TOBIAS RIBEIRO SOMBRA

Reconhecimento de padrões em rede social científica: aplicação do algoritmo Naive Bayes para classificação de *papers* no Mendeley

Dissertação de mestrado
Março de 2018

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO – UFRJ

ESCOLA DE COMUNICAÇÃO – ECO

INSTITUTO BRASILEIRO DE INFORMAÇÃO EM CIÊNCIA E TECNOLOGIA – IBICT

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA INFORMAÇÃO - PPGCI

TOBIAS RIBEIRO SOMBRA

RECONHECIMENTO DE PADRÕES EM REDE SOCIAL CIENTÍFICA: aplicação do algoritmo Naive Bayes para classificação de *papers* no Mendeley

Rio de Janeiro

2018

TOBIAS RIBEIRO SOMBRA

RECONHECIMENTO DE PADRÕES EM REDE SOCIAL CIENTÍFICA: aplicação do algoritmo Naive Bayes para classificação de *papers* no Mendeley

Dissertação de Mestrado apresentada ao programa de Pós-Gradação em Ciência da Informação, convênio entre o Instituto Brasileiro de Informação em Ciência e Tecnologia e a Universidade Federal do Rio de Janeiro/Escola de Comunicação, como requisito parcial à obtenção do título de Mestre em Ciência da Informação.

Orienradora: Prof. Dra. Rose Marie Santini

Coorientador: Prof. Dr. Emerson Cordeiro Morais

Rio de Janeiro

2018

CIP - Catalogação na Publicação

S693r

Sombra, Tobias
Reconhecimento de padrões em rede social científica: aplicação do algoritmo Naive Bayes para classificação de papers no Mendeley / Tobias Sombra. -- Rio de Janeiro, 2018.
198 f.

Orientadora: Prof. Dra. Rose Santini.
Coorientador: Prof. Dr. Emerson Morais.
Dissertação (mestrado) - Universidade Federal do Rio de Janeiro, Escola da Comunicação, Instituto Brasileiro de Informação em Ciência e Tecnologia, Programa de Pós-Graduação em Ciência da Informação, 2018.

1. Redes Sociais Científicas. 2. Mendeley. 3. Lógicas Sociais. 4. Inteligência Artificial. 5. Naive Bayes. I. Santini, Prof. Dra. Rose , orient. II. Morais, Prof. Dr. Emerson, coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a).

TOBIAS RIBEIRO SOMBRA

RECONHECIMENTO DE PADRÕES EM REDE SOCIAL CIENTÍFICA: aplicação do algoritmo Naive Bayes para classificação de *papers* no Mendeley

Dissertação de Mestrado apresentada ao programa de Pós-Gradação em Ciência da Informação, convênio entre o Instituto Brasileiro de Informação em Ciência e Tecnologia e a Universidade Federal do Rio de Janeiro/Escola de Comunicação, como requisito parcial à obtenção do título de Mestre em Ciência da Informação.

Aprovada em: 22 de março de 2018

_____

Profa. Dra. Rose Marie Santini (Orientadora)
PPGCI/IBICT-UFRJ/ECO

_____

Prof. Dr. Emerson Cordeiro Morais (Coorientador)
ICIBE/UFRA

_____

Prof. Dr Jorge Calmon de Almeida Biolchini (Membro interno)
PPGCI/IBICT-UFRJ/ECO

_____

Profa. Dra. Maria Luiza Machado Campos (Membro externo)
PESC/UFRJ

AGRADECIMENTOS

À **Deus**, que me deu forças para seguir em frente diante das dificuldades.

Aos meus **pais**, **familiares** e **amigos** que sempre me apoiaram e deram forças para que eu pudesse me animar e seguir minha jornada acadêmica.

Aos meus orientadores, professora **Rose Marie Santini** e Professor **Emerson Cordeiro Morais**. Graças a eles, aprendi bastante durante minha vida acadêmica. Agradeço também por aceitarem me orientar no desenvolvimento deste trabalho e por confiar em mim para concluí-lo. Agradeço de coração por todo o apoio e ajuda.

À todos os membros da banca, prof. **Jorge Biolchini**, prof**. Maria Luiza Campos**, prof. **Rosali Souza** e prof. **Sergio Serra**, por terem aceitado o convite para participar da defesa de mestrado. Agradeço muito pelas orientações que foram de grande ajuda para o desenvolvimento deste trabalho. Agradeço também a prof. **Jonice Oliveira** por ter trazido ótimas contribuições durante a qualificação desta dissertação.

À todos os alunos que conheci no curso de **Ciência da Informação** tanto do mestrado quanto do doutorado. Com vocês, vivenciei ótimos momentos quando estive no Rio de Janeiro. Levarei comigo a lembrança desse período maravilhoso que passei ao lado de vocês.

À todos os **professores** e **profissionais do IBICT**, que garantiram um curso de boa qualidade.

SOMBRA, T. **Reconhecimento de padrões em rede social científica: Aplicação do algoritmo Naive Bayes para classificação de *papers* no Mendeley**. 196 f. 2018. Dissertação (Mestrado em Ciência da Informação) – Universidade Federal do Rio de Janeiro, Instituto Brasileiro de Informação em Ciência e Tecnologia, Rio de Janeiro, 2018.

RESUMO

Este trabalho apresenta uma pesquisa exploratória usando o algoritmo Naive Bayes com capacidade para classificar documentos no Mendeley usando até cinco classes de saída, definidas com base na quantidade de leitores dos documentos. Usando uma série de atributos que foram encontrados durante a coleta de dados, é realizada a classificação para tentar identificar padrões nos atributos, a fim de reconhecer lógicas sociais dos cientistas, que envolve tanto o comportamento quanto sua dinâmica nas redes sociais científicas. Para fins de concretização deste trabalho, foi aplicada uma Revisão Sistemática de Literatura, a fim de buscar o estado da arte de pesquisas que envolvam o uso de Reconhecimento de Padrões em Redes Sociais Científicas, além da aplicação de um método que envolve o uso de algoritmos desenvolvidos para o tratamento automático de todos os dados coletados no Mendeley.

Palavras-Chave: Redes sociais científicas. Mendeley. Lógicas sociais. Naive Bayes. Inteligência artificial. Classificação. Ciência da Informação

SOMBRA, T. **Reconhecimento de padrões em rede social científica: Aplicação do algoritmo Naive Bayes para classificação de *papers* no Mendeley**. 196 f. 2018. Dissertação (Mestrado em Ciência da Informação) – Universidade Federal do Rio de Janeiro, Instituto Brasileiro de Informação em Ciência e Tecnologia, Rio de Janeiro, 2018.

ABSTRACT

This work is an exploratory research using the Naive Bayes algorithm with the ability to classify documents in Mendeley using the output classes, based on the amount of reading of the documents. Using a series of data that was found during a data collection, a classification is given to check the patterns in the attributes, an end to recognize the social logics of the scientists, that involve both the behavior and its dynamics in scientific social networks. For the purpose of this work, a literature systematic review was applied, with emphasis on the use of methods that involve the use of social networking concepts, as well as the application of a method for the use of algorithms. Created for automatic processing of all data collected at Mendeley.

Keywords: Scientific social networks. Mendeley. Social logics. Naive Bayes. Artificial intelligence. Classification. Information science.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
A&HCI	*Arts & Humanities Citation Index*
AHCI	*Arts and Humanities Citation Index*
AID	Análise Inteligente de Dados
AI	*Artificial Intelligence*
AM	Aprendizado de Máquina
AS	Aprendizagem Supervisionada
CC	Coeficiente de Correlação (do inglês *Correlation Coefficient*)
CPCI-SSH	*Conference Proceedings Citation Index - Social Science & Humanities*
CPCI-S	*Conference Proceedings Citation Index*
DCBD	Descoberta de Conheciento em Bases de Dados
IR	*Information Retrieval*
LISA	*Library and Information Science Abstracts*
LISTA	*Library, Information Science and Technology Abstracts*
MD	Mineração de Dados
ML	*Machine Learning*
MLP	Multilayer Perceptron
PECC	Porcentagem de Exemplos Corretamente Classificados
RG	*Research Gate*
RSL	Revisão Sistemática de Literatura
SCIE	*Science Citations Index Expanded*
SCI-EXPANDED	*Science Citation Index Expanded*
SSCI	*Social Sciences Citation Index*
SVM	*Support Vector Machine*
TC	Total de citações (do inglês *Total Citations*)
WoS	*Web Of Science*

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações Iniciais

No campo científico, é comum existirem revistas científicas. Essas revistas, ao longo dos anos, apresentaram recursos que foram modificados ao longo do tempo conforme o avanço tecnológico. Esses recursos envolvem, principalmente, a modificação das versões das revistas, que eram somente impressas e começaram a ganhar versões digitais. Os artigos científicos são um grande exemplo disso. A publicação digital de artigos veio, principalmente, devido à transição das revistas científicas para o meio digital. Além disso, muitas plataformas de redes sociais científicas foram desenvolvidas para criar um ambiente onde usuários podem publicar seus artigos, além de terem várias vantagens ao usá-las, como a capacidade de compartilhar informações de pesquisa e até mesmo de conhecer outros cientistas do mesmo ramo de estudo.

Todo esse processo de transição necessita também de meios que possam medir seu impacto. Por isso que métricas como a webometria e altmetria vão surgindo, para que apresentem possibilidades de medir o impacto científico para além das fronteiras da revista. Este trabalho tenta propor um pouco disso, mas usando técnicas de reconhecimento de padrões. A ideia não é exatamente medir o impacto científico, mas sim de tentar reconhecer lógicas nos atributos dos documentos que ajudem a identificar padrões sobre o comportamento dos cientistas e sua dinâmica nas redes sociais. Esse reconhecimento será feito pensando na popularidade dos documentos que, para este trabalho, será considerado os mais acessados (frequência). Podem existir casos de temas considerados populares que não são acessados. Por isso, a pesquisa apontará para popularidade de acesso independente do tema.

