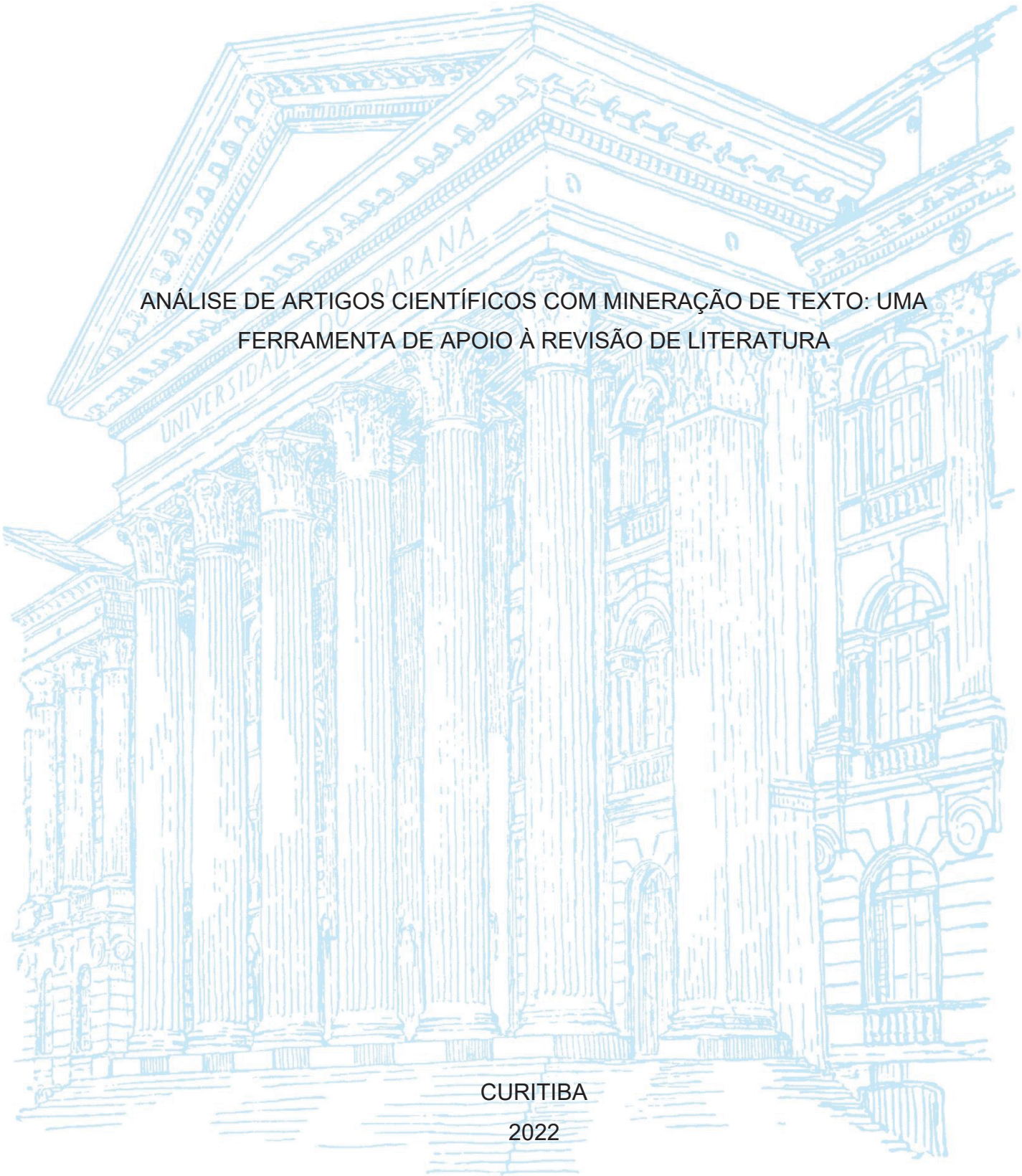
UNIVERSIDADE FEDERAL DO PARANÁ

CHRISTIANE SHOJI WAGNER



ANÁLISE DE ARTIGOS CIENTÍFICOS COM MINERAÇÃO DE TEXTO: UMA FERRAMENTA DE APOIO À REVISÃO DE LITERATURA

CURITIBA

2022

CHRISTIANE SHOJI WAGNER

ANÁLISE DE ARTIGOS CIENTÍFICOS COM MINERAÇÃO DE TEXTO: UMA FERRAMENTA DE APOIO À REVISÃO DE LITERATURA

Trabalho de Conclusão de Curso apresentado ao curso de Pós-Graduação em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador(a): Prof(a). Dr(a). Rafaela Mantovani Fontana

CURITIBA

2022

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **CHRISTIANE SHOJI WAGNER** intitulada: **Análise de Artigos Científicos com Mineração de Texto: Uma Ferramenta de Apoio à Revisão de Literatura** , que após terem inquirido a aluna e realizada a avaliação do trabalho , são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de especialista 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, 17 de Novembro de 2022.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

RAZER ANTHOM NIZER ROJAS MONTAÑO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

ANÁLISE DE ARTIGOS CIENTÍFICOS COM MINERAÇÃO DE TEXTO: UMA FERRAMENTA DE APOIO À REVISÃO DE LITERATURA

Christiane Shoji Wagner
Especialização em Inteligência Artificial Aplicada
SEPT - Setor de Educação Profissional e Tecnológica
Curitiba,Brasil
cswagnner@gmail.com

Rafaela Mantovani Fontana
Especialização em Inteligência Artificial Aplicada
SEPT - Setor de Educação Profissional e Tecnológica
Curitiba,Brasil
rafaela.fontana@ufpr.br

***Resumo*—O propósito da mineração de texto é extrair informações úteis de um conjunto de dados textuais podendo auxiliar na tomada de decisões. Entender o problema que se quer resolver, definir as estratégias que serão aplicadas como na fases de extração e limpeza dos dados textuais, podem ser consideradas como parte do processo de descobrimento de conhecimento. Nesse contexto, para apoiar a revisão de literatura, desenvolveu-se a aplicação "Caçador de Palavras" com objetivo de processar dados textuais de artigos científicos digitais em formato PDF. O principal produto do processamento dos artigos pela aplicação é a frequência das palavras, obtida utilizando os métodos da linguagem R e os pacotes de biblioteca para Mineração de Texto, realizando a remoção das palavras indesejadas do texto, "lematização", "tokenização", transformando vetor em *dataframe* e por fim, realizando a contagem das palavras. O conhecimento da frequência das palavras pode auxiliar a decisão na seleção de um artigo na condução de pesquisas científicas.**

***Palavras-chaves*—Mineração de Texto, Frequência de palavras, Associação de palavras, Nuvem de palavras.**

***Abstract*—The purpose of text mining is to extract useful information from a set of textual data that can aid in making. Understanding the problem to be solved, defining the strategies that will be applied, such as in the extraction and cleaning phases of textual data, can be considered as part of the knowledge discovery process. In this context, to support the literature review, the "The Word Hunter" application was developed in order to process textual data from digital scientific articles in PDF format. The main product of the processing of the articles by the application is the frequency of the words, obtained using the methods of the R language and the library packages for Text Mining, performing the removal of unwanted words from the text, "lemmatization", "tokenization", transforming vector into *dataframe* and finally, counting the words. Knowing the frequency of words can help the decision to select an article in conducting scientific research.**

***Keywords*—Text Mining, Word Frequency, Word Association, Word Cloud.**

I. DESENVOLVIMENTO

A descoberta de conhecimento em base de dados foi caracterizada pela primeira vez em 1989 no primeiro *workshop* de *Knowledge Discovery in Databases* (KDD) e tinha como objetivo evidenciar que o "conhecimento" é o produto final. A descoberta de conhecimento foi definida como sendo um "processo, de várias etapas, não trivial, interativo e iterativo, de identificar padrões válidos, novos, potencialmente úteis e, em última análise, compreensíveis

em dados" [1, p.83] que envolve o "pre-processamento, a mineração de dados e o pós-processamento" [2, p.83].

O conceito da descoberta de conhecimento é, frequentemente, utilizado como sinônimo para descrever o processo de Mineração de Dados, *Data Mining*. Numa comparação, a Mineração de Texto, *Text Mining*, também pode ser compreendida como pertencente ao processo de descoberta de conhecimento, visto que, são aplicados métodos para obter "conhecimento" dos dados textuais.

