaborasse tais afirmações. Identificamos que Fan et al. (2019), em uma publicação em março de 2019, apontaram uma grande possibilidade para o surgimento de um novo coronavírus tipo SARS ou MERS na China. Smith e Wang (2013) também evidenciaram o risco de vírus zoonóticos surgirem de morcegos, bem como a importância de estudos em relação à dinâmica de infecção nos animais como medida preventiva.

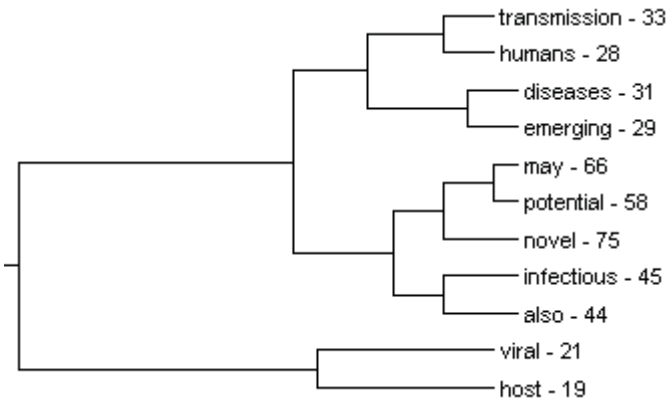
GRÁFICO 6 - RAMO DA ÁRVORE DE PALAVRAS (DEFINIÇÃO DA DOENÇA)



FONTE: O autor (2020).

O GRÁFICO 7 exibe outro ramo com muitos termos com relevância alta. Nele é possível notar indícios de uma infecção viral em potencial em humanos.

GRÁFICO 7 - RAMO DA ÁRVORE DE PALAVRAS (POTENCIAL INFECCIOSO)

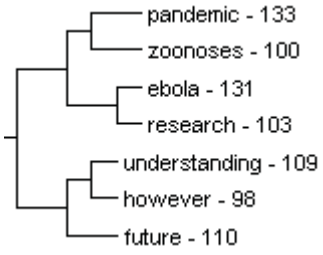


FONTE: O autor (2020).

Mais um ramo interessante é apresentado no GRÁFICO 8, que coloca próximas as palavras "*pandemic*" ("pandemia"), "*zoonoses*" ("zoonoses") e "*future*" ("futuro"), sendo mais uma referência para eventos futuros associados ao SARS. Também existe uma associação com a doença ebola, provavelmente por também ser uma zoonose.

Novamente buscamos nos artigos um que comprovasse a hipótese, identificamos que Smith e Wang (2013) citam o coronavírus e o ebola como exemplos de famílias virais que são associadas a morcegos.

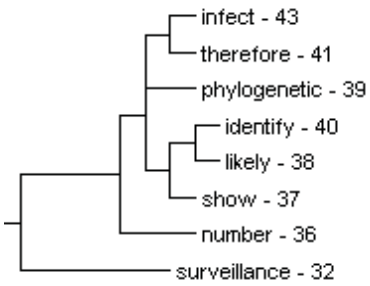
GRÁFICO 8 - RAMO DA ÁRVORE DE PALAVRAS (FUTURO)



FONTE: O autor (2020).

No ramo apresentado no GRÁFICO 9 é possível identificar palavras que indicam apresentação de resultados de análise, como "*show*" ("apresentação"), "*identify*" ("identificar"), e "*therefore*" (portanto). Tais palavras aparecem junto com "*phylogenetic*" ("filogenética"), indicando que o método de muitos dos artigos é análise filogenética.

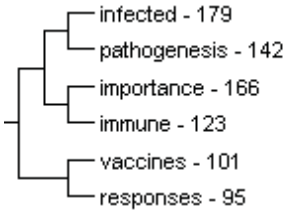
GRÁFICO 9 - RAMO DA ÁRVORE DE PALAVRAS (APRESENTAÇÃO)



FONTE: O autor (2020).

No GRÁFICO 10 é possível identificar estudos referentes a vacinas, com a palavra "*vacines*" ("vacinas") aparecendo junto com "*immune*" ("imune") e "*responses*" ("respostas"). Também aparece a palavra "*importance*" (importância), sugerindo provavelmente a importância do desenvolvimento da vacina.

GRÁFICO 10 - RAMO DA ÁRVORE DE PALAVRAS (VACINAS)



FONTE: O autor (2020).