1.2 Objetivos

O principal objetivo do trabalho é reconhecer as lógicas nos documentos que auxiliem na identificação de padrões dos cientistas (que envolve comportamento e dinâmica nas redes sociais). Para que isso se concretize, têm-se, como objetivos específicos, um estudo a respeito das métricas que medem o impacto científico, um estudo envolvendo uma Revisão Sistemática com Reconhecimento de Padrões e Redes Sociais Científicas, a fim de buscar os estudos dessa área e o desenvolvimento de um método que garanta a coleta de dados no

Mendeley e todo o tratamento necessário para que os dados sejam executados no algoritmo Naive Bayes.

1.3 Justificativa

Como principal justificativa, têm-se a interdisciplinaridade entre Ciência da Computação e Ciência da Informação, usando o reconhecimento de padrões como um recurso para tentar resolver problemas de classificação em redes sociais científicas. A altmetria tem crescido como um novo indicador de produção científica devido, principalmente, ao crescimento do uso das redes sociais pelos cientistas. Esse crescimento vem, em boa parte, pela comunicação. Porém, muitas redes sociais, em particular as científicas, ganham crescimento também pela possibilidade de publicação de artigos. É baseado na publicação dos artigos que a aplicação do Reconhecimento de Padrões será feita.

1.4 Questão de Pesquisa

Como questão principal de pesquisa, tem-se a seguinte pergunta: Como a Inteligência Artificial, através do Reconhecimento de Padrões, pode ajudar a identificar as novas lógicas sociais características do campo científico?

Para auxiliar a questão principal da pesquisa, têm-se as seguintes perguntas específicas.

1 – É possível reconhecer padrões trabalhando com dados coletados cujo nome de publicação contém as palavras *Proceedings* e *Journal* no Mendeley?

2 – É possível, por meio do reconhecimento de padrões, gerar conhecimento sobre esses dados coletados no Mendeley?

3 – De que forma esse processo deve ser realizado para que se consiga obter resultados no Mendeley usando o algoritmo Naive Bayes?

Nesse sentido, percebe-se que se pretende coletar o universo de documentos cujo nome de publicação no Mendeley contenha as palavras *Proceedings* e *Journal*. Com isso, eles serão passados por um processo de tratamento de dados para que estejam adequados para receberem a classificação do algoritmo Naive Bayes, a fim de descobrir o que resultará disso.

1.5 Hipótese

Baseando-se nas questões de pesquisa apresentadas na Seção anterior, acredita-se que a aplicação da inteligência artificial em redes sociais científicas pode ser uma solução para o

reconhecimento de padrões sobre o comportamento dos pesquisadores, além de trazer a possibilidade de revelar suas dinâmicas nessas redes.

1.6 Objeto de Estudo e Métodos Utilizados

Para este trabalho, será usado o Mendeley como objeto de estudo. Mendeley é uma plataforma de redes sociais científicas que também funciona como gestor de conteúdo. Nele, existe a possibilidade de encontrar diversos artigos de várias áreas do conhecimento. O Mendeley também apresenta uma api para desenvolvedores produzirem algoritmos que se conectam com o Mendeley para fazer consultas de dados. Como procedimentos metodológicos, são utilizados vários, como o estudo de uma revisão sistemática de literatura envolvendo redes sociais científicas e reconhecimento de padrões e o uso de técnicas de aprendizado de máquina aplicando o algoritmo Naive Bayes para classificar os documentos coletados no Mendeley.

1.7 Organização do Trabalho

Como forma de atender aos objetivos propostos na Seção 1.2, o trabalho foi dividido em seis capítulos, incluindo a introdução. A organização do trabalho segue da seguinte maneira:

No Capítulo 2, será apresentado uma revisão de literatura sobre as métricas, discutindo o surgimento delas e sua importância para a ciência, uma descrição do Mendeley e como surgiu e uma aplicação de Revisão Sistemática de Literatura sobre Reconhecimento de Padrões em Redes Sociais Científicas.

No Capítulo 3, serão descritos alguns conceitos básicos de aprendizado de máquina importantes para a compreensão do método utilizado nesta dissertação, além de descrever a medida de erro usada para problemas de classificação e o método de amostragem *Holdout*. Por fim, será abordado sobre o algoritmo Naive Bayes, explicando seu funcionamento através de exemplos.

No Capítulo 4, será apresentado o método com base no Reconhecimento de Padrões utilizado nesse trabalho. O método é dividido na etapa de coleta, pré-adaptação e mineração. Será mostrado todo o processo de preparação dos dados para posterior classificação no algoritmo Naive Bayes.

No Capítulo 5, serão apresentados os resultados obtidos pelo algoritmo Naive Bayes, além de uma discussão e análise qualitativa dos resultados. Apresentam também os desafios e limitações encontrados no trabalho.

No Capítulo 6, serão apresentadas as conclusões deste trabalho, bem como propostas para sua continuação.

2 RECONHECIMENTO DE PADRÕES EM REDES SOCIAIS CIENTÍFICAS: UMA REVISÃO DE LITERATURA

O presente capítulo apresenta uma revisão de literatura sobre o conceito de Altmetria: problema de pesquisa, como surgiu, vantagens e desvantagens e os principais trabalhos publicados sobre o tema. Além disso, este Capítulo apresenta um breve histórico sobre a transição da Bibliometria para a Altmetria e a necessidade de uso desta métrica tendo em vista o grande crescimento das Redes Sociais. Também apresenta-se uma descrição do Mendeley e como surgiu, além da revisão de alguns artigos que usam o Mendeley como ferramenta para tratar alguns problemas da Altmetria, mostrando que essa métrica pode ser uma alternativa para avaliação de citações. Ao final, discutimos porquê essa métrica deve ser sempre considerada com cautela, devido a algumas lacunas metodológicas, ainda em processo de consolidação.

2.1 Altmetria: Novos indicadores para o impacto científico

Ao longo dos anos, a comunidade científica desenvolveu várias métricas que trazem indicadores para avaliar o impacto científico. Dentre elas, pode-se destacar a Bibliometria, Webometria e, inclusive, a Altmetria, esta que é uma métrica bastante recente e que demanda por pesquisas para se consolidar cada vez mais.

Segundo Fenner (2014, p. 179):

> A avaliação de impacto dos pesquisadores e suas pesquisas são fundamentais para a comunicação acadêmica. Nos últimos 25 anos, vimos uma mudança da avaliação qualitativa individual por pares para avaliação quantitativa sistemática usando a análise de citações de artigos e revistas. Provavelmente, o impacto da pesquisa não pode ser quantificado, e a análise de citações é insuficiente para uma análise abrangente, mas a revista como um filtro para o conteúdo acadêmico relevante e o fator de impacto para quantificar a relevância das revistas são o cerne de como a pesquisa é comunicada e avaliada hoje[1]. (FENNER 2014, p. 179).

Como é possível verificar, a avaliação de impacto torna-se fundamental para a comunicação científica. Porém, a análise quantitativa e baseada somente em citações é insuficiente para avaliar o impacto, principalmente com a mudança das revistas impressas para eletrônicas. Fenner (2014) também afirma que, com esta transição das revistas científicas para a web, outras medidas de impacto podem ser utilizadas. Neste sentido, não é mais

[1] Esta e as demais citações que aparecerem neste e demais Capítulos foram traduzidas pelo autor desta dissertação.

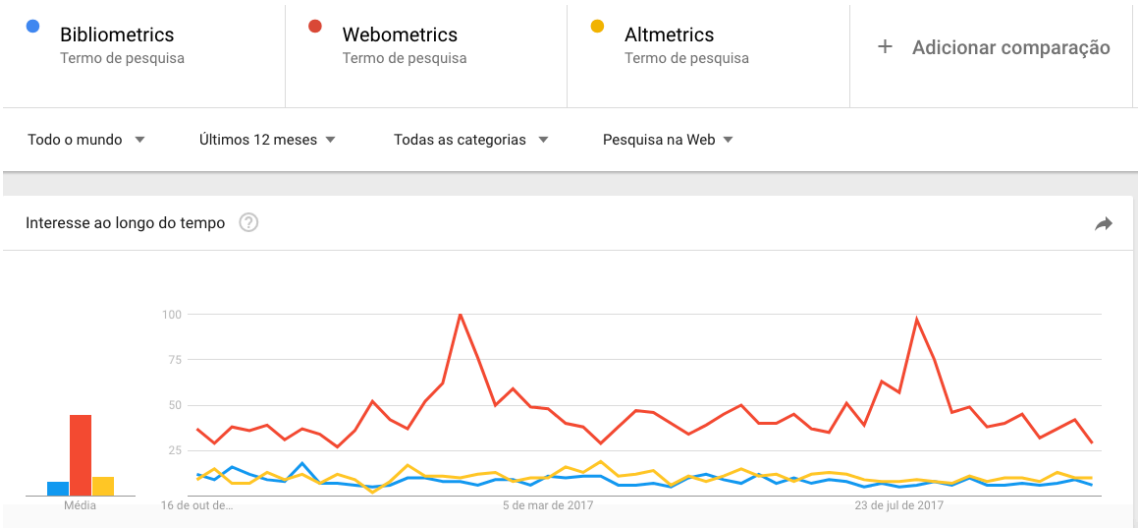
possível limitar-se somente aos artigos das revistas, visto que os resultados de uma pesquisa, como a publicação de dados, também podem ser incluídos para avaliação.