O propósito da mineração de texto é extrair informações úteis de um conjunto de dados textuais não estruturados. Antes de iniciar o processo de extração, o entendimento do problema é essencial para definir as estratégias que poderão ser aplicadas no processo. Com o principal objetivo de processar os dados de artigos científicos digitais em formato PDF, foi desenvolvida a aplicação "Caçador de Palavras" utilizando a linguagem R e os pacotes de biblioteca como *pdftools, tm, gglpot2, wordcloud2, tidyverse, tibble*, entre outros. O produto desse processamento é o "conhecimento" que pode ser aplicado para auxiliar na identificação de artigos de uma revisão de literatura sobre um determinado assunto.

O processo de seleção dos artigos, geralmente, resulta numa coleção que são inspecionados e analisados para verificar se encaixam-se no escopo do assunto [15]. Essa inspeção e triagem dos artigos é feita manualmente e consome um tempo considerável. Diante desse contratempo, o principal resultado do processamento do texto pela aplicação desenvolvida é a frequência das palavras. E esse conhecimento poderá auxiliar a tomada de decisão na seleção de um artigo ou de outro, para apoiar a condução de pesquisas científicas.

A. Descrição de Dados

O conjunto de dados utilizado para o estudo e desenvolvimento da aplicação é composto pelos dados textuais de artigos científicos no formato digital e em PDF. O acrônimo significa *Portable Document Format*, Formato de Documento Portátil, e a característica mais expressiva desse formato é a portabilidade identificada pela apresentação do conteúdo. Ou seja, a aparência do documento exibido numa tela de computador é o mesmo apresentado na impressão em papel.

O PDF é um subproduto da linguagem de descrição de página, *PostScript*, que descreve o texto e o conteúdo gráfico de uma página. Diante disso, o PDF também pode ser considerado como uma linguagem de descrição de página [8].

A linguagem PDF declara objetos com instruções que devem ser seguidas. São regras, por exemplo, para renderizar o texto na tela ou o mecanismo de compressão. Esses objetos compõem e conferem consistência ao documento. Alguns dos objetos que podem ser declarados são [8]:

- Fonte do texto: codificação de caracteres;
- Vetores de imagens: conteúdo gráficos;
- Cores: escala de cinza, rgb, cmyk;
- Metadados: título, autor, palavras-chaves;
- Navegação: *hyperlinks*;
- Conteúdo opcional;
- Multimídia;
- Formulários interativos;
- Estrutura do documento;
- Segurança: senha;
- Mecanismo de Compressão.

As informações para a mineração de texto são encontradas nos objetos Estrutura do documento e Metadados.

1) Estrutura do documento

Os objetos que compõem o documento são do tipo Dicionários que são definidos como estruturas de chave-valor. Os valores, que podem ser qualquer objeto PDF, sempre estão associados à uma chave, ou a um nome. O documento é constituído pelos seguintes dicionários: *Trailer Dictionary*, *Document Information Dictionary*, *Document Catalog* e *Page Tree*. O *Trailer Dictionary* é o Dicionário "Pai", o *Document Information Dictionary* é o Dicionário de Informações do Documento, *Document Catalog* é o Catálogo de Documentos e *Page Tree* é a

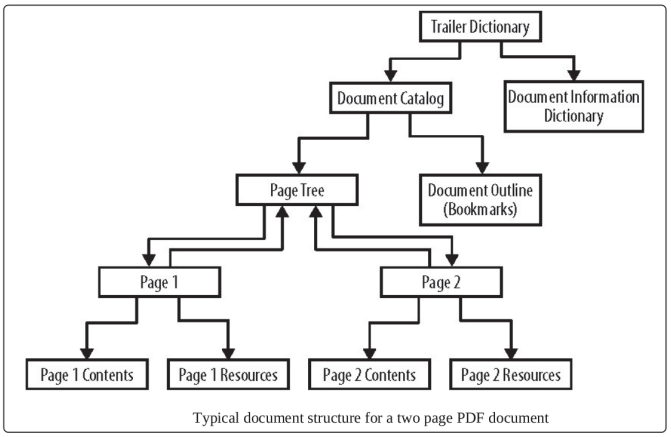


Figure 1. Estrutura do documento PDF com duas páginas. Fonte: Adaptado Livro *PDF Explained* [8, p.58]

Árvore da Página. A Figura 1 ilustra um exemplo da estrutura dos dicionários [8].

Quando o documento é processado por um leitor de PDF, o Dicionário "Pai" deve ser o primeiro a ser lido, pois nele estão as chaves que informam as rotas para acessar os outros dicionários. As principais chaves do Dicionário "Pai" estão relacionadas na Tabela I [8].

Table I
CHAVES DO DICIONÁRIO "PAI"

Chave	Valor
Tamanho (*/Size*)	Número total de entradas na tabela de referência cruzada do arquivo.
Raiz (*/Root*)	O Catálogo de Documentos.
Info (*/Info*)	O Dicionário de Informações do Documento.
ID (*/ID*)	Identificadores para a criação e alteração do arquivo.

O Dicionário de Informação do Documento é o próximo a ser lido. Esse armazena as informações como a data de criação, de modificação, palavras-chaves (*keywords*), os programas utilizados para escrever o documento. As principais chaves deste dicionário estão listadas na Tabela II [8].

Para acessar os dados do Dicionário de Informação do Documento pode ser utilizado o método *pdf_info()*, do pacote *pdftools*. Por exemplo, fornecendo o caminho do artigo, *"Lessons from applying the systematic literature review process within the software engineering domain"* [15], como argumento para o método *pdf_info(caminho)* e

Table II
CHAVES DO DICIONÁRIO DE INFORMAÇÃO DO DOCUMENTO

Chave	Valor
Título (*/Title*)	Título do documento. Não tem relação com o título exibido na primeira página.
Assunto (*/Subject*)	O assunto do documento. É apenas metadado e não tem relação com o conteúdo do texto.
Palavra-chaves (*/Keywords*)	Palavras-chaves associadas ao assunto.
Autor (*/Author*)	Nome do autor.
Data da criação (*/CreationDate*)	Data de criação.
Data de modificação (*/ModDate*)	Data da última modificação.
Criador (*/Creator*)	O programa de criação.
Produtor (*/Producer*)	O programa de conversão do arquivo para o formato PDF.

Name	Value
info	List of length 11

Figure 2. Conteúdo da variável *info*

atribuindo o retorno a variável *info*, tem-se como resultado uma lista de tamanho 11, conforme a Figura 2.

O tamanho da variável *info* indica que 11 chaves podem ser acessadas. Expandindo-a, observa-se, na Figura 3, os nomes das chaves e seus respectivos valores.

Para acessar os valores deve-se utilizar a variável *info* em conjunto com o nome da chave. Por exemplo, para a chave *version* o acesso é *info$version*, que retorna o valor *1.4*, ilustrado na Figura 4.

Name	Value
info	List of length 11
version	'1.4'
pages	13
encrypted	FALSE
linearized	TRUE
keys	List of length 13
created	2007-02-14 14:00:10
modified	2014-10-10 16:35:02
metadata	'<?xpacket begin="ï»¿" id...
locked	FALSE
attachments	FALSE
layout	'single_page'

Figure 3. Variável *info* e as 11 chaves

Seguindo na leitura do documento, o próximo dicionário é o Catálogo do Documento que é o objeto raiz. A partir dele, outros objetos podem ser alcançados e um desses é o Metadados. As principais chaves deste dicionário estão descritas na Tabela III [8].

Table III
CHAVES DO CATÁLOGO DO DOCUMENTO

Chave	Valor
Tipo (*/Type*)	Obrigatoriamente tem que ser */Catalog*.
Páginas (*/Pages*)	Nó primário da árvore de páginas.
Rótulos das páginas (*/PageLabels*)	Fornece rótulos para as página do documento.
Nomes (*/Names*)	O nome do dicionário. Contém os nomes das árvores, e mapeia os nomes das entidades.
Destino (*/Dest*)	Dicionário que mapeia os nomes dos destinos dos *hyperlinks*.
Visualização (*/ViewerPreferences*)	Dicionário de preferência de visualização. Permite que um comportamento específico seja marcado no leitor de PDF quando é visto.
Layout da página (*/PageLayout*)	Especifica o leiaute da página para os leitores de PDF.
Modo da página (*/PageMode*)	Especifica o modo da página para os leitores de PDF.
Marcadores (*/Outlines*)	Bookmarks.
Metadados (*/Metadata*)	O metadados XMP do documento (XML Metadados).