Outras observações podem ser feitas em outros pontos da árvore, em alguns casos são necessárias buscas adicionais para compreender o motivo de determinadas palavras estarem próximas, no entanto o que apresentamos é suficiente para inferir uma resposta em relação ao proposito inicial do estudo de caso: de fato haviam indícios para o surgimento da COVID-19 na literatura antes do surgimento da doença em efetivo, sendo previsível inclusive a origem filogenética do vírus. Os trabalhos de Fan et al. (2019) e Smith e Wang (2013) são exemplos disso.

5 CONCLUSÃO

Desenvolvemos o pacote BioTEX, uma ferramenta que funciona como um intermediário para possibilitar a intersecção entre mineração de textos e bioinformática.

Demonstramos que é possível executar a codificação de textos escritos em linguagem natural para um formato baseado na representação de sequências biológicas. Com os textos codificados, evidenciamos que é possível executar o alinhamento e a vetorização utilizando ferramentas que foram idealizadas para o uso em bioinformática.

Exemplificamos a representação do resultado de mineração de textos no formato comumente utilizado para análise de filogenética de forma coerente, evidenciado mais uma vez a aplicação de um recurso comum na bioinformática.

Através do estudo de caso, o nosso método foi capaz de identificar padrões em um conjunto de textos, indicando evidências de um evento futuro a partir do conteúdo textual.

O BioTEX é uma ferramenta de mineração de textos eficaz, que pode ser facilmente compreendida por cientistas de dados.

5.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Destacamos a importância de futuros experimentos com o BioTEX em diferentes áreas do conhecimento, para comprovar a versatilidade da estratégia. Também é válida a exploração do BioTEX na resolução de problemas comuns na mineração de textos, por exemplo:

a) geração de resumos automaticamente;
b) escrita automática de artigos;
c) geração de explicações sobre um conjunto de arquivos FASTA de entrada através da análise das *headers*;
d) simulação de expressões humanas através de textos, para uso em mecanismos de diálogo automático;
e) construção automática de protocolos de procedimento laboratoriais;
f) automação na resposta de perguntas.

Outro possível trabalhos futuro é o desenvolvimento de *softwares* com interface gráfica e aplicações *web* que utilizem os recursos do BioTEX, com objetivo de democratizar o uso para usuários em geral.

REFERÊNCIAS

AICH, S.; SAIN, M.; PARK, J.; CHOI, K.-W.; KIM, H.-C. A text mining approach to identify the relationship between gait-Parkinson's disease (PD) from PD based research articles. 2017 International Conference on Inventive Computing and Informatics (ICICI). **Anais...** . p.481–485, 2017. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/8365398/>.

ALKALBANI, A. M.; GHAMRY, A. M.; HUSSAIN, F. K.; HUSSAIN, O. K. Sentiment analysis and classification for software as a service reviews. **Proceedings - International Conference on Advanced Information Networking and Applications, AINA**, v. 2016-May, p. 53–58, 2016.

ANEES, A. F.; SHAIKH, ARSALAAN; SHAIKH, ARBAZ; SHAIKH, S. Survey Paper on Sentiment Analysis: Techniques and Challenges. **EasyChair Preprint**, n. 2389, 2020.

ANTONY, S.; WAGH, R. Study on Text Clustering For Topic Identification. **International Journal of Advanced Research in Computer Science**, v. 8, n. 1, 2017. Disponível em: <http://www.ijarcs.info/index.php/Ijarcs/article/view/2874>.

AYYILDIZ, D.; PIAZZA, S. **Introduction to Bioinformatics**. 2019.

BAHARUDIN, B.; LEE, L. H.; KHAN, K. A Review of Machine Learning Algorithms for Text-Documents Classification. **Journal of Advances in Information Technology**, v. 1, n. 1, p. 4–20, 2010.

BANIK, N.; HAFIZUR RAHMAN, M. H.; CHAKRABORTY, S.; SEDDIQUI, H.; AZIM, M. A. Survey on Text-Based Sentiment Analysis of Bengali Language. 1st International Conference on Advances in Science, Engineering and Robotics Technology. **Anais...** . p.1–6, 2019. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/8934481/>.

BERRY, M. W.; CASTELLANOS, M. **Survey of Text Mining : Clustering , Classification , and Retrieval , Second Edition**. 2007.

BHALLA, R. Document Level Classification Using Twitter Sentiments. **Journal of the Gujarat Research Society**, v. 21, n. 6, p. 797–802, 2019.

COCK, P. J. A.; ANTAO, T.; CHANG, J. T.; et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. **Bioinformatics**, v. 25, n. 11, p. 1422–1423, 2009. Disponível em: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btp163>.