A chegada da Altmetria, que veio depois da bibliometria e webometria como uma nova métrica para medir o impacto acadêmico surgiu por meio de diversas demandas dentro da comunidade científica. Essas demandas foram supridas, ao longo dos anos, por outras métricas, como a Bibliometria e a Webometria. O surgimento da Altmetria não significa que outras métricas estão obsoletas, pois ainda são publicados estudos envolvendo essas métricas. Como exemplo, pode-se citar pesquisas de 2017 tanto para Bibliometria quanto para Webometria. Schneider *et. al.* (2017), apresentam um trabalho com o uso da Bibliometria para estimar a produtividade e influência utilizando dados de publicações da área de Ciências Clínicas. Damayanti *et. al.* (2017) trabalharam com estratégias para aumentar a visibilidade, impacto e atividade do site stikon.edu no rank de Webometria. Leung *et. al.* (2017), usaram a Bibliometria para analisar a evolução dos temas "negócios" e "hospitalidade/turismo" ao longo do tempo, além de aplicar uma Revisão Sistemática e Holística da literatura relacionada as mídias sociais dentro dos temas analisados. Verma e Brahma (2017) realizaram uma análise webométrica dos sites das bibliotecas nacionais do sul da Ásia para avaliar o fator de impacto dessas bibliotecas[2].

Ao realizar uma breve pesquisa sobre a tendência dos três temas como tópico de pesquisa no Google, a Webometria aparenta liderar nas pesquisas, enquanto que Altmetria e Bibliometria estão praticamente com a mesma média, sendo que a curva dos gráficos apontam um interesse relativamente maior em Altmetria do que Bibliometria no momento. A Figura 1 apresenta o gráfico gerado no Google Trends usando *Bibliometrics*, *Webometrics* e *Altmetrics* como termos de busca baseando-se no mundo todo em um período de 12 meses. É importante informar que este gráfico é apenas ilustrativo e não tem grande representatividade, pois, para isso, seria necessária uma pesquisa ainda mais ampla para avaliar o interesse dos tópicos ao longo do tempo. Porém, usando o Google Trends é possível ter uma ideia deste interesse. A imagem foi gerada dia 15/10/2017.

[2] Os artigos foram encontrados realizando uma breve pesquisa no Google *Scholar* usando os termos *Bibliometrics, Webometrics, Bibliometrics Study* e *Webometrics Study* no ano de 2017.

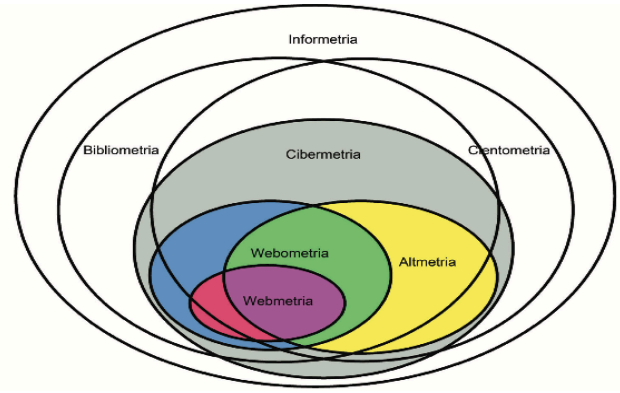
Figura 1 - Interesse dos tópicos *Bibliometrics*, *Webometrics* e *Altmetrics* usando o Google Trends nos últimos 12 meses.



Fonte: Google Trends

As métricas aqui apresentadas estão envolvidas em disciplinas. Fenner (2014) afirma que tanto Bibliometria quanto Altmetria são subdisciplinas da Cientometria, campo científico que mede e analisa a ciência. Björneborn e Ingwersen (2004) apud Gouveia e Lang (2013, p. 174), afirmam que a Webometria está contida na Bibliometria e ambas as métricas estão contidas na Infometria, que é considerada o grande campo do conhecimento. Gouveia (2013) apresenta uma interface onde tenta inserir as métricas em subdisciplinas de outras, sendo que a Infometria é considerada a maior delas. A Figura 2 apresenta esta relação.

Figura 2 - Interfaces entre os campos da Webometria, Webmetria, Altmetria e Cibermetria com a Bibliometria, Cientometria e infometria.



Fonte: Gouveia (2013)

Como foi dito anteriormente, a Altmetria surgiu por meio de demandas dentro da

comunidade científica. Para entender esse surgimento, é necessário um breve estudo apontando a transição e mudança de métricas como Bibliometria e Webometria para Altmetria, com o intuito de identificar o motivo do surgimento desta métrica.

A Bibliometria, segundo Fenner (2014, p. 180), "é uma grande subdisciplina da cientometria que mede o impacto das publicações científicas. A análise de citações é a aplicação mais popular de bibliometria". Neste caso, a bibliometria é uma métrica que busca avaliar o impacto, principalmente, por meio de citações.

A Webometria, segundo Björneborn (2004, p. 1217) é "o estudo dos aspectos quantitativos da construção e uso de recursos de informação, estruturas e tecnologias na web, se aproximando de abordagens bibliométricas e infométricas". O autor também afirma que o interesse pelas redes de citações e a hiper-estrutura de textos na web gerou pesquisas em meados da década de 90 e este interesse foi crescendo com o passar do tempo. Com relação a produção no campo da Webometria, Gouveia e Lang (2013) fizeram uma pesquisa na *Web Of Science* usando o período de 1997 até 2012 e foram publicados 211 trabalhos com os termos *webometrics* e *webometry*. Nos períodos de 1997 a 2002 foram encontrados poucos trabalhos na área. Porém, em 2003 houve um crescimento brusco de pesquisas envolvendo Webometria. As pesquisas encontradas pelos autores tiveram como principais áreas do conhecimento a Ciência da Informação e Computação (172 pesquisas para a primeira e 131 pesquisas para a segunda). Com relação ao idioma dos trabalhos, há uma predominância do inglês, com 194 publicações, seguido por espanhol, com 14 publicações e o português, com apenas 3 publicações. Os autores também analisaram o número de publicações por país e a Inglaterra apresentou o maior número (55) seguido da Espanha (39). O Brasil ficou em oitavo lugar, juntamente com Bélgica e Holanda, apresentando 8 publicações.

A transição da Webometria para a Altmetria veio de uma necessidade de uso de novas métricas. Segundo Butler (2017), "[...] a explosão das mídias sociais, juntamente com o desenvolvimento de sites e blogs populares profissionais e científicos, levou a necessidade de métricas alternativas, conhecida como Altmetria, para quantificar um impacto mais amplo da pesquisa". Neste caso, a Altmetria surgiu da necessidade de buscar outras métricas, levando em consideração o grande crescimento das mídias sociais.

Butler (2017) tem a seguinte definição de Altmetria:

> A Altmetria usa métricas baseadas na web para avaliar o maior impacto do material acadêmico, com ênfase nas mídias sociais como fontes de dados. O termo *article-level metrics* refere-se aos tipos de dados coletados, que incluem visualizações, downloads, cliques, notas, tweets, compartilhamentos, recomendações, *tags*, postagens, *trackbacks*, discussões, marcadores e comentários; não apenas citações de um artigo em um banco de dados ou por um editor. Butler (2017, p. 226).

Com isso, é possível perceber que a Altmetria busca ampliar o campo de pesquisa para avaliar o impacto. Segundo Priem et. al (2010), a Altmetria apresenta mais indicadores que as métricas tradicionais. A Altmetria busca o impacto não somente em citações, mas também em comentários de redes sociais, blogs, etc.

Algumas pesquisas sugerem que o termo Altmetria ainda é bastante questionado no âmbito acadêmico. Sugimoto (2017) realizou uma revisão sistemática envolvendo a Altmetria e mídias sociais e descobriu que o termo "alt" em Altmetria é bem discutido, levando em consideração que muitos estudos mostram que os indicadores de Altmetria podem ser considerados complementares e não como alternativa às citações. Porém, outros autores, segundo a Revisão Sistemática de Sugimoto (2017), mostram que a Altmetria fornece "diferentes abordagens para diferentes perguntas".

Butler (2017) apresenta algumas vantagens e desvantagens da Altmetria. Como vantagens, destaca-se:

- Descrições mais transparentes sobre o interesse, uso e alcance dos produtos acadêmicos;
- Envolve uma analise de impacto mais diversificada do que as métricas tradicionais;
- Acessa um público mais amplo, como profissionais, estudantes de graduação, funcionários governamentais e público de interesse em geral;
- Pode medir o fluxo de pesquisa na sociedade, que é objetivo para políticos, organizações de pesquisa e de financiadores;
- Pode medir impactos de maior alcance, como efeitos na prática clínica, aplicações técnicas, educação e políticas de saúde;
- Possibilidade de medir o impacto de produtos acadêmicos mais diversos, como conjunto de dados, software, direitos autorais, algoritmos, literatura cinzenta[3] e slides;
- Examina o impacto dos produtos acadêmicos na ciência e além do campo científico;
- Pode acompanhar uma variedade de atividades acadêmicas, como ensino e serviço;
- Tem velocidade significativa, permitindo que o impacto imediato da pesquisa seja medido logo após a publicação;

Como desvantagens da Altmetria, segundo Butler (2017), pontua-se:

[3] Literatura cinzenta são publicações não-comerciais, difíceis de encontrar em canais tradicionais de distribuição, como resultados de reuniões científicas ou não, folhetos das mais diversas procedências e assuntos, relatórios e anais de conferências, teses, publicações oficiais, pré-publicações, entre outros, que muitas vezes são fundamentais para bibliotecas especializadas e universalizadas. ANDRADE; VERGUEIRO (1996, p. 66).

- Não há estatísticas de usuário precisas ou descrições de amostra para plataformas de mídias sociais e, portanto, esse viés não pode ser quantificado;
- Não existe o impacto exato nos documentos de governo ou em sites de comentários em redes sociais;
- As publicações podem levar a ambiguidade, levando em consideração que elas existem em várias versões diferentes;
- As citações podem ter simples menções ou extensas discussões de um artigo citado. Podem ser também uma discussão muito técnica e detalhada. Portanto, é desejável ter diferentes significados levados em consideração;
- Todo o cientista sabe que está sendo medido por contagens de citações (ou documentos publicados posteriormente). No entanto, muitas vezes não está claro o que está sendo medido na Altmetria, mesmo que a fonte métrica seja a mesma. Os números reais podem se referir a diferentes formas de engajamento;
- Na bibliometria tradicional, regras precisas indicam quando, onde e em que documentos são citados. Nenhuma regra semelhante aplica-se às plataformas de redes sociais;
- Há uma escassez de estudos empíricos sofisticados em grande escala para sustentar a Altmetria. Não existem evidências sistemáticas sobre a confiabilidade, validade e contexto em dados altmétricos e muitos estudos de dados da Altmetria apresentaram erros devido à inadequação da metodologia de pesquisa;
- É muito fácil manipular dados de Altmetria. Nas métricas tradicionais, os periódicos podem aumentar o impacto com citações em editoriais, ou é possível criar citações com documentos falsos.
- É possível gerar sistematicamente pontuações de Altmetria altas para qualquer pesquisador ou conjunto de artigos, como menções do Twitter geradas através de contas falsas e o uso de "*Robot Tweeting*"[4].