Por fim, o Dicionário de Páginas que contém instruções sobre a renderização da parte gráfica e do conteúdo textual. É a partir dele que qualquer página pode ser encontrada até duas referências distante do nó raiz. As chaves podem ser vista Tabela IV [8].

```
> info$version
[1] "1.4"
> info$pages
[1] 13
> info$created
[1] "2007-02-14 14:00:10 -02"
```

Figure 4. Variável info e chaves

Table IV
CHAVES DA ÁRVORE DE PÁGINAS

Chave	Valor
Tipo (*/Type*)	Obrigatoriamente tem que ser */Pages*.
Pais (*/Parent*)	O nó pai do nó da árvore .
Recursos (*/Resources*).	Os recursos da página como imagens, fontes, ...
Conteúdo (*/Contents*)	O conteúdo gráfico das páginas.

Empregando o método *pdf_toc()*, do pacote *pdftools* da linguagem R, o conteúdo do dicionário pode ser acessado. Fornecendo o caminho do artigo e atribuindo a uma variável, *toc* ← *pdf_toc(caminho)*, obtém-se uma lista de tamanho igual a 2. Essa variável pode ser visualizada na Figura 5.

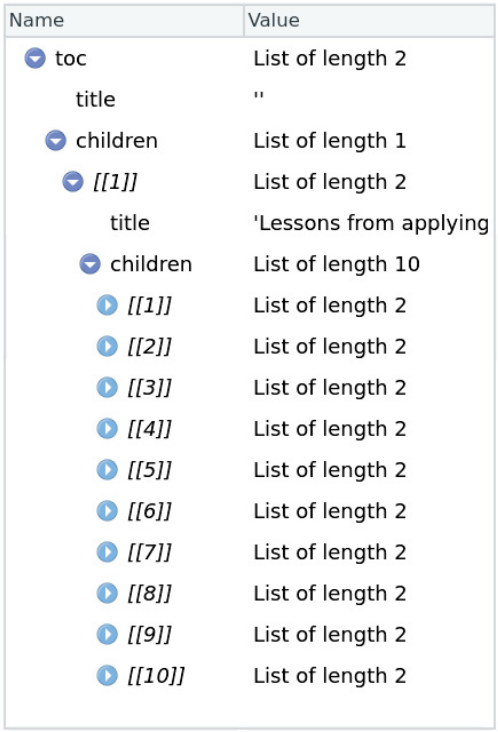


Figure 5. Árvore de Páginas

Nesta variável, tem-se a chave *title* e o primeiro nó *children*, este ramifica-se para o nó *1* que se ramifica para a chave *title* e a outro nó *children*. Esse nó apresenta tamanho igual a 10. Para acessar os dados desse segundo nó *children* é necessário seguir o seguinte caminho: *toc[["children"]][[1]][["children"]][[i]]*, sendo i o

número do nó. Desse modo, os dados são acessado e podem ser extraídos.

2) Metadados

Os metadados armazenam dados sobre título, autor, data da criação, modificação, o programa usado para criar e o para converter o documento para o formato PDF. Os metadados também podem ser acessados pelo arquivo no formato *eXtensible Markup language* XML, pela chave *metadatas* [8].

B. Métodos

Nessa seção são descritos os métodos empregados na preparação e no processamento dos dados textuais.

1) Preparação dos dados

A finalidade da preparação é remover o que não é interessante e deixar os dados limpos para que possam ser utilizados no processamento.

O primeiro passo é extrair os dados utilizando os métodos: *Reader* e *readPDF()*, do pacote *Text Mining* (*tm*).

A assinatura do método *Reader* possui parâmetros, conforme o Método 1, que devem ser fornecidos:

1) *uri*: caminho do arquivo;
2) *language*: o idioma do arquivo;
3) *id*: identificação para o documento de texto criado.

Método 1 - *Reader*

```
function(elem, language, id){
    uri ← processURI(elem$uri)
    meta ← pdf_info(uri)
    content ← pdf_text(uri)
    PlainTextDocument (
        content,
        meta$Author,
        meta$CreationDate,
        meta$Subject,
        meta$Title,
        basename(elem$uri),
        language,
        meta$Creator
        )
}
```

No entanto, utilizando somente o método *Reader* não é possível efetuar a extração, sendo necessário o uso de um segundo método para atuar em conjunto. Este método é o *readPDF()* e fará a leitura do documento no formato específico PDF. Sua assinatura pode ser vista no Método 2:

Método 2 - *ReaderPDF*

```
readPDF(engine, control)
```

Combinando os dois métodos, *Reader* e *readPDF()*, tem-se a função de extração que retorna um *PlainTextDocument* na forma de um vetor com todas as linhas do documento e os metadados.

Método 3 - *extração*

```
extracao ← function(caminho){
    language ← "pt"
    mecanismo ← "xpdf"
    leitor ← readPDF(engine = mecanismo)
    texto ← leitor(elem = list(uri = caminho),
        language = language,
        id = "id1")
}
```

O argumento *engine*, destacado no Método 3, é o mecanismo de extração. As opções de valores disponíveis para este mecanismo são:

1) *"pdftools"*: utiliza a biblioteca de renderização *Poppler PDF* baseada no xpdf-3.0 que é disponibilizada nas funções *pdf_info* e *pdf_text* do pacote *pdftools*;
2) *"xpdf"*: disponível por *Xpdf* Leitor de PDF;
3) *"Rpoppler"*: utiliza a biblioteca de renderização *Poppler PDF* disponibilizadas nas funções *PDF_info* e *PDF_text* do pacote *Rpoppler*;
4) *"ghostscript"*: é um interpretador da linguagem *PostScript* e de arquivos PDF;
5) *"Rcampdf"*: *PerlCAM::PDF* biblioteca de manipulação disponibilizada pelas funções *pdf_info* e *pdf_text* do pacote *Rcampdf*;

6) *"custom"*: mecanismo de extração desenvolvido pelo usuário.

A opção utilizada na aplicação foi *xpdf*. Além do argumento *engine*, também foi destacado a variável **texto** para indicar que é ela que recebe os dados extraídos do documento. A partir da variável texto, os dados do texto e os metadados podem ser acessados pelas chaves *content* e *meta* e atribuídos cada um à uma variável. O acesso às variáveis pode ser vista no Método 4.

Método 4 - Forma de acesso as variáveis

```
conteudo ← texto$content
metadados ← texto$meta
```

Concluído o processo de extração, inicia-se a limpeza dos dados. Esta etapa consiste em remover os dados considerados irrelevantes.

Os dados textuais são tudo que compõem o texto. São palavras, preposições, verbos de ligações, pontuação, números, caracteres não imprimíveis. Muitos elementos da composição textual são considerados ruídos, sujeira e não possuem um significado. São esses elementos que devem ser removidos do texto [2].

2) Quantidade de ruído na seção Referência

Um ponto observado nos artigos é que a seção de Referências contém muitos itens, que chega a ser superior a 50.

As referências são compostas, geralmente, do nomes dos autores: sobrenome em caixa alta, o nome abreviado seguido de ponto, se for mais de um autor, devem ser separados por ponto e vírgula. O título, subtítulo, os dados sobre a edição, indicada pela abreviação *ed.*, o nome da cidade, o ano da publicação. Se tiver indicativo de páginas, deve ser abreviado por *pp*, seguido do intervalo de número das páginas. Tudo isso gera muito ruído para a análise do conteúdo do artigo.

Além do fato que os títulos e subtítulos podem conter palavras que sejam de interesse. Como exemplo a palavra *agile*, pode ocorre várias vezes a mais e ser contabilizada, afetando de alguma forma a tomada de decisão, pois, erroneamente pode-se selecionar um artigo que não fala exatamente sobre o assunto, mas foi pesquisado em livros que no título continha a palavra *agile*.

Dessa forma, com o objetivo de eliminar essa sujidade, foi desenvolvido um método, descrito em Método 5, para remover a seção de Referências e assim retirar do texto uma quantidade significativa de pontos, números, abreviações e palavras que não devem ser contabilizadas. Outro motivo para essa remoção é para reduzir a frequência da expressão "*et al.*".

O método se baseia nas palavras *Abstract* e *References* e procura a posição delas no vetor de caracteres que representa o texto. Com os valores dessas posições faz-se a remoção do conteúdo entre elas resultando em um novo vetor.

Contudo, se por conta da formatação a função não reconhecer a grafia da palavras *Abstract* e *References*, no caso de artigos no idioma inglês, ou Resumo e Referências, para o idioma português, nesse caso, serão contabilizadas as palavras dessa seção.

3) StopWords - *"Para-Palavras"*

O passo seguinte é a remoção de uma coleção de palavras do texto que não possuem um real significado. Para realizar a remoção foi utilizada uma sequência de métodos começando pela função *removeWords()*, do pacote *tm*. Essa função remove as palavras do texto que são passadas como argumento na função, resultando em um novo vetor sem estas palavras. A assinatura do método está descrita no Método 6:

Método 6 - removeWords()

```
removeWords(x, words)
```

Na função, *x* representa o texto e *words* as palavras a serem removidas. O parâmetro *words* aceita, como valor, um vetor de palavras.