EDDY, S. R. What is a hidden Markov model? **Nature Biotechnology**, out. 2004. Nature Publishing Group. Disponível em: <https://www.nature.com/articles/nbt1004-1315>.

FAN, Y.; ZHAO, K.; SHI, Z.-L.; ZHOU, P. Bat Coronaviruses in China. **Viruses**, v. 11, n. 3, p. 210, 2019. MDPI AG. Disponível em: <https://www.mdpi.com/1999-4915/11/3/210>.

FANTOZZI, P.; LAURA, L.; NANNI, U. Weighted-Distance Sliding Windows and Cooccurrence Graphs for supporting Entity-Relationship Discovery in Unstructured Text. **International Journal of Computer, Electrical, Automation, Control and Information Engineering**, v. 12, n. 8, p. 663–669, 2018. Disponível em: <http://scholar.waset.org/1307-6892/10009445>.

GHOSH, S.; ROY, S.; BANDYOPADHYAY, S. K. A tutorial review on Text Mining Algorithms. **International Journal of Advanced Research in Computer and Communication Engineering**, v. 1, n. 4, p. 223–233, 2012.

GOLDBERG, X. **Introduction to semi-supervised learning**. 2009.

GUIZELINI, D.; RAITTZ, R. T.; CRUZ, L. M.; et al. GFinisher: a new strategy to refine and finish bacterial genome assemblies. **Scientific Reports**, v. 6, n. 1, p. 34963, 2016. Nature Publishing Group. Disponível em: <http://www.nature.com/articles/srep34963>.

HATZIGEORGIU, N.; MIKROS, G.; CARAYANNIS, G. Word length, word frequencies and Zipf's law in the Greek language. **Journal of Quantitative Linguistics**, v. 8, n. 3, p. 175–185, 2001.

HUNTER, J. D. Matplotlib: A 2D Graphics Environment. **Computing in Science & Engineering**, v. 9, n. 3, p. 90–95, 2007. Disponível em: <http://ieeexplore.ieee.org/document/4160265/>.

HUSON, D. H.; SCORNAVACCA, C. Dendroscope 3: An Interactive Tool for Rooted Phylogenetic Trees and Networks. **Systematic Biology**, v. 61, n. 6, p. 1061–1067, 2012. Disponível em: <https://academic.oup.com/sysbio/article/61/6/1061/1666897>.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering. **Data Clustering: A Review**, v. 31, n. 3, p. 264–323, 1999. Disponível em: <http://dl.acm.org/doi/10.1145/331499.331504>.

JONES, S. K. A Statistical Interpretation of Term Specificity and its Application in Retrieval. **Journal of Documentation**, v. 28, p. 11–21, 1972.

K. DALAL, M.; A. ZAVERI, M. Automatic Text Classification: A Technical Review. **International Journal of Computer Applications**, v. 28, n. 2, p. 37–40, 2011. Disponível em: <https://www.researchgate.net/publication/266296879>.

KHALEGHI, T.; MURAT, A.; ARSLANTURK, S.; DAVIES, E. Automated Surgical Term Clustering: A text mining approach for unstructured textual surgery descriptions. **IEEE Journal of Biomedical and Health Informatics**, v. 24, n. 7, p. 2107–2118, 2020. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/8918246/>.

LIMA, E.; SILVA, M. Comportamento bibliométrico da língua portuguesa, como veículo de representação da informação. **Ciência da Informação**, v. 2, n. 2, p. 99–138, 1973. Disponível em: <http://revista.ibict.br/ciinf/article/view/31>.

LUHN, H. P. The Automatic Creation of Literature Abstracts. **IBM Journal of Research and Development**, v. 2, n. 2, p. 159–165, 1958. Disponível em: <http://ieeexplore.ieee.org/document/5392672/>.

MANNING, C. D.; SCHIITZE, H. **Foundations of Statistical Natural Language Processing**. Cambridge: The MIT Press, 1999.

MCKINNEY, W. Data Structures for Statistical Computing in Python. **Proceedings of the 9th Python in Science Conference**, p. 56–61, 2010. Disponível em: <http://conference.scipy.org/proceedings/scipy2010/mckinney.html>.

MITSUMORI, T.; FATION, S.; MURATA, M.; DOI, K.; DOI, H. Gene/protein name recognition based on support vector machine using dictionary as features. **BMC Bioinformatics**, v. 6, n. S8, 2005. BioMed Central. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/15960842>.

OLIPHANT, T. E. **Guide to NumPy**. 2006.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; et al. **Scikit-learn: Machine Learning in Python**. 2011.