Existem estudos onde autores tentam mostrar o interesse da Altmetria para os cientistas. Valiente et. al (2016) realizaram uma pesquisa que analisa a produção científica de publicações em Altmetria e seus resultados foram interessantes. A Altmetria, no período de 2005 a 2010 teve um crescimento pequeno de artigos, com média de 2,1 artigos por ano. As publicações sofreram aumento após 2011, com média de 48 artigos. Com relação a

[4] É uma expressão usada para definir contas falsas que usam de robôs para fazer publicações no Twitter.

produtividade dos autores, Valiente et. al (2016) identificaram 447 autores, dos quais 56 publicaram dois ou mais artigos. Muitos desses artigos foram publicados em revista (212) e *conference proceedings* (26), enquanto que são menos frequentes em séries (7), livros (4) e seções de livros (4).

Com relação aos países onde os artigos são publicados por revistas nos resultados de Valiente et. al (2016), encontram-se um total de 26, se destacando o Reino Unido (34), Estados Unidos (28) e os Países Baixos (10) e, em menor grau, Alemanha (4), Índia (4), Canadá (3) e Croácia (3). Os periódicos correspondem a 26 áreas da Scopus, onde as Ciências Sociais (48) se destacam, seguidas pela Medicina (32), Ciência da Computação (24) e Negócios, Gestão e Contabilidade (11). Em termos de idioma dos documentos, 12 foram identificados no trabalho de Valiente et. al (2016), onde o inglês (237) tem dominância, seguido por espanhol (4), persa (2), português (2), árabe (1), bósnio (1), chinês (1), alemão (1), húngaro (1), italiano (1) e japonês (1).

Valiente et. al (2016) identificaram um total de 1014 palavras-chave, das quais apenas 224 ocorreram. Os termos com maior frequência foram *altmetrics* (122), *Social Media* (55), *Bibliometrics* (50), *Article* (29), *Social Networks (online)* (23), *publication* (27), *Citation Analysis* (25), *Human* (23), *Journal Impact Factor* (20), Twitter (20), *Impact Factor* (20), Internet (18), *Research* (18), *Research Evaluation* (17), *Medic Research* (16) e Mendeley (15).

Com base em todas as definições, pesquisas, vantagens e desvantagens da Altmetria discutidas nesta Seção, percebe-se que esta métrica é recente, que demanda por agenda de pesquisa e que precisa se consolidar empiricamente. A Altmetria é cercada por dificuldades e desafios, levando em consideração o alto avanço das mídias sociais e, junto a elas, o uso de ferramentas que permitem "burlar" a avaliação da Altmetria por meio de comentários em documentos falsos ou uso de *robots* em redes sociais (como afirma Butler (2017) em vantagens e desvantagens da Altmetria). Apesar dessas dificuldades serem persistentes na Altmetria, o fato dela buscar métricas alternativas se faz importante, levando em consideração que os artigos científicos estão tomando rumos que vão além das citações, uma vez que as revistas científicas passaram por uma transição de impressas para a web, como afirma Fenner (2014), pontuado no início deste Capítulo.

Levando em consideração os problemas da Altmetria, como a incerteza dos dados coletados, é necessário, no momento, não considerar os resultados da Altmetria diretamente para tomadas de decisão. Segundo Butler (2017), "Os resultados baseados em Altmetria não devem levar a uma tomada de decisão direta sobre a pesquisa, mas, em vez disso, devem ser

usados para auxiliar os especialistas na tomada de decisões". A próxima Seção irá apresentar a plataforma Mendeley, que é um gerenciador de referência e que foi, inclusive, uma das palavras-chave que sofreu ocorrência no trabalho de Valiente et. al (2016). A Seção apresentará uma discussão sobre Redes Sociais Científicas, uma explicação sobre o Mendeley, como surgiu e uma revisão de literatura sobre o Mendeley.

2.2 Redes Sociais Científicas: O surgimento do Mendeley

Mendeley é uma plataforma que apresenta um site e um aplicativo para computadores e dispositivos móveis, que permite armazenar e gerenciar arquivos PDF e citações, mantendo-os sincronizados através da computação em nuvem. Mendeley também permite que sejam compartilhados documentos com outras pessoas, como se fosse uma rede social (RUSSO et. al. 2013).

Segundo a revista *Eletronic Resources Review*, o modelo inicial do Mendeley foi uma popular rede de músicas, chamada Last.fm. Os usuários da Last.fm podem compartilhar suas playlists de música e se beneficia ao ter acesso aos padrões de escuta de milhares de usuários para descobrir novos artistas e músicas. Neste sentido, a ideia de compartilhamento de conhecimento, proposta na Last.fm, foi transposta ao Mendeley enquanto compartilhamento de documentos entre os usuários.

Pattillo (2010) apresenta alguns recursos que o Mendeley pode oferecer:

- Adicionar documentos a base de dados através de diferentes fontes, como base de dados online (PubMed, OvidSP, SpringerLink, ScienceDirect, CiteULike, *Google Scholar*, entre outros);
- Possui um Plugin que importa documentos da web para a biblioteca do Mendeley;
- Possibilidade de visualizar os artigos (*papers)* em multitarefa, e é possível criar notas, compartilhar arquivos, salva-los ou imprimi-los;
- Possui capacidade de armazenamento em nuvem gratuita de 1 GB, permitindo fazer backup e sincronizar automaticamente a biblioteca Mendeley para vários dispositivos, sejam eles o aplicativo para Desktop, web ou dispositivos móveis, possibilitando uma alta portabilidade;
- Funciona como um gerenciador de referência gratuito (como o endnote), possibilitando criar ou escolher estilos de citação de um banco de dados enorme e constantemente atualizado para criar rapidamente uma biblioteca automática, economizando tempo;

- Permite o compartilhamento de bibliografias em grupos particulares, onde qualquer membro pode adicionar e editar citações;
- Possibilita a criação de grupos de trabalho, públicos ou privados, simplificando o compartilhamento instantâneo de um documento. Todos os membros do grupo podem ver quando outros adicionam documentos, comentam ou compartilham discussões;
- É possível criar um programa acadêmico, fazer upload de publicações e fazer pesquisas. Desta forma, é fácil inserir *papers* em grupos públicos e também se conectar com outros usuários, criando uma rede de acordo com as necessidades da pesquisa;
- As publicações enviadas ao Mendeley são indexadas na plataforma de acordo com a popularidade de leitores-usuários.

Segundo a equipe do Mendeley, em 2013, a plataforma já tinha mais de 2.5 milhões[5] de usuários. Isso pode ser um indício de que o Mendeley foi bem aceito pela comunidade científica, além deste feito ser uma verdadeira conquista para a plataforma.

Ao longo dos anos, diversos trabalhos científicos sobre o Mendeley foram publicados. Pode-se citar trabalhos como: 1) uso de marcadores do Mendeley como uma pesquisa que reflete motivações do usuário (MOHAMADI, 2015); 2) busca de correlações entre número de citações e quantidade de leitores no Mendeley nas áreas de Ciências Sociais e Humanas, (MOHAMADI, THELWALL, 2014); 3) no campo médico (THELWALL, WILSON, 2015); 4) entre categorias de usuário (MOHAMADI *et. al.*, 2015); 5) e ao longo do tempo (THELWALL, SUD, 2015).

O primeiro artigo coletou os dados usando uma amostra imparcial de usuários do Mendeley e fez uma lista como ponto de partida. Com essa lista, foi enviado o questionário aos usuários para eles preencherem e, assim, coletar os dados. Seus resultados foram divididos em 4 etapas: ocupação e disciplina dos usuários, motivações para usar o Mendeley, motivações dos marcadores de *papers* nas bibliotecas pessoais, diferenças de ocupação e leitura de publicações marcadas. Mohamadi (2015) conseguiu obter o seguinte resultado: a maioria dos usuários entrevistados eram estudantes de doutorado (27%), seguido por pesquisadores de pós-doutorado (26%). Estudantes de graduação tinham a menor quantidade na amostra (1%). Cerca de 78% dos usuários entrevistados tinham bibliotecas Mendeley e a maioria deles informou que usam Mendeley como gerenciador de referência (87%), como

[5] https://blog.mendeley.com/2013/09/03/mendeley-has-2-5-million-users/. Acesso em: 12/11/2017.

base de dados para pesquisas acadêmicas (30%), como ferramenta para divulgar suas publicações (30%) e site de redes sociais (15%). A maioria dos usuários entrevistados da área de ciências médicas (63%) e engenharia (54%) afirmam marcar registros acadêmicos para uso profissional, mas a maioria dos usuários em ciência básica (49%), artes e humanidades (48%) e ciências sociais (43%) não fazem isso. Cerca de 85% dos alunos de doutorado e mestrado adicionam documentos nas suas bibliotecas do Mendeley para citarem em suas teses e dissertações. Cerca de 27% dos usuários com biblioteca Mendeley leram ou pretendiam ler todos os registros marcados, 55% leram ou pretendiam ler pelo menos metade e 18% leram ou pretendiam ler metade dos itens e apenas 0,4% dos entrevistados não haviam lido nenhum dos registros marcados e não pretendiam lê-los. Mohamadi (2015) sugeriu que as contagens de marcadores do Mendeley são um indicativo de público porque a maioria dos registros marcados pelos entrevistados foram lidos e quase todos os resultados apresentados foram amplamente consistentes com o que se sabe sobre por que os acadêmicos leem artigos, dando mais evidências para o valor da contagem do Mendeley.