A função *stopwords()* disponibiliza fontes de palavras do tipo *stopwords*, ("para-palavras"), em diferentes idiomas. O resultado dessa função é um vetor de palavras no idioma escolhido. Por exemplo, definindo o idioma

Método 5 - Extração do conteúdo

```
extracao_conteudo_en ← function(texto_extraido)
{
conteudo ← texto_extraido$content
valorA ← is_empty(
          which(conteudo in "Abstract"))
if valorA then
  posicao_1 ← 1
else
  titulo ← c("Abstract", "ABSTRACT")
  vetorAbstract ← match(conteudo, titulo)
  for i in 1:length(vetorAbstract) do
    if !is.na(vetorAbstract[i]) then
      posicao_1 ← i
    end if
  end for
end if
posicao_1
valoR ← is_empty(
          which(conteudo in "References"))
if valorR then
  posicao_2 ← length(conteudo)
else
  titulo2 ← c("References", "REFERENCES",
  "Bibliography")
  vetorReferences ← match(conteudo, titulo2)
  for i in 1:length(vetorReferences) do
    if !is.na(vetorReferences[i]) then
      posicao_2 ← i
    end if
  end for
end if
posicao_2
conteudo ← conteudo[c(posicao_1:posicao_2)]
return conteudo
}
```

inglês, valor "en", para a função: *stopwords(en)*, um vetor de palavras inglesas como: *this*, *that*, *the*, *for*, é retornada e esse vetor pode ser utilizada pela função *removeWords()*. Dessa forma, as palavras a serem removidas podem ser definidas pela função *stopwords()*.

Método 7 - stopwords()

```
stopwords(language, source, simplify)
```

Para o idioma inglês foi utilizado duas fontes de palavras para a função *stopwords()*. Para uma fonte foi definido o valor "en" para o parâmetro *language* e para outra, foi ajustado o valor *"smart"* para o parâmetro *source*. A diferença entre as duas é a quantidade de palavras. A primeira retorna um vetor de 175 caracteres, enquanto a segunda, um vetor de 571 caracteres. As duas opções foram utilizadas para se complementarem e remover o máximo de palavras.

Para o idioma português foram utilizadas mais de uma fonte de palavras. Uma delas foi a utilização dos parâmetros *language* com o valor "pt" e *source* com o valor *"snowball"*. Contudo, essa fonte retorna uma quantidade de palavras igual a 203 e, de forma a complementar o conjunto de palavras a serem removidas, foram criados três vetores com palavras consideradas como "para-palavras", que são preposições como: a, ante, após, até, com, contra, de, entre outras e verbos de ligações. Os ajustes dos parâmetros no método pode ser visto no Método 8.

Método 8 - removeWords() e stopwords()

```
removeWords(stopwords("en"))
removeWords(stopwords(source ="smart"))

removeWords(stopwords(languange = "pt",
        source ="smart"))
```

Seguindo com o processo de limpeza, o próximo passo foi remover os caracteres especias representados pelos símbolo matemáticos como arroba (@) e porcentagem (%), os caracteres de pontuação como vírgula e ponto, os de formatação como tabulação ($\backslash t$) e nova linha ($\backslash n$) e outros.

Para essa remoção foi empregada a função *str_replace_all()*, do pacote *stringr*, que permite a substituição de múltiplos elementos. A assinatura do método está descrito no Método 9.

Método 9 - str_replace_all

```
str_replace_all(string, pattern, replacement)
```

Na função, *string* representa o texto, *pattern* os caracteres especiais a serem substituídos e *replacement*, pelo o quê serão trocados. No caso, os caracteres especiais foram substituídos por um espaço em branco. Por consequência desta substituição, o vetor resultante apresentou uma quantidade maior de espaços em branco e para removê-los utilizou-se da função *stripWhitespace()*, do pacote *tm*. Essa função remove os espaços extras para um único espaço em branco.

A etapa de limpeza foi repetida mais uma vez com o objetivo de remover a sujidade remanescente. Terminada esta etapa, inicia-se o processamento dos dados.

O objetivo do processamento é transformar os dados limpos em conhecimento.

4) Lematização versus Stematização

As palavras possuem uma raiz, que é o elemento que concentra o significado das palavras e da qual derivam as suas inflexões. Podem se apresentar no plural, singular, conjugadas nos tempos verbais como: ir, foi, fui, ia, foram etc. Para tratar as diferentes formas das palavras pode-se utilizar dois processos: Lematização ou Stematização.

A stematização é um processo que remove sufixos e afixos das palavras flexionadas extraindo a raiz da palavra. Por exemplo, as palavras: livro, livrinho, livreiro, livraria tem a raiz da palavra livr, e este será o resultado do processo de stematização. O resultado pode ser uma palavra sem sentido.

A lematização é um processo que também reduz a palavra à raiz, mas mantém o significado da palavra. Por exemplo, as palavras: "*Search*", "*Research*", "*Review*" e "*View*" são preservadas nessa forma e tem o seu significado mantido.

Devido ao interesse no significado optou-se pela lematização das palavras. Os métodos utilizados para esse fim foram a *make_lemma_dictionary()* e *lemmatize_strings()* [2], descritos em Método 10.

Método 10 - Lematização

```
make_lemma_dictionary(..., engine, lang)
lemmatize_strings(x, dictionary)
```

A função *make_lemma_dictionary()* gera um dicionário com as palavras que correspondem as inflexões das palavras. O parâmetro *engine*, destacado no método, é quem gera o dicionário e possui 3 opções, *lexicon*, *hunspell* e *treetragger*, de geradores para utilizar.

Os métodos *make_lemma_dictionary()* e *lemmatize_strings()* foram utilizados na criação das funções *lematizacaoLex()*, descrito no Método 11, e *lematizacaoHun()*. Estas retornam um vetor de palavras com as palavras lematizadas. A diferença entre elas é o uso do mecanismo. Para a função *lematizacaoLex()* foi empregado o *lexicon* que é uma coleção de tabelas *hash*, dicionários e listas de palavras. Para *lematizacaoHun()*, o mecanismo utilizado foi o *hunspell*, que é um verificador ortográfico de baixo nível que utiliza a biblioteca de mesmo nome. O mecanismo é utilizado como verificador ortográfico por editores de texto como o *LibreOffice*.

Método 11 - lematizacaoLex()

```
lemmatizacaoLex ← function(texto){
  lemma_dictionary_lex ←
    make_lemma_dictionary(
      texto,
      engine = "lexicon"
    )

lex_lem ← lemmatize_strings(
    texto,
    dictionary = lemma_dictionary_lex
    )

return(lex_lem)
}
```

Para explicar o uso da função *lematizacaoLex()*, utilizou-se o mecanismo *lexicon* e o texto do artigo na função *make_lemma_dictionary()* para gerar o dicionário com as palavras na forma flexionadas a partir das palavras do texto. A função *lemmatize_strings()* utiliza o dicionário resultante da função *make_lemma_dictionary()* para realizar o processo de lematização do texto e retornar um vetor de *strings* lematizados, essa utilização pode ser vista no Método 11

Avançando no processamento, o vetor resultante da lematização é convertido em *dataframe* para passar pelo processo de *tokenização*. Para a conversão do vetor em *dataframe* e a *tokenização* foram empregadas, respectivamente, as funções: *enframe()* e *unnest_tokens()*. As assinaturas dos métodos podem ser visto no Método 12.

Método 12 - Tokenização

```
enframe(text, name, value)
unnest_tokens(tbl, output, input, token, ...)
```

Com *enframe()*, do pacote *tibble*, o vetor foi convertido em *dataframe* com duas colunas, uma para armazenar o nome e outra os valores. Nessa função os parâmentros são *text* que representa o texto, *name*, o nome e *value*, os dados do texto.

A função *unnest_tokens()*, do pacote *tidytext*, realiza a tokenização baseado no valor utilizado no parâmetro *token*, que indica a unidade em que os dados serão "tokenizados". As opções de valores são *"word"*, que é a opção padrão, *"characteres"*, *"sentences"*, *"lines"* etc. A unidade utilizada para *token* foi *"word"*, o que significa que a tokenização foi feita palavra por palavras. Ao final desse processo, o resultado obtido é o texto "tokenizado", pronto para ser utilizado.

Método 13 - Funções de tokenização ajustados

```
enframe(texto,
        name = NULL,
        value = "text")
unnest_tokens(token = "word", text)
```

Para a aplicação "Caçador de Palavras", o *dataframe* "tokenizado" foi utilizado para a gerar a visualização dos dados. São gerados dois gráficos de colunas para demonstrar a frequência das palavras, um para os dados lematizados com o mecanismo *lexicon* e outro com *hunspell*, para que os dados possam ser comparados. Também será gerada uma nuvem de palavras para visualizar a frequência pelo tamanho das palavras.