DE PIERRI, C. R.; VOYCEIK, R.; SANTOS DE MATTOS, L. G. C.; et al. SWeeP: representing large biological sequences datasets in compact vectors. **Scientific Reports**, v. 10, n. 1, p. 91, 2020. Nature Research. Disponível em: <http://www.nature.com/articles/s41598-019-55627-4>.

PORTER, M. F. An algorithm for suffix stripping. **Program**, v. 14, n. 3, p. 130–137, 1980. MCB UP Ltd. Disponível em: <http://www.emeraldinsight.com/doi/10.1108/eb046814>.

PORTER, M. F. Snowball: A language for stemming algorithms. 2001. Disponível em: <http://snowball.tartarus.org/texts/introduction.html>.

QI, Y.; ZHANG, Y.; SONG, M. Text mining for bioinformatics: State of the art review. Proceedings - 2009 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009. **Anais...** , 2009.

ROBERTSON, S. Understanding inverse document frequency: On theoretical arguments for IDF. **Journal of Documentation**, v. 60, n. 5, p. 503–520, 2004.

RUNESON, P.; ALEXANDERSSON, M.; NYHOLM, O. Detection of duplicate defect reports using natural language processing. **Proceedings - International Conference on Software Engineering**, p. 499–508, 2007.

SCHNEIDER, T. D.; STEPHENS, R. M. Sequence logos: A new way to display consensus sequences. **Nucleic Acids Research**, v. 18, n. 20, p. 6097–6100, 1990. Oxford Academic. Disponível em: <https://academic.oup.com/nar/article/18/20/6097/1141316>.

SHAFIE, A. S.; MOHTAR, I. A.; MASROM, S.; AHMAD, N. Backpropagation neural network with new improved error function and activation function for classification problem. **2012 IEEE Symposium on Humanities, Science and Engineering Research**, p. 1359–1364, 2012.

SHEREEN, M. A.; KHAN, S.; KAZMI, A.; BASHIR, N.; SIDDIQUE, R. COVID-19 infection: Origin, transmission, and characteristics of human coronaviruses. **Journal of Advanced Research**, 1. jul. 2020. Elsevier B.V.

SHTRIKMAN, S. Some comments on Zipf's law for the Chinese language. **Journal of Information Science**, v. 20, n. 2, p. 142–143, 1994.

SIEVERS, F.; HIGGINS, D. G. Clustal Omega for making accurate alignments of many protein sequences. **Protein Science**, v. 27, n. 1, p. 135–145, 2018. Blackwell Publishing Ltd. Disponível em: <https://onlinelibrary.wiley.com/doi/full/10.1002/pro.3290>.

SIEVERS, F.; WILM, A.; DINEEN, D.; et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. **Molecular Systems Biology**, v. 7, p. 539, 2011. Disponível em: <www.molecularsystemsbiology.com>.

SIMON, C.; DAVIDSEN, K.; HANSEN, C.; et al. BioReader: A text mining tool for performing classification of biomedical literature. **BMC Bioinformatics**, v. 19, 2019. BioMed Central Ltd.

SMITH, I.; WANG, L.-F. Bats and their virome: an important source of emerging viruses capable of infecting humans. **Current Opinion in Virology**, v. 3, n. 1, p. 84–91, 2013. Elsevier B.V. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1879625712001861>.

STEINBACH, M.; KARYPIS, M.; KUMAR, V. A comparison of document clustering techniques. **In KDD Workshop on Text Mining**, 2000.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. **Nature Methods**, v. 17, n. 3, p. 261–272, 2020. Nature Research. Disponível em: <http://www.nature.com/articles/s41592-019-0686-2>.

WILLETT, P. The Porter stemming algorithm: Then and now. **Program**, v. 40, n. 3, p. 219–223, 2006.

YAN, S.; WONG, K. C. Context awareness and embedding for biomedical event extraction. **Bioinformatics (Oxford, England)**, v. 36, n. 2, p. 637–643, 2020.

ZHANG, H.; LI, D. Naive Bayes Text Classifier. 2007 IEEE International Conference on Granular Computing (GRC 2007). **Anais...** . p. 708–708, 2007. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/4403192/>.

ZIPF, G. K. **Human behavior and the principle of least effort**. Addison-Wesley 1949.

ANEXO 1 – ALGORITMO PORTER STEMMER

O QUADRO 1 representa os passos do algoritmo “Porter Stemmer”, enquanto o QUADRO 2 exibe as regras de leitura das esquematizações no QUADRO 1. O conteúdo de ambos os quadros é uma reformatação do material apresentado por Porter (1980).