O segundo artigo de Mohamadi e Thelwall (2014) intitulado *Mendeley Readership Altmetrics for the Social Sciences and Humanities: Research Evaluation and Knowledge Flows*, utilizaram as opções avançadas de pesquisa da WoS (*Web Of Science*) para recuperar todas as citações de documentos das áreas de ciências sociais e humanidades. Os resultados limitaram-se a artigos de pesquisa em inglês e selecionaram o ano de 2008 para a coleta, por que o período de pico para as citações é geralmente três anos após a publicação de um artigo. Mohamadi e Thelwall (2014) usaram a opção de "analisar resultado" e depois "áreas de pesquisa", para classificar os registros recuperados nas disciplinas de Ciências Sociais e Humanas na WoS. O procedimento de coleta foi essencial para poder fazer a correlação entre as citações da WoS com as contagens de leitores do Mendeley. Como resultado, Mohamadi e Thelwall (2014) encontraram uma correlação entre citações na WoS com a quantidade e leitores do Mendeley para as disciplinas das Ciências Sociais maior que das disciplinas de humanidades. As Ciências Sociais, em geral, apresentaram correlações médias, enquanto que nas disciplinas humanas, religião e filosofia apresentaram menor correlação e linguística apresentou maior correlação entre as disciplinas humanas. Mohamadi e Thelwall (2014) analisaram também o fluxo de conhecimento se baseando nos leitores do Mendeley e identificaram que artigos de Psicologia têm leitores de Artes e Humanidades, embora algumas publicações de Psicologia sejam lidas por pessoas de Biologia (7%) e Medicina (6%) devido a, talvez, essas disciplinas apresentarem interseções com a Psicologia, a Neuropsicologia e a Psicofarmacologia. Entretanto, existem algumas diferenças em termos dos pontos fortes das

conexões entre as disciplinas. Mohamadi e Thelwall (2014) citam como exemplo que os dados do Mendeley trazem relações mais fortes entre História, Psicologia e Ciência da Computação, enquanto que os dados de citações proporcionam relações mais fortes entre História, Medicina e Biologia. Mohamadi e Thelwall (2014) sugerem que os dados de leitura no Mendeley podem ser uma medida complementar útil para remediar algumas limitações da análise de citações em todas as Ciências Sociais e Humanas. O Mendeley pode ser universalmente útil para estimar impacto de pesquisa em todas as áreas de estudos devido a sua vantagem de abranger vários tipos de usuários, como estudantes de graduação e pós-graduação, enquanto que os dados de citação são apenas dos autores. Mohamadi e Thelwall (2014) também indicam, com base nos achados da pesquisa, que os dados do Mendeley podem ser particularmente úteis quando os fluxos de informações de curto prazo são necessários ou quando precisa de uma perspectiva mais ampla sobre o impacto de um documento do que apenas a publicação das citações.

O terceiro artigo de Thelwall e Wilson (2015), cujo título é *Mendeley Readership Altmetrics for Medical Articles: An Analysis of 45 Fields*, tentaram buscar correlações entre número de citações e quantidade de leitores do Mendeley no campo médico. A coleta de dados foi realizada na base de dados Scopus usando o período de 2009, com os 47 campos da área de medicina que existem na base. Todos os artigos coletados foram submetidos a Mendeley API v. 1 (lançada em 24 de setembro de 2014), via *Webometric Analyst* entre 11 e 15 de novembro de 2014, para contar o número de usuários em Mendeley que registraram o artigo. Os artigos foram encontrados no Mendeley com uma pesquisa baseada no ano de publicação, o primeiro e último nome do autor e o título. Com isso, Thelwall e Wilson (2015) encontraram o seguinte: as correlações entre os leitores e citações, que foram significativamente positivas e fortes para quase todas as áreas temáticas, exceto a categoria Guias de Drogas, que tem uma correlação positiva, mas não tão forte como as outras. A baixa correlação neste grupo, segundo Thelwall e Wilson (2015), pode ser devido ao tamanho pequeno e natureza incomum da área. Os autores consideraram razoável alegar que o contador de leitor do Mendeley se correlaciona fortemente com as contagens de citações da Scopus em campos médicos. Além disso, as correlações entre citações e leitores diminui quando ambas as classes identificadas de leitores estudantis (bacharel e mestre) são removidas. Com isso, Thelwall e Wilson (2015) sugerem que os resultados provam que a contagem de leitores do Mendeley podem ser uma boa alternativa para a contagem de citação para todos os campos de pesquisa médica, exceto para guia de drogas. No entanto, recomenda-se que as contagens de leitores sejam usadas para avaliações de pesquisa em fase inicial e para outros tipos de

avaliação e aplicações de pesquisa, mas apenas quando a manipulação das informações (limitação da Altmetria) for improvável de ser um problema e quando as distorções de Mendeley pareçam improváveis de afetar os conjuntos de artigos a serem comparados.

O quarto artigo de Mohamadi *et. al.* (2015) cujo título é *Who Reads Research Articles? An Altmetrics Analysis of Mendeley User Categories*, buscaram correlações entre citações e quantidade de leitores do Mendeley entre categorias de usuários. A coleta de dados foi realizada na base WoS usando o período de 2008. Foi excluído os tipos de documentos não eletrônicos, como editoriais e revisões de livros. Os dados de citação provêm do *Science Citations Index Expanded* (SCIE), do *Arts and Humanities Citation Index* (AHCI) e do *Social Sciences Citation Index* (SSCI) em dezembro de 2012. O ano de 2008 foi selecionado para permitir que todos os artigos tenham pelo menos 4 anos para receber citações. Algumas das 15 categorias ocupacionais dos autores relatadas por Mendeley são semelhantes e foram incorporadas em uma única categoria. Como exemplo, Mohamadi *et. al.* (2015) informaram que alunos de pós-graduação e mestrandos foram incorporados em uma única categoria de estudantes de pós-graduação. Com as categorias selecionadas e os dados coletados, começou-se a buscar as correlações entre citações e quantidade de leitores do Mendeley. Os autores dividiram os resultados em 3 categorias distintas, chamadas de: Todos os Artigos, Artigos com pelo menos 66% de leitores e Artigos com contagem de 100% de leitores. Com isso, Mohamadi *et. al.* (2015) mostraram que estudantes de doutorado foram os principais leitores de artigos Mendeley em 2008 para as categorias "Todos os Artigos" (de todos os artigos do Mendeley, a maioria foi lido por estudantes de doutorado), "Artigos com pelo menos 66% de leitores" (em artigos com até 66% de leitores do Mendeley, a maioria foi lido por estudantes de doutorado) e "Artigos com contagem de 100% de leitores" (artigos onde todos ou praticamente todos os leitores de Mendeley visualizaram ou leram, a maioria foi de estudantes de doutorado), embora as porcentagens variem em diferentes disciplinas. Estudantes de pós-graduação e pós-doutores foram os principais leitores após estudantes de doutorado em diferentes disciplinas. Da área de Medicina Clínica na categoria Outras Profissões, cerca de 7.2% foram da classe "Artigos com 100% de leitores", 5.9% foram de "Artigos com pelo menos 66% de leitores" e 5.6% foram da categoria "Todos os Artigos". Os bibliotecários eram 3,7%, 2,8% e 2,5% dos leitores relatados de artigos de Ciências Sociais, mas eram leitores menos comuns em artigos de outras disciplinas. As correlações entre o contador de leitores do Mendeley e citações baseadas nas profissões dos usuários indicaram que todas as correlações são mais baixas para categoria "Artigos com 100% de leitores". Geralmente, as maiores correlações são para professores titulares, professores assistentes, pós-doutores e

estudantes de doutorado, enquanto as mais baixas são para estudantes de graduação, outras profissões e bibliotecários em todas as disciplinas e em todos os três conjuntos de dados (100%, 66% e todos os artigos). No entanto, as correlações para outras profissões são mais altas para a Medicina Clínica do que para outras disciplinas. Com isso, Mohamadi *et. al.* (2015) sugerem que o Mendeley fornece evidências de que os artigos de pesquisa são lidos por uma variedade de tipos de usuários dentro e fora da academia. A maioria dos leitores de Mendeley são estudantes de doutorado em Engenharia e Tecnologia, Ciências Sociais, Física e Química, com professores sendo uma minoria em todos os casos. Mohamadi *et. al.* (2015) também concluem que a contagem de leitores do Mendeley dependem das ocupações dos leitores, indicando que a leitura no Mendeley pode refletir o impacto da citação tradicional, mas em outros casos podem refletir os usos educacionais ou o impacto nos contextos educacionais. Portanto, o leitor do Mendeley é uma fonte de dados promissora que é diferente das citações e dos dados de uso bruto, segundo os autores. Além disso, Mohamadi *et. al.* (2015) acreditam que Mendeley parece ser a única opção para revelar aspectos de leitura de artigos de pesquisa. Isso pode ser útil em disciplinas para as quais os indicadores baseados em citações são menos confiáveis, como nas Ciências Sociais, Artes e Humanidades e talvez também à pesquisa aplicada.