Outra visualização é a associação das palavras, em que são exibidas as que mais se associam em par. E por fim, um gráfico de coluna da seção, em que é selecionada uma seção do artigo e o gráfico de coluna com a frequência das palavras da seção, lematizadas por *lexicon*, é gerado.

A seguir serão descritos os métodos utilizados para cada visualização da aplicação.

5) Gráfico de Colunas

Para a geração do gráfico de colunas utilizou-se a função *ggplot()* em conjunto com as *geom_col()* e *geom_label()*, todos do pacote *ggplot2*. A função *ggplot()* não apresenta nada sem as funções *geom_col()* e *geom_label()*, que são as que realmente plotam o gráfico. A função *geom_col()* faz a plotagem no formato de coluna e a *geom_label()* cuida dos rótulos do gráfico. Em Método 14 estão as funções e seus ajustes para gerar o gráfico de colunas.

Método 14 - gráfico de colunas

```
ggplot(palavras_filtradas) +
  geom_col(aes(x = frequency, y = word,
    fill = frequency, group = frequency)) +
  geom_label(aes(x = frequency, y = word,
    label = frequency), size = 6) +
  labs(x = "Frequência", y = "Palavras") +
  geom_label(aes(x = frequency, y = word,
    label = word), size = 6,
    position = position_stack(vjust = 0.5))
```

6) Gráfico de Colunas por Seção

O gráfico de colunas por seção faz uso do mesmo processo empregado para o gráfico de colunas utilizando o mecanismo *lexicon*. A diferença está no conteúdo, que consiste do texto da seção.

Para extrair o conteúdo da seção foi utilizado o mesmo método para remover o conteúdo entre as seções *Abstract* e *References*, apenas que os valores serão os títulos das seções. No lugar de *Abstract* usa o título da seção selecionada e no lugar de *References*, o título da próxima seção referente a seção selecionada.

Para obter os títulos das seções foi utilizada a função *pdf_toc()*, do pacote *pdftools* e desenvolvida a função lista() para extrair os títulos, descrita no Método 15. Esta lista é exibida na aplicação para que o título da seção possa ser selecionado.

Método 15 - Lista dos títulos

```
lista ← function(caminho)
{
toc ← pdf_toc(caminho)
toc_lista ← c()
condicao ← length(toc) > 0
if condicao then
  toc_lista ← toc[["children"]][[1]][["children"]]
  condicao2 ← length(toc_lista) > 0
  if condicao2 then
    listaTitulos ← c()
    tamanhoToc ← length(toc_lista)
    for i in 1:tamanhoToc do
      listaTitulos ← c(listaTitulos,
      toc[["children"]][[1]][["children"]][[i]][["title"]])
    end for
    listaTitulos ←
    stringi::stri_remove_empty(listaTitulos)
    return(listaTitulos)
  end if
else
  return("Lista Vazia")
end if
}
```

*7) Nuvem de Palavras (*WordCloud*)*

A geração da Nuvem é simples, pois é aplicação direta da função *wordcloud2()*, do pacote de mesmo nome, e realizando ajustes nos valores dos parâmetros. A função empregada está descrita no método 16.

Método 16 - Wordcloud2

```
wordcloud2(palavras_filtradas,
        size = 1,
        color = "random-light",
        backgroundColor = "white")
```

8) Associação de Palavras

Para gerar a visualização dos dados para a associação foi empregada a função *ggraph(graph, layout, ...)*, do pacote de mesmo nome. Sendo, *graph* o objeto que contém o gráfico, *layout*, o tipo do *leiaute* criado.

Para a associação de palavras foi preciso realizar o processo de tokenização novamente. Pois, para a geração do gráfico de colunas foi utilizado o valor *"word"* para o parâmetro *token*. Para a associação é necessário utilizar o valor *"ngrams"* e informar a quantidade de palavras a serem associadas, para duas palavras, n = 2. Outras funções foram utilizadas para estilizar a visualização da flecha, do tamanho da flecha, do ponto etc.

Método 17 - Ngrams

```
unnest_tokens(bigram,
              text,
              token = "ngrams",
              n = 2)
```

Após a realização da tokenização, faz-se a modelagem da visualização dos dados. As funções utilizadas para o leiaute estão descritas no Método 18.

Método 18 - Ggraph

```
ggraph(bigram_graph, layout = "graphopt") +
  geom_edge_link(arrow = arrow(
  length = unit(5, 'mm')),
  end_cap = circle(3, 'mm')) +
  geom_node_point() +
  geom_node_text(aes(label = name),
  size = 8, vjust = 0.5, hjust = 0.5) +
  coord_cartesian(xlim = ranges$x,
  ylim = ranges$y, expand = FALSE) +
  theme_graph(background = "grey98")
```

A aplicação desse conjunto de funções e dos ajustes dos valores dos parâmetros de cada função, obteve-se os resultados discutidos na seção Discussões e Resultados. A seguir serão detalhados as tecnologias utilizadas.

II. Tecnologias

Para o desenvolvimento da aplicação "Caçador de Palavras" foi utilizado um *laptop* pessoal da marca *Dell* G15, core i5, com sistema operacional *Ubuntu 20.04*. Para a aplicação utilizou-se do ambiente de desenvolvimento *RStudio*, edição *Open Source*, versão 4.2.1 (2022-06-23) "Funny-Looking Kid", e as seguintes bibliotecas:

- Pacote *base*, versão 4.2.1, *The R Base Package*;
- Pacote *bslib*, versão 0.4.0, *Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'*;
- Pacote *dplyr*, versão 1.0.10, *A Grammar of Data Manipulation*;
- Pacote *ggplot2*, versão 3.3.6, *Create Elegant Data Visualisations Using the Grammar of Graphics*;
- Pacote *ggraph*, versão 2.0.6, *An Implementation of Grammar of Graphics for Graphs and Networks*;
- Pacote *igraph*, versão 1.3.4, *Network Analysis and Visualization*;
- Pacote *pdftools*, versão 3.3.0, *Text Extraction, Rendering and Converting of PDF Documents*;
- Pacote *rlang*, versão 1.0.6, *Functions for Base Types and Core R and 'Tidyverse' Features*;
- Pacote *shiny*, versão 1.7.2, *Web Application Framework for R*;
- Pacote *shinycssloaders*, versão 1.0.0, *Add Loading Animations to a 'shiny' Output While It's Recalculating*;
- Pacote *shinyjs*, versão 2.1.0, *Easily Improve the User Experience of Your Shiny Apps in Seconds*;
- Pacote *stopwords*, versão 2.3, *Multilingual Stopword Lists*;
- Pacote *stringr*, versão 1.4.1, *Simple, Consistent Wrappers for Common String Operations*;
- Pacote *tau*, versão 0.0-24, *Text Analysis Utilities*;
- Pacote *textstem*, versão 0.1.4, *Tools for Stemming and Lemmatizing Text*;
- Pacote *tibble*, versão 3.1.7, *Simple Data Frames*;
- Pacote *tidytext*, versão 0.3.4, *Text Mining using 'dplyr', 'ggplot2', and Other Tidy Tools*;
- Pacote *tidyverse* versão 1.3.2, *Easily Install and Load the 'Tidyverse'*;
- Pacote *tm*, versão 0.7-8, *Text Mining Package*;
- Pacote *wordcloud2*, versão 0.2.1, *Create Word Cloud by 'htmlwidget'*.

III. Discussões e Resultados

Nessa seção são descritos os resultados obtidos, os reveses encontrados, a motivação da escolha entre uma a outra função, artigos e aplicação similares que utilizaram a mineração de texto como uma ferramenta de apoio.

Para a apresentação dos resultados foi utilizado o artigo *"Lessons from applying the systematic literature review process within the software engineering domain",* do autor Pearl Brereton et al.[15].

Ao acessar a aplicação, o primeiro passo é selecionar o idioma do texto, português ou inglês. Informado o idioma, o campo *upload* (carregar documento) é revelado, conforme a mostra a Figura 6.

Seis abas são apresentadas, cada qual exibe os resultados relacionados com os títulos, respectivamente: Informações do artigo, Gráfico de Palavras - *Lexicon*, Gráfico de Palavas - *Hunspell*, '*WordCloud*' - Nuvem de Palavras, Associação de Palavras e Gráfico por Seção.