QUADRO 1 – REGRAS DE SUBSTITUIÇÃO DO ALGORITMO Porter Stemmer

Passo	Descrição	Substituição	Exemplo
1a	Tratar plurais	SSES → SS	caresses → caress
		IES → I	ponies → poni ties → tie
		SS → SS	caress → caress
		S →	cats → cat
1b	Tratar flexões “ED” e “ING”	(m>0) EED → EE	feed → feed agreed → agree
		(*v*) ED →	plastered → plaster bled → bled
		(*v*) ING →	motoring → motor sing → sing
	Nas mesmas palavras tratadas pelo passo anterior	AT → ATE	conflat(ed) → conflate
		BL → BLE	troubl(ing) → trouble
		IZ → IZE	siz(ed) → size
		(*d e não (*L ou *S ou *Z)) → letra única	hopp(ing) → hop tann(ed) → tan fall(ing) → fall hiss(ing) → hiss fizz(ed) → fizz
		(m=1 e *o) → E	fail(ing) → fail fil(ing) → file
1c	Tratar palavras finalizadas com a letra “Y”	(*v*) Y → I	happy → happi sky → sky
2	Tratar finalizações específicas	(m>0) ATIONAL → ATE	relational → relate
		(m>0) TIONAL → TION	conditional → condition rational → rational
		(m>0) ENCI → ENCE	valenci → valence
		(m>0) ANCI → ANCE	hesitanci → hesitance
		(m>0) IZER → IZE	digitizer → digitize
		(m>0) ABLI → ABLE	conformabli → conformable
		(m>0) ALLI → AL	radicalli → radical
		(m>0) ENTLI → ENT	differentli → different
		(m>0) ELI → E	vileli → vile
		(m>0) OUSLI → OUS	analogousli → analogous
		(m>0) IZATION → IZE	vietnamization → vietnamize
		(m>0) ATION → ATE	predication → predicate
		(m>0) ATOR → ATE	operator → operate
		(m>0) ALISM → AL (m>0) IVENESS → IVE	feudalism → feudal decisiveness → decisive
		(m>0) FULNESS → FUL	hopefulness → hopeful
		(m>0) OUSNESS → OUS	callousness → callous
		(m>0) ALITI → AL	formaliti → formal
		(m>0) IVITI → IVE	sensitiviti → sensitive

		(m>0) BILITI → BLE	sensibiliti → sensible
3		(m>0) ICATE → IC	triplicate → triplic
		(m>0) ATIVE →	formative → form
		(m>0) ALIZE → AL	formalize → formal
		(m>0) ICITI → IC	electriciti → electric
		(m>0) ICAL → IC	electrical → electric
		(m>0) FUL →	hopeful → hope
		(m>0) NESS →	goodness → good
4		(m>1) AL →	revival → reviv
		(m>1) ANCE →	allowance → allow
		(m>1) ENCE →	inference → infer
		(m>1) ER →	airliner → airlin
		(m>1) IC →	gyroscopic → gyroscop
		(m>1) ABLE →	adjustable → adjust
		(m>1) IBLE →	defensible → defens
		(m>1) ANT →	irritant → irrit
		(m>1) EMENT →	replacement → replac
		(m>1) MENT →	adjustment → adjust
		(m>1) ENT →	dependent → depend
		((m>1) e (*S ou *T)) ION →	adoption → adopt
		(m>1) OU →	homologou → homolog
		(m>1) ISM →	communism → commun
		(m>1) ATE →	activate → activ
		(m>1) ITI →	angulariti → angular
		(m>1) OUS →	homologous → homolog
		(m>1) IVE →	effective → effect
		(m>1) IZE →	bowdlerize → bowdler
5a		(m>1) E →	probate → probat rate → rate
		(m=1 e não *o) E →	cease → ceas
5b		(m>1 e *d e *L) → letra única	controll → control roll → roll

FONTE: Reformatação do conteúdo de Porter (1980).

QUADRO 2 – REGRAS DE LEITURA PARA O QUADRO 1

Representação	**Leitura**
Letras maiúsculas	A própria letra
*	Qualquer letra
c	Consoante
v	Vogal
m	Número de vezes que o padrão “vc” (vogal seguida por consoante) ocorrer, não necessariamente em sequência
v	Presença de vogal
*d	Final com duas consoantes
*o	Padrão “cvc”, com o segundo “c” não sendo “W”, “X” ou “Y”

FONTE: Reformatação do conteúdo de Porter (1980).