O quinto artigo de Thelwall e Sud (2015), intitulado *Mendeley Readership Counts: An Investigation of Temporal and Disciplinary Differences,* tentaram verificar a correlação entre citações e quantidade de leitores ao longo do tempo. Para isso, foram obtidos artigos no período de 2004 até o final de 2014 na base de dados Scopus divididos em cinco categorias (agricultura, negócios, ciência da decisão, farmácia e ciências sociais) e 50 subcategorias. As categorias da Scopus foram utilizadas como fonte básica de delimitação de campo. Thelwall e Sud (2015) acharam que as categorias da Scopus têm vantagens de serem transparentes e reprodutíveis. As categorias selecionadas foram: Ciências Agrícolas e Biológicas; Negócios, Gestão e Contabilidade; Ciência da Decisão; Farmacologia, Toxicologia e Farmacêutica; e Ciências Sociais. Os autores consideraram uma limitação o período de 2 meses usados para obter as citações e leitores no Mendeley, devido os dados coletados mais tarde poderem ser maiores nas contagens de leitores e citações, de modo que pequenas diferenças podem ser ignoradas em comparações entre diferentes conjuntos de dados, especialmente para artigos de 2014. Com os dados coletados, obteve-se os resultados que foram divididos em duas partes: contagem de citações e contagem de leitores. Para a contagem de citações, Thelwall e Sud (2015) descobriram que parecem diminuir quase linearmente ao longo do tempo na maioria das categorias, embora tendam a se estabilizar antes em algum momento anterior, devido a

obsolescência, embora os pesquisadores aumentem. Quanto a contagem de leitores, descobriram que são substancialmente diferentes em relação a contagem de citações, ou seja, a contagem de leitores tende a ser relativamente estável em 2004 até 2011 antes de começarem a diminuir mais rapidamente até 2014. Com isso, Thelwall e Sud (2015) confirmam que existe um determinado padrão onde as correlações entre as contagens de citações e leitores para artigos de revistas tendem a aumentar ao longo dos cinco anos e depois se estabilizam. Além disso, a causa das baixas correlações nos primeiros anos sugerem ser um número muito baixo de citações, porque tendem a ter mais leitores do que citadores em poucos anos após a publicação. Neste sentido, Thelwall e Sud (2015) sugerem que a contagem de leitores em Mendeley é um bom substituto para as contagens de citações para as aplicações científicas que envolvem artigos com menos de cinco anos de idade e especialmente quando os artigos estão em seus primeiros anos e em campos com níveis mais altos de uso em Mendeley. No entanto, é alertado que as contagens em Mendeley devem ser tratadas com precaução, devido a possibilidade de manipulação. As diferenças disciplinares no Mendeley também devem ser levadas em consideração devido as diferenças nos números médios de leitores por artigo.

Ao verificar resumidamente os métodos e resultados apresentados nos artigos, percebe-se que o principal estudo é buscar correlações entre quantidade de citações e quantidade de leitores no Mendeley como uma forma alternativa de avaliar citações. Como principal limitação, pode-se considerar as próprias desvantagens da Altmetria, principalmente o fato de que os dados podem ser manipuláveis por meio de softwares de Spam. Portanto, deve-se tomar cuidado antes de generalizar alguma conclusão que envolva o uso das ferramentas de Redes Sociais, incluindo o Mendeley. De qualquer maneira, esses estudos trazem grandes contribuições principalmente para o crescimento metodológico da Altmetria, que ainda precisa de agenda de pesquisa para se consolidar empiricamente.

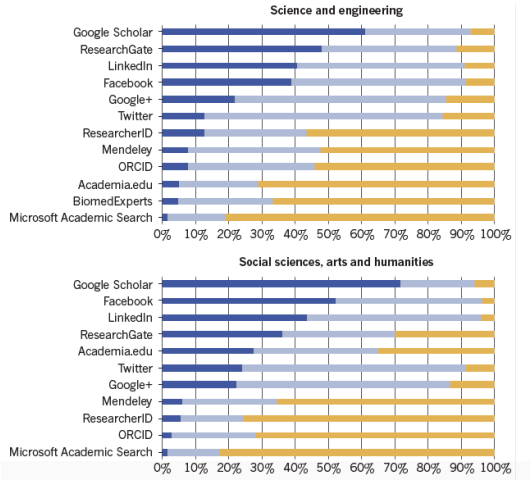
A Altmetria, ao longo dos anos, ganhou bastante trabalhos empíricos como os que foram apresentados aqui. Esses trabalhos estão surgindo, principalmente, devido ao crescimento enorme das Redes Sociais Científicas. Segundo artigo da *Nature*[6] intitulado *Online Colaboration: Scientists and Social Networks* (Colaboração Online: Cientístas e Redes Sociais), as Redes Sociais Científicas cresceram a nível que ninguém esperava e em pouco tempo. A *Nature* conduziu uma investigação para saber por que os estudiosos estão usando redes sociais. Segundo a pesquisa realizada, os motivos se baseiam em possibilitar o

[6] http://www.nature.com/news/online-collaboration-scientists-and-the-social-network-1.15711

compartilhamento de documentos, colaboração online, discutir pesquisas e rastrear visualizações e download dos documentos.

A *Nature* também realizou uma pesquisa para verificar como os cientistas usam as Redes Sociais e outros serviços populares de hospedagem de perfil ou pesquisa e receberam mais de 3500 respostas de 95 países diferentes. Os resultados serão apresentados na Figura 3.

Figura 3 - Resultados que apresentam como os cientistas usam as Redes Sociais.



Fonte: *Nature*. Acesso em: 15/11/2017.

A Figura 3 foi gerada baseada em questionário com 3000 cientistas da área de Ciência e Engenharia e 480 da área de Ciências Sociais, Artes e Humanidades ao redor do mundo. As opções feitas na pesquisa foram: Eu conheço esse site e visito regularmente (marcado de azul escuro), eu conheço esse site, mas não visito regularmente (marcado de azul claro) e eu não conheço esse site (marcado de amarelo). Percebe-se, na Figura 3, que muitos cientistas usam o Google *Scholar* e desconhecem a Rede Social *Microsoft Academic Search*.

A *Nature* também perguntou aos cientistas o que eles mais fazem nestas redes sociais. A atividade mais selecionada em ResearchGate e Academia.edu foi manter um perfil no caso de alguém querer entrar em contato, o que sugere que muitos pesquisadores consideram seus perfis uma maneira de aumentar sua presença profissional online. Depois disso, as opções mais populares envolvem publicar conteúdo relacionado ao trabalho, descobrir pares relacionados, monitorar métricas e encontrar documentos de pesquisa recomendados. Neste sentido, aparentemente as Redes Sociais Científicas são usadas mais para que os cientistas sejam descobertos do que, de fato, uma ferramenta de interação social, como afirma Deni Auclair, analista principal da Outsell, empresa de consultoria de mídia, tecnologia e informação em Burlingame, Califórnia, e mencionada pelo artigo da *Nature*.

Outro ponto interessante a ser colocado e que foi mencionado no artigo da *Nature* é que, após o Mendeley ser comprado pela Elsevier em 2013[7], ganhou um grande potencial, levando em consideração que a plataforma agora se intercepta com outros produtos da Elsevier, como a base de dados Scopus, por exemplo. Com isso, Mendeley pode ter um grande potencial ao longo dos anos como uma das principais plataformas de gerenciador de conteúdo e rede social, por ter ganho uma quantidade grande de documentos.

Na próxima Seção apresentamos a realização de uma revisão sistemática de literatura (RSL) sobre a aplicação do método de reconhecimento de padrões em redes sociais cientificas. A RSL foi baseada nas Redes Sociais Científicas apresentadas no artigo da *Nature*, além de usar termos oriundos da Webometria e Altmetria, com o intuito de verificar o estado da arte das pesquisas que envolvam o uso de Reconhecimento de Padrões em Redes Sociais Científicas Online (este que é um dos pontos principais desta dissertação). A Revisão Sistemática aplicada e as discussões serão apresentadas na Seção a seguir.

2.3 O Reconhecimento de Padrões Aplicado ao Campo Científico: Uma Revisão Sistemática de Literatura

Nesta Seção será apresentado um trabalho que envolve o uso da Revisão Sistemática de Literatura, aplicando-a ao tema de Redes Sociais Científicas e Reconhecimento de Padrões. Antes de apresentar o método da RSL, resultados e discussões, é necessário entender o que é e como funciona a Revisão Sistemática de Literatura.

Segundo Kitchenham (2004), uma Revisão Sistemática de Literatura é "um meio de identificar, avaliar e interpretar toda a pesquisa relevante disponível para uma determinada questão de pesquisa, ou um tópico em uma área, ou algum fenômeno de interesse". Com isso, a revisão sistemática pode ser considerada como um meio de avaliação de todas as pesquisas disponíveis que envolvam uma questão de pesquisa em comum.

De forma a complementar o conceito de revisão sistemática, pode-se citar a definição de Biolchini *et. al.* (2007):

> Uma Revisão Sistemática é desenvolvida, como o termo denota, de maneira formal e sistemática. Isso significa que o processo de condução de pesquisa do tipo sistemático segue uma sequencia muito bem definida e rigorosa de etapas metodológicas, de acordo com um protocolo desenvolvido a priori. Este instrumento é construído em torno de uma questão central, que representa o núcleo da investigação, e que expressa através da utilização de conceitos e termos específicos, que podem ser dirigidos para uma informação relacionada a uma questão de pesquisa específica, pré-definida, focada e estruturada. (BIOLCHINI *et. al.,* 2007, p. 135).

[7] Confirmed: Elsevier has bought Mendeley for $69M - $100M to Expand its Open, Social Education Data Efforts. Disponivel em: https://techcrunch.com/2013/04/08/confirmed-elsevier-has-bought-mendeley-for-69m-100m-to-expand-open-social-education-data-efforts/. Aceso em: 29/01/2018.

Além de Biolchini *et al.* citarem que a Revisão Sistemática se baseia em uma questão central, também afirmam que o método é, como sugere o próprio termo, sistemático, passando por uma rigorosa sequencia de passos metodológicos. Essa sequencia de passos, em termos práticos, vai desde a definição dos termos e *strings* para realização das consultas até a análise qualitativa dos documentos, para que seja mapeado o estado da arte da pesquisa em questão.

Outra definição a ser considerada quando se estuda revisão sistemática está entre os estudos primários e secundários. Kitchenham (2004) afirma que estudos primários são pesquisas individuais que são passíveis de uma futura revisão sistemática, porém, precisam compartilhar da mesma questão central de pesquisa. Já a revisão sistemática é considerada uma forma de pesquisa secundária.