1) Informação do artigo

As informações do documento mostradas na aplicação são: Autor, Título, Número de Páginas, Palavras-Chaves, Criador, DOI e Outros. Para o Idioma selecionado e a Lista de seções os dados são gerados pela aplicação, os demais são extraídos dos dados informados no

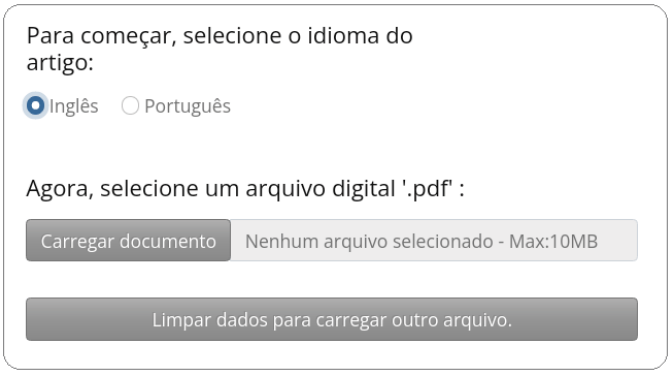


Figure 6. Tela da aplicação.

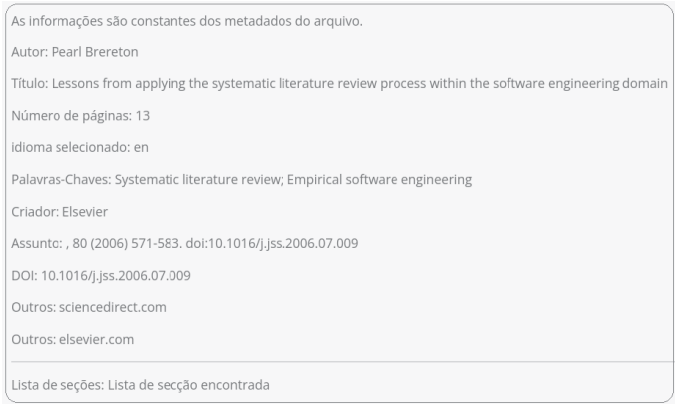


Figure 7. Informação do artigo

artigo científico. Estes dados podem ou não terem sido informados no artigo.

A lista de seções indica se no Dicionário de Páginas estão descritos os títulos das seções. Se estiver, o valor de Lista de seções é igual a "Lista de seção encontrada - O gráfico de seção poderá ser gerado", mostrando que o gráfico de coluna por seção poderá ser gerado. Se não tiver, o valor é "Lista de seção NÃO encontrada - Não será possível gerar o gráfico por seção" (Figura 7). Nesse caso, não será possível gerar o gráfico de colunas.

2) Gráfico de Palavras - Lexicon

Na opção de geração de Gráfico de Palavras - *Lexicon*, a aplicação oferece um filtro que permite alterar o valor de frequência das palavras num intervalo entre 1 a 100. O gráfico é atualizado quando o valor é alterado.

O gráfico de coluna é resultado do processamento da lematização utilizando o mecanismo *Lexicon* para gerar o dicionário. A Figura 8 mostra como o gráfico será plotado, as palavras e seus respectivos valores de frequência. Para demonstrar, a palavra "Review" apresenta a maior frequência igual a 176, "Search" apresenta 55 e "Research" 64. Como a menor frequência mostrada no gráfico é igual a 51, a frequência da palavra *"Reviewer"* não é exibida por estar fora do intervalo, mas ela ocorre 19 vezes.

3) Gráfico de Palavras - Hunspell

O gráfico de coluna é resultante do processamento de lematização utilizando o mecanismo *Hunspell* para gerar o dicionário. Na Figura 9 pode ser visto no gráfico a palavra *"View"* com frequência igual a 198 e *"Search"* igual a 136. Observa-se um aumento nos valores de frequência dessas palavras e a ausência da palavra *"Research"* .

Essa diferença é decorrente do uso dos diferentes mecanismos para gerar os dicionários utilizados na lematização dos dados textuais. O mecanismo *Lexicon* é menos criterioso, pois no resultado encontra-se palavras com significados diferentes, mas com a mesma raiz, *"Search"* , *"Research"* e *"Review"*. Já o *hunspell* é mais criterioso, *"Research"* e *"Review"* foram diminuídos para *"Search"* e *"View"*. Tanto que essa diferença pode ser observada pelo aumento no número de ocorrência dessas palavras.

Para demonstrar, no gráfico *lexicon*, a palavra *"Review"* apresenta valor igual a 176. *"Reviewer"* tem valor igual a 19. Somando os dois valores o resultado é 195. No gráfico *hunspell*, a palavra *"View"* apresenta valor igual a 198. O que significa que as palavras *"Review"* e *"Reviewer"* foram diminuídas a palavra *"View"* (Figura 9).

Da mesma forma, no gráfico *Lexicon*, as palavras *"Search"* e *"Research"* apresentam valores iguais a 55 e 64, respectivamente. No gráfico *hunspell*, a palavra *"Research"* não é apresentada, mas a frequência da palavra *"Search"* aumentou, passando de 55 para 136, o que indica que a palavra *"Research"* foi diminuída a *"Search"* (Figura 8).

4) Nuvem de Palavras

Uma nuvem de palavras é uma coleção de palavras que se apresentam com tamanho diversos. As maiores contrastam com as menores, indicando que as maiores ocorrem mais vezes. Para a nuvem de palavras utilizou os dados lematizados pelo mecanismo *Lexicom*, por consequência, a nuvem apresenta palavras com significados diferentes, mas que derivam da mesma palavra base, conforme mostra a Figura 10.

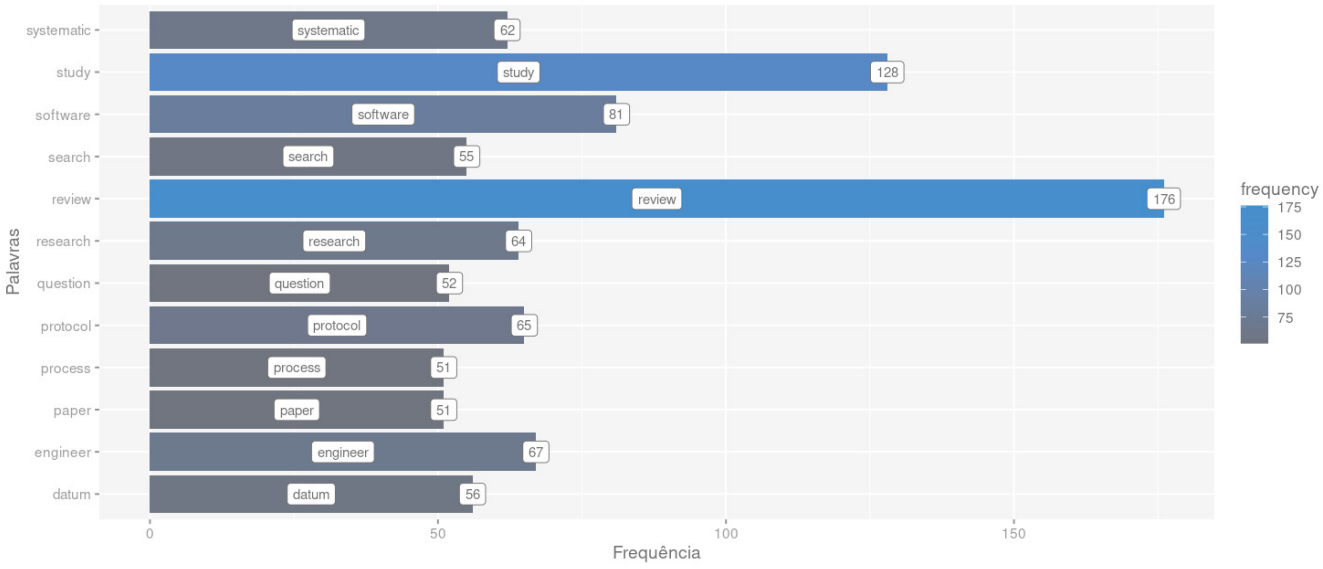


Figure 8. Resultado dos dados lematizado com mecanismo *Lexicon*

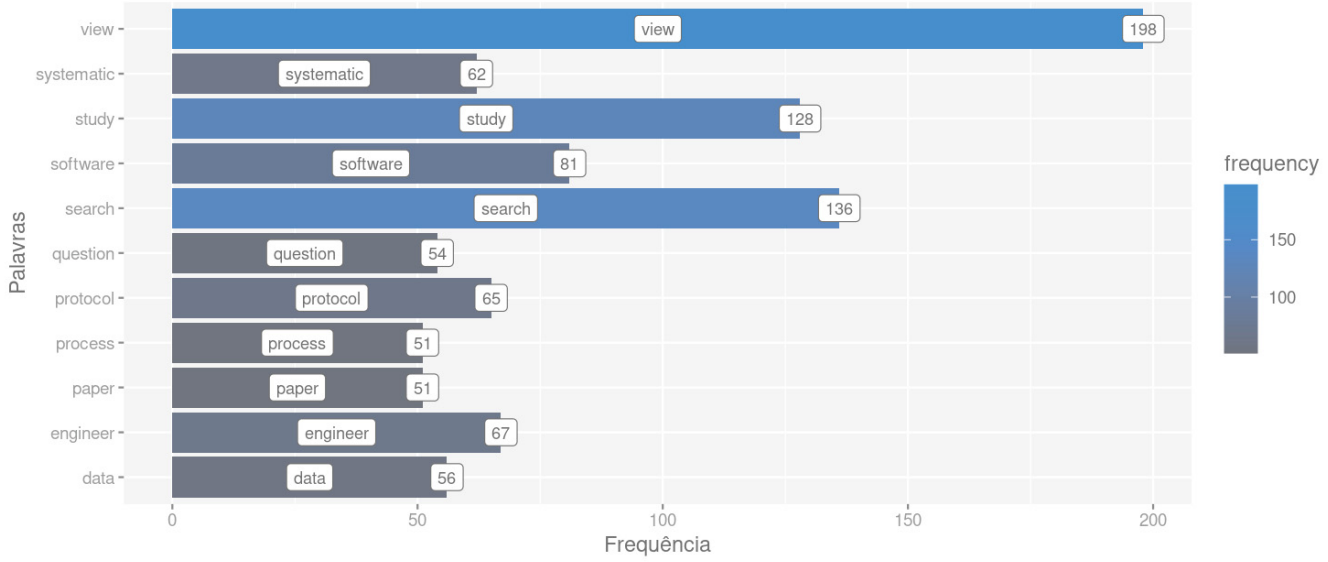


Figure 9. Gráfico resultante dos dados lematizado com mecanismo *Hunspell*

5) Associação de palavras

Associação descreve a relação das palavras, ou seja, quantas vezes a ocorrência de duas palavras ocorrem associadas. A Figura 11 ilustra a associação de algumas. No topo, observa-se as palavras: *journal*, *system*, *software*, *engineer* e *domain* associadas. As palavras *journal*, *system* e *software* tem sua razão por estarem na mesma frase, estão no cabeçalho do artigo (Figura 12) e ocorrem algumas vezes. Já o segundo conjunto de palavras associadas, apresentam-se em dupla:

- *review-report*;
- *develop-review*;
- *review-teams*;
- *review-protocol*;
- *systematic-review*;
- *systematic-literature*;

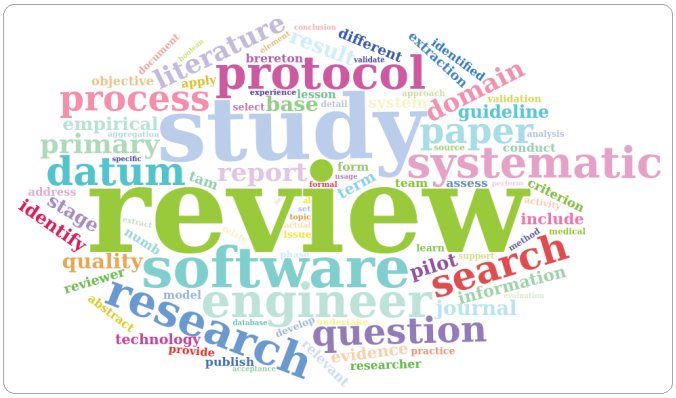


Figure 10. Nuvem de palavras (wordcloud)

- *review-literature*;
- *review-process*.

A palavra *review* é predominante, ocorre em quase todas as associações, indicando que o assunto relacionado a palavra *review* (revisão) é abordado no artigo.

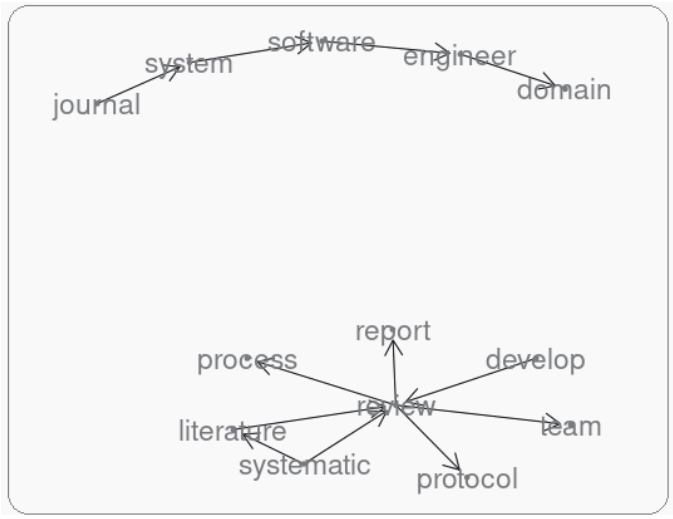


Figure 11. Associação de palavras

The Journal of Systems and Software 80 (2007) 571–583

Figure 12. Texto do cabeçalho do artigo

6) Gráfico de Palavras por Seção

O gráfico por seção refere-se à informação do conteúdo textual de uma seção, permitindo que o usuário obtenha uma análise específica por seção do artigo científico. O título da seção é escolhido a partir da lista de títulos gerada com as informações inseridas no documento do artigo. Portanto, existe uma dependência dessas informações para gerar o gráfico por seção. Se essas informações não estiverem no documento, o gráfico não poderá ser plotado.

Pode ocorrer, também, que a lista de títulos das seções tenha sido gerada, mas o gráfico da seção não ter sido plotado. O motivo seria que o título da seção nos dados do documento ter sido informado com uma formatação textual e o conteúdo, processado em outra formatação, o que leva o algoritmo não encontrar o título devido à essas diferenças na formatação textual. Isso pode acontecer para títulos de seções com uma palavra, com duas, mas o usual é com mais de duas palavras.

Para demonstrar, a lista de títulos das seções do artigo possui os seguintes títulos [15]:

- *Introduction*;
- *Systematic literature review*;
- *Service-based systems: a systematic review of issues (R1)*;
- *Systematic literature review of the technology acceptance model and its predictive capabilities (R2)*;
- *Systematic literature review of guidelines for conducting systematic literature reviews (R3)*;
- *Lessons learned*;
- *Discussion*;
- *Conclusions*;
- *Acknowledgements*;
- *References*.

Selecionando o título da seção *"Lesson Learned"*, o gráfico para essa seção foi plotado, conforme ilustra a Figura 13. No entanto, selecionando o título *"Systematic literature review of guidelines for conducting systematic literature reviews (R3)"* o gráfico não foi plotado, conforme a Figura 14.

No desenvolvimento de uma aplicação, é usual deparar-se com algumas situações que devem ser contornadas. Para o "Caçador de Palavras" destacam-se

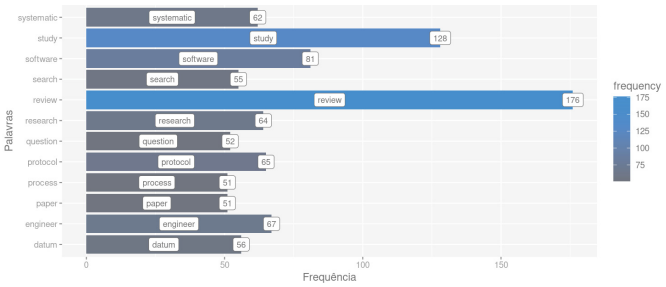


Figure 13. Gráfico de palavras da seção “Lesson Learned”

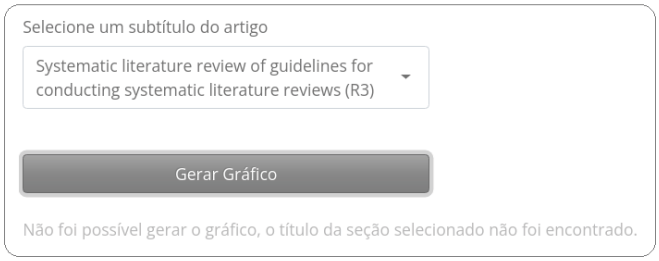


Figure 14. Gráfico de palavras da seção “Systematic literature ... (R3)”

duas: a extração dos dados e a ordem de remoção dos elementos textuais. Além disso, foi encontrada uma ferramenta similar utilizada no contexto educacional, denominada “Sobek” [18]. Ainda, outros artigos que utilizaram a mineração de texto como uma ferramenta de apoio para a pesquisa.