Com base nas definições acima, realizamos uma Revisão Sistemática de Literatura, com o objetivo de identificar o estado da arte do tema de pesquisa proposto nesta Seção. O método consiste em buscar e analisar pesquisas existentes na área de Reconhecimento de Padrões aplicadas a Redes Sociais Científicas Online. Portanto, a RSL foi realizada em 3 etapas sequenciais:

1 – A busca da literatura
2 – A limpeza e tratamento dos dados
3 - A Estratégia de Revisão

Para coletar os documentos, 8 bases de dados foram selecionadas como fontes. As bases utilizadas foram: *Library And Information Science Abstracts* (LISA), *Library, Information Science And Technology Abstracts* (LISTA), *Sociological Abstracts*, *SocINDEX*, IEEE, *Web Of Science*, *Scopus e SAGE Journals Online*. Abaixo segue a justificativa de utilização das bases de dados citadas[8]:

- **LISA**: É uma base de dados de referências e resumos destinada aos profissionais de bibliotecas, ciência da informação e demais especialistas de áreas correlatas. As principais áreas da cobertura incluem: gerenciamento da informação; tecnologia da

[8] Todas as informações das bases de dados foram coletadas no seguinte endereço <https://www.periodicos.capes.gov.br/?option=com_pcollection&mn=70&smn=79> no dia 06/09/2017, incluindo somente o nome de determinada base de dados no campo termo e usando a expressão “Todas as áreas do conhecimento” no campo áreas do conhecimento. O link mencionado acima é imutável mesmo pesquisando várias bases de dados diferentes.

informação; internet; gerenciamento do conhecimento; biblioteconomia; bibliotecas e arquivos; gerenciamento bibliotecário; uso e usuários da biblioteca; recuperação de informações. Possui acesso a mais de 480 títulos de periódicos com disponibilidade de acesso que varia desde 1966 até o presente.

- **LISTA**: A base LISTA é composta de 493 periódicos, sendo que mais de 286 são em texto completo. Muitos desses títulos são revisados por pares. Também oferece acesso a livros, publicações de conferências e relatórios. O acesso aos resumos varia desde 1964 até o presente e o conteúdo de texto completo de 1965 até o presente. As principais áreas de cobertura da base incluem[9]: bibliometria, catalogação, classificação, gerenciamento da informação, biblioteconomia e recuperação da informação.
- ***Sociological Abstracts***: A base de dados oferece referências e resumos especializados em Sociologia e disciplinas afins das ciências sociais, humanas e comportamentais. As principais áreas de cobertura incluem: cultura e estrutura social; família e casamento; história e teoria da sociologia; sociologia organizacional; sociologia política; pobreza e falta de moradia; raça e etnicidade; mudanças sociais e desenvolvimento econômico; controle social; sociologia na saúde e medicina; sociologia da educação; controle social. Permite o acesso a mais de 5.350 títulos de periódicos, além de livros, relatórios, anais de congressos, traduções e documentos de circulação restrita. A disponibilidade de acesso varia desde 1900 até o presente.
- ***SocINDEX***: É uma base de dados que oferece acesso a mais de 4.000 periódicos revisado por pares na área de Ciências Sociais, sendo mais de 809 em texto completo, além de títulos de livros e outros materiais, como recursos educacionais, jornais, panfletos, relatórios e anais de congressos. A disponibilidade de acesso a resumos e referências varia desde 1905 até o presente e a disponibilidade de acesso aos textos completos varia desde 1908 até o presente.
- **IEEE:** A base de dados oferece cerca de 463 periódicos, com disponibilidade de acesso que varia desde 1.872 até o presente; mais de 17.874 conferências com disponibilidade de acesso que varia desde 1951 até o presente e mais de 2.895 normas técnicas com disponibilidade de acesso que varia desde 1949 até o presente. Abrange área de campos elétricos, eletrônicos e computação e áreas afins de ciência e tecnologia.

---

[9] Disponível em: <https://www.ebsco.com/products/research-databases/library-information-science-and-technology-abstracts> . Acesso em: 06/09/2017 (tradução nossa).

- ***Web Of Science*:** A base de dados permite acesso a referências e resumos de todas as áreas do conhecimento. Por meio da *Web Of Science*, estão disponíveis ferramentas para análise de citações, referências e índice h, permitindo análises bibliométricas. Cobre aproximadamente 12.000 periódicos. A assinatura deste conteúdo pelo portal de periódicos da Capes oferece a possibilidade de consulta a 5 coleções: 1) *Science Citation Index Explanded* (SCI-*EXPANDED*) – com disponibilidade de acesso desde 1945 até o presente; 2) *Social Sciences Citation Index* (SSCI) – com disponibilidade de acesso desde 1956 até o presente; 3) *Conference Proceedings Citation Index – Science* (CPCI-S) – com disponibilidade de acesso desde 1991 até o presente; 4) *Arts & Humanities Citation Index* (A&HCI) – com disponibilidade de acesso desde 1975 até o presente; 5) *Conference Proceedings Citation Index - Social Science & Humanities* (CPCI-SSH) - com disponibilidade de acesso desde 1991 até o presente.
- ***Scopus*:** A base *Scopus* indexa títulos acadêmicos revisado por pares, títulos de acesso livre, anais de conferências, publicações comerciais, séries de livros, páginas web de conteúdo científico (reunidos no *Scirus*) e patentes de escritórios. Dispõe de funcionalidades de apoio à análise de resultados (bibliometria) como identificação de autores e filiações, análise de citações, análise de publicações e índice h. Cobre as áreas de Ciências Biológicas, Ciências da Saúde, Ciências Físicas e Ciências Sociais. O período de acesso é desde 1823 até o presente.
- ***SAGE Journals Online*:** A base de dados oferece acesso a 608 títulos de periódicos nas áreas de Ciências Sociais Aplicadas, Ciências Humanas e Medicina. A disponibilidade de acesso ao texto completo varia desde janeiro de 1999 até o presente.

Após a seleção das bases de dados, foi pensada na estratégia de busca. Em geral, ela baseou-se em um conjunto de termos selecionados para dois eixos temáticos: o primeiro "Redes Sociais Científicas" e segundo "Reconhecimento de Padrões". Foi utilizado também o termo *Systematic Review* separadamente, apenas para verificar se existem outras pesquisas sobre o tema baseadas no método de Revisão Sistemática. O Quadro 1 apresenta todos os termos separados por eixos temáticos utilizados na pesquisa.

Quadro 1 - Termos utilizados para busca separados por eixos temáticos.

Redes Sociais Científicas		**Reconhecimento de Padrões**	
Termos	**Strings**	**Termos**	**Strings**
Scientific Social Networks	*Scien* Social Network**	*Machine Learning*	*Machine Learning*
Reference Manager	*Reference Manager*	*Pattern Recognition*	*Pattern Recognition*
Google Scholar	*Google Scholar*	*Artificial Intelligence*	*Artificial Intelligence*
Mendeley	*Mendeley*	*Decision Trees*	*Decision Trees*
ResearcherID	*ResearcherID*	*Neural Networks*	*Neural Network**
Orcid	*Orcid*	*Bayesian Networks*	*Bayesian Network**
academia.edu	*academia.edu*	*K-Nearest Neighbor*	*K-Nearest Neighbor*
Microsoft Academic Search	*Microsoft Academic Search*	*Genetic Algorithms*	*Genetic Algorithms*
ResearchGate	*ResearchGate*	*K-NN*	*K-NN*
Altmetrics	*Altmetric**	*Mining*	*Mining*
Webmetrics	*Webmetric**		
Scientia.net	*Scientia.net*		

Fonte: elaboração própria

Ao observar o Quadro 1, é importante comentar sobre alguns termos utilizados no eixo de "Redes Sociais Científicas". Os termos *Altmetrics* e *Webmetrics* foram utilizados para encontrar trabalhos sobre altmetria ou webmetria e Reconhecimento de Padrões. O termo *Scientia.net*[10] foi incluído devido a testes que foram realizados e foi possível encontrar estudos envolvendo esta Rede Social Científica juntamente com métodos de Reconhecimento de Padrões. Os demais termos usados para compor o eixo de Redes Sociais Científicas foram escolhidos com base nos resultados apresentados na pesquisa realizada pela revista *Nature*[11] que aponta o crescimento do interesse e os hábitos de uso destas Redes Sociais pelos cientistas no mundo.

As consultas nas bases de dados basearam-se em duas combinações de termos com o uso dos eixos apresentados no Quadro 1. A Combinação 1 contem os termos de ambos os eixos, enquanto que na Combinação 2 adicionou-se o termo *Systematic Review* para consulta. O Quadro 2 apresenta o padrão de combinações de termos.

[10] *Scientia.net* é uma Rede Social Científica desenvolvida pela UFPI que se baseia em um ambiente acadêmico para cientistas. Ela permite a interações entre usuários (alunos, professores e pesquisadores) com base em seus interesses comuns. A Rede Social também utiliza ferramentas de aprendizado de máquina para recomendação, fornecendo itens considerados importantes de acordo com o perfil de cada usuário. Os itens podem ser artigos, eventos, etc. Mais detalhes desta Rede Social serão apresentados na discussão da Revisão Sistemática, após o tópico de Resultados.

[11] "*Online Collaboration: Scientists and the social network*". Disponível em: <http://www.nature.com/news/online-collaboration-scientists-and-the-social-network-1.15711>. Acesso em: 12/02/2017

Quadro 2 - Estrutura padrão de combinações usadas nas bases de dados.

Combinações	**-**	**Expressões**
Combinação 1	**AND**	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*"*
		"Machine Learning" OR "Artificial Intelligence" OR "Pattern Recognition" OR "Decision Trees" OR "Neural Network" OR "Bayesian Network*" OR "K-Nearest Neighbor" OR K-NN OR "Genetic Algorithm*" OR Mining*
Combinação 2	**AND**	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*"*
		"Machine Learning" OR "Artificial Intelligence" OR "Pattern Recognition" OR "Decision Trees" OR "Neural Network" OR "Bayesian Network*" OR "K-Nearest Neighbor" OR K-NN OR "Genetic Algorithm*" OR Mining*
		"Systematic Review"

Fonte: elaboração própria

Os padrões apresentados no Quadro 2 foram usados para combinações em todas as bases de dados, porém algumas modificações foram feitas para a IEEE e *Scopus*. Na IEEE existe um limite de termos a serem usados em uma única consulta (15 termos no total) enquanto que na Scopus há um limite de caracteres a serem inseridos em uma busca. Portanto, para que fosse possível a busca nessas bases de dados usando todos os termos listados, foi necessário desmembrar o eixo de Reconhecimento de Padrões em 4 grupos para reduzir o número de caracteres para a *Scopus* e diminuir a quantidade de termos em uma busca para a IEEE. Portanto, o tratamento de busca foi diferenciado para as duas bases de dados. O Quadro 3 apresenta a divisão do eixo de Reconhecimento de Padrões em grupos para a IEEE.

Quadro 3 - Divisão do eixo de Reconhecimento de Padrões em grupos na IEEE.

Grupos	***Strings***
Grupo 1	*"Machine Learning", "Artificial Intelligence", "Pattern Recognition"*
Grupo 2	*"Decision Trees", "Neural Network*", K-NN*
Grupo 3	*"Bayesian Network*", "K-Nearest Neighbor", Mining*
Grupo 4	*"Genetic Algorithm*"*

Fonte: elaboração própria

Para a *Scopus*, o tratamento de divisão de grupos foi diferenciado devido a limitação de caracteres que a base de dados tem. O Quadro 4 apresenta a divisão do eixo de Reconhecimento de Padrões em grupos para a *Scopus*.

Quadro 4 - Divisão do eixo de Reconhecimento de Padrões em grupos na Scopus.

Grupos	***Strings***
Grupo 1	*"Machine Learning"*
Grupo 2	*"Artificial Intelligence"*
Grupo 3	*"Pattern Recognition"*
Grupo 4	*"Decision Trees"*
Grupo 5	*"Neural Network*"*
Grupo 6	*"Bayesian Network*"*
Grupo 7	*"K-Nearest Neighbor"*
Grupo 8	*K-NN*
Grupo 9	*"Genetic Algorithm*"*
Grupo 10	*Mining*

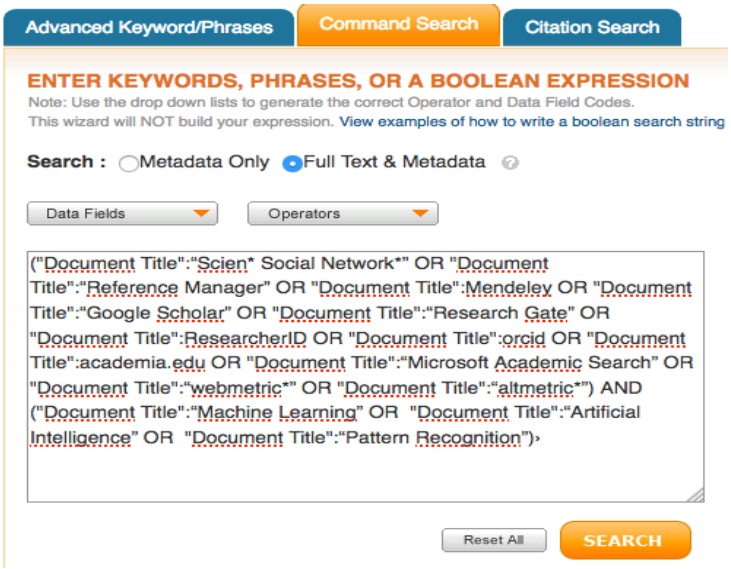
Fonte: elaboração própria

Observando o Quadro 4, percebe-se que a quantidade de grupos na *Scopus* foi maior em comparação com a IEEE justamente pelo limite de caracteres que a base de dados apresenta. Todos os termos no eixo de Redes Sociais Científicas praticamente preenchiam o limite máximo de caracteres na *Scopus*.

Outra diferença que pode ser citada envolvendo a IEEE e a *Scopus* em comparação com as demais bases de dados usadas para coleta dos documentos é em relação ao termo *Scientia.net*. Este termo foi usado separadamente dos demais para o caso da IEEE e *Scopus*, enquanto que, para as demais bases de dados, isso não foi necessário. Neste caso, usou-se apenas o termo *Scientia.net*, seguido pelos termos do eixo de Reconhecimento de Padrões.

Mais uma diferença que pode ser dita está na base de dados IEEE: nela houve a inserção da expressão "*Document Title*" nas *Strings* para indicar que os resultados devem apresentar os termos apenas no título. A Figura 4 apresenta um exemplo de utilização dessa expressão na IEEE. Imagem gerada dia 25/05/2017.

Figura 4 - Exemplo de consulta na IEEE usando *Document Title*.



Fonte: IEEE

Todos os procedimentos citados foram essenciais para a coleta sistemática dos documentos. Após esse processo, foi necessário realizar um tratamento e limpeza dos dados. O primeiro passo foi retirar os documentos repetidos. Após essa etapa, houve a definição de critérios para avaliação da relevância do artigo em relação a questão principal da RSL que se basearam nas seguintes perguntas:

1 – Trata-se de Revisão Sistemática?
2 – O trabalho trata de Redes Sociais Científicas como objeto de pesquisa principal?
3 – O trabalho trata de métricas (altmetria e webometria)?
4 – O trabalho aborda a questão de Reconhecimento de Padrões?

Para que um artigo fosse incluído na seleção da literatura, era necessário que, no mínimo, a resposta fosse "sim" para o conjunto de perguntas 2 e 4 ou 3 e 4. Ou seja, o artigo obrigatoriamente deveria abordar, no mínimo, Redes Sociais Científicas e Reconhecimento de Padrões ou as métricas e Reconhecimento de Padrões. A pergunta 1, referente a Revisão Sistemática, não foi critério de exclusão. Esta pergunta foi usada para identificar e também separar os estudos que representam uma revisão sistemática de literatura sobre o mesmo tema.

Um artigo não é considerado pertinente para a pesquisa quando, no mínimo, a resposta para as perguntas 2, 3 ou 4 for "não", ou seja, se o artigo não abordar Redes Sociais Científicas, métricas (altmetria e webometria) ou Reconhecimento de Padrões, automaticamente será desconsiderado para análise. Os critérios de seleção desses artigos foram baseados na leitura do título e resumo.

Com a limpeza dos dados concluída, a próxima etapa foi verificar os artigos e ver o resultado final retornado na pesquisa de literatura nas bases mencionadas. A próxima Seção consistirá na apresentação dos resultados obtidos desta Revisão Sistemática e do total de documentos encontrados.

2.3.1 Resultados da RSL

Nesta Seção serão apresentados os resultados de todas as buscas que retornaram documentos. Nas bases de dados LISA, LISTA, *SAGE Journals* Online, *SocINDEX*, *Sociological Abstracts* e *Web Of Science*, foram feitas um total de duas consultas para cada uma, conforme o padrão de combinações que foram mencionados na Seção anterior. Para a IEEE, foram realizadas um total de 10 combinações, enquanto que para a *Scopus* foi 22. Neste caso, somando todas as combinações realizadas nas bases de dados, tem-se um total de 44 consultas.

A Tabela 1 apresenta o total de documentos encontrados por pesquisa realizada somente do conjunto de termos que apresentou algum resultado. Todas as buscas foram realizadas no dia 02 de março de 2017.

Tabela 1 - Total de artigos encontrados por pesquisa.

	Termos	**Total de Artigos**
Pesquisa 1 (LISA)	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*" OR Scientia.net AND "Machine Learning" OR "Artificial Intelligence" OR "Pattern Recognition" OR "Decision Trees" OR "Neural Network*" OR "Bayesian Network*" OR "K-Nearest Neighbor" OR K-NN OR "Genetic Algorithm*" OR Mining*	1
Pesquisa 2 (LISTA)	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*" OR Scientia.net AND "Machine Learning" OR "Artificial Intelligence" OR "Pattern Recognition" OR "Decision Trees" OR "Neural Network*" OR "Bayesian Network*" OR "K-Nearest Neighbor" OR K-NN OR "Genetic Algorithm*" OR Mining*	1
Pesquisa 3 (*Web Of Science*)	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*" OR Scientia.net AND "Machine Learning" OR "Artificial Intelligence" OR "Pattern Recognition" OR "Decision Trees" OR "Neural Network*" OR "Bayesian Network*" OR "K-Nearest Neighbor" OR K-NN OR "Genetic Algorithm*" OR Mining*	6
Pesquisa 4 (IEEE)	*("Document Title":"Scien* Social Network*" OR "Document Title":"Reference Manager" OR "Document Title":Mendeley OR "Document Title":"Google Scholar" OR "Document Title":"Research Gate" OR "Document Title":ResearcherID OR "Document Title":orcid OR "Document Title":academia.edu OR "Document Title":"Microsoft Academic Search" OR "Document Title":"webmetric*" OR "Document Title":"altmetric*") AND ("Document Title":"Decision Trees" OR "Document Title":"Neural Network*" OR "Document Title": K-NN)*	2
Pesquisa 5 (*Scopus*)	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*" AND "Decision Trees"*	1
Pesquisa 6 (*Scopus*)	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*" AND "Neural Network*"*	1
Pesquisa 7 (*Scopus*)	*"Scien* Social Network*" OR "Reference Manager" OR Mendeley OR "Google Scholar" OR "Research Gate" OR ResearcherID OR orcid OR academia.edu OR "Microsoft Academic Search" OR "webmetric*" OR "altmetric*" AND Mining*	4
Pesquisa 8 (*Scopus*)	*Scientia.net AND "Machine Learning" OR "Artificial Intelligence" OR "Pattern Recognition" OR "Decision Trees" OR "Neural Network*" OR "Bayesian Network*" OR "K-Nearest Neighbor" OR K-NN OR "Genetic Algorithm*" OR Mining*	1

Fonte: elaboração própria