7) *Extração dos dados*

Para a extração dos dados, a situação foi em relação a função empregada para extrair o conteúdo do documento. Nas primeiras tentativas de extração do texto a função utilizada foi a *pdf_text()*, do pacote *pdftools*. Essa função processa os quadros, páginas, de texto e retorna um vetor de tamanho igual ao número de páginas do documento, sendo cada página uma posição do vetor.

Normalmente, os artigos científicos são redigidos em uma ou duas colunas de textos. A princípio, a função utilizada atendia o propósito de extração do texto. No entanto, ela funciona bem para artigos redigidos em uma coluna, não sendo satisfatória para artigos com duas colunas, pois observou-se que a leitura do texto é feita da margem esquerda para a direita em linha reta, sem considerar o espaço entre as colunas. Dessa forma, uma linha da coluna da esquerda juntava-se com a linha da coluna da direita que estivesse na sequência, ou na mesma altura da linha, o que pode ser vista na Figura 15.

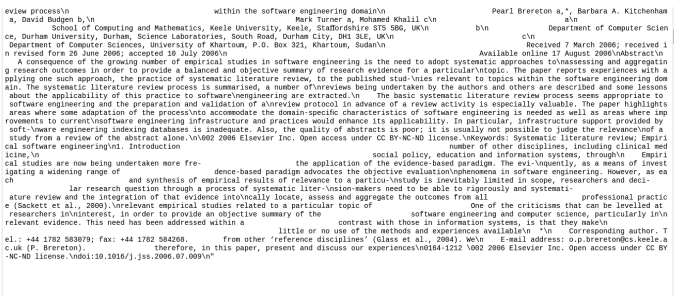


Figure 15. Visualização do texto extraído com *pdf_text()*

Esse viés tornou complexa a extração do conteúdo por seção e fez com que outra função fosse pesquisada para contornar a situação. A escolhida foi a *readPDF()*, do pacote *tm*. Diferente da outra, essa função considera as colunas do texto, de forma que a leitura é feita por linha de coluna.

8) *Remoção dos dados*

O processo de limpeza dos dados textuais consiste da etapa de remoção dos dados e a ordem em que é executada pode influenciar no resultado final.

Um exemplo na ordem é a remoção dos caracteres de pontuação. O hífen é um item delicado pois pode afetar as palavras hifenizadas. Se o elemento hífen é removido antes, as palavras hifenizadas perdem o hífen e resultando em duas palavras com significado distinto da palavra original. Por esse motivo, foi avaliada a utilização de expressões regulares para remover as palavras com hífen, contudo o resultado não foi satisfatório. Capturar uma palavra que termina com hífen “\-” e o final da linha, “\n” ou “\r” demandava muito esforço para uma quantidade pequena de palavras hifenizadas.

Outro ponto é a remoção das “para-palavras” no idioma português. Observou-se que as palavras que na sua grafia apresentam cedilha (ç) ou til (˜), tem as vogais seguintes removidas. Por exemplo, a palavra “atenção”, após o processo de remoção, fica “atenç”, o que pode interferir no resultado da frequência das palavras.

9) Áreas de atuação

A mineração de texto pode ser uma ferramenta auxiliar em outras áreas. A exemplo, o grupo de pesquisa GTech.Edu da Universidade Federal do Rio Grande do Sul (UFRGS) desenvolveu a aplicação *Sobek*, que inicialmente foi projetada para ajudar "os professores do ensino a distância a avaliarem o trabalho dos alunos" [17]. O resultado da ferramenta Sobek é um quadro mostrando os termos centrais e a relações entre eles, e desde de seu desenvolvimento inicial, a ferramenta tem ampliado seu uso.

Além do uso educacional, a mineração de texto pode ser empregada na revisão de literatura de outras áreas. Por exemplo, para o entendimento das formas de tecnologias utilizadas pelas de cidades inteligentes no "planejamento urbano, na engenharia eletrônica, e na ciência da computação" escrito pelo autor Chiehyeon Lim [18].

Outro exemplo do uso da mineração de texto na revisão de literatura é sobre o artigo pelo autor Amit Kumar Kushwaha [19], em que a revisão de literatura é utilizada para analisar as áreas emergentes de gerenciamento suportada por *big data* utilizando análise de redes empregada pelo processamento de linguagem natural.

Para finalizar, foi apresentada a aplicação "Caçador de Palavras" desenvolvida em linguagem R, que processa os dados textuais de um artigo científico digital no formato PDF. Como resultado do processamento, a aplicação apresenta a frequência das palavras na forma de gráfico de coluna resultante de dois mecanismos de lematização diferente. Além de uma nuvem de palavras e a associação delas. A aplicação merece um estudo e aprimoramento na remoção dos elementos textuais não relevantes para melhorar a eficiência, assim como, uma opção para remover o cabeçalho dos artigos que apresentam essa estrutura no texto. Outro aperfeiçoamento é diminuir a limitação do diagrama de associação de palavras, para que possa ser ampliado para além da associação em par com a criação de uma rede de palavras. Outro incremento a aplicação seria o estudo e, se possível, o desenvolvimento de uma funcionalidade para comparar dois artigos, gerando um texto sobre essa comparação. Ademais, a aplicação está disponível em https://aneshoji.shinyapps.io/textmining/ e o código-fonte no repositório https://github.com/AneShoji/wordHunter.

REFERENCES

[1] FAYYAD, U.; PIATESKY-SHAPIRO, G.; SMYTH, P. **Knowledge Discovery and Mining: Towards a UniFyng Framework**© 1996.

[2] SILGE, J.; ROBINSON, D. **Text Mining with R.** O'Reilly. 2022. E-book. Disponível em: https://www.tidytextmining.com/index.html. Acesso em:15 jun 2022.

[3] WICKHAM, H.; GROLEMUND, G. **R for Data Science**. O'Reilly. 2017. E-book. Disponível em: https://r4ds.had.co.nz/index.html. Acesso em: 20 jun 2022.

[4] WICKHAM, H.; NAVARRO, D.; PEDERSEN, T. L. **ggplot2: Elegant Graphics for data Analysys**, 3 ed. E-book. Disponível em: https://ggplot2-book.org/ Acesso em: 29 jun 2022.

[5] HADRIEN, H. **How to Build a Shiny Application from Scratch**. E-book. Disponível em: https://bookdown.org/hadrien/howtobuildashinyappfromscratch/. Acesso em: 05 ago 2022.

[6] HVITFELDT, E.; SILGE, J. **Supervised Machine Learning for Texttext Analysis in Rtext**. 2022. E-book. Disponível em: https://smltar.com/ Acesso em: 16 ago 2022.

[7] WICKHAM, H. **Mastering Shiny**. O'Really Media. 2022. Disponível em: https://mastering-shiny.org/index.html#license. Acesso em: 28 ago 2022

[8] WHITINGTON, J.; LAURENT, S. St **PDF Explained**. O'Reilly. 2011.

[9] R. Rafael A., **Introduction to Data Sciecne. Data Analysis and Prediction Algorithms with R.**. 2019. Disponível em: http://rafalab.dfci.harvard.edu/dsbook/

[10] **dplyr 1.0.10**. Disponível em: https://dplyr.tidyverse.org/index.html Acesso em: 24 set 2022.

[11] **RDocumentation**. Disponível em: https://www.rdocumentation.org/ Acesso em: 02 ago 2022.

[12] **Shiny**. Disponível em: https://shiny.rstudio.com/. Acesso em: 01 jun 2022.

[13] **R Package Documentation**. Disponível em: https://rdrr.io/. Acesso em: 26 ago 2022.

[14] **Hunspell** Disponível em: https://hunspell.github.io/ Acesso em: 28 out 2022.

[15] BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, B.; TURNER, A.; KHALIL, M. Lessons from applying the systematic literature review process within the software engineering domain. ***The Journal of Systems and Software***, 572-583, 17 aog 2006.

[16] HOTHO, A.; NÜRNBERGER, A.; PAASS, G. A Brief Survey of Text Mining. 1-37, 13 mai 2005.

[17] **Mineração de Texto Educacional**. Disponível em: http://sobek.ufrgs.br/#/ Acesso em: 29 out 2022.

[18] LIM, C.; CHO,G.; KIM, J. Understanding the linkages of smart-city technologies and applications: Key lessons from a text mining approach and a call for future research. ***Technological Forecasting and Social Change*** set,2021.

[19] KUSHWAHA, A. K.; KAR, A. K.; DWIVEDI, Y. Applications of big data in emerging management disciplines: A literature review using text mining. ***International Journal of Information Management Data Insights*** nov,2021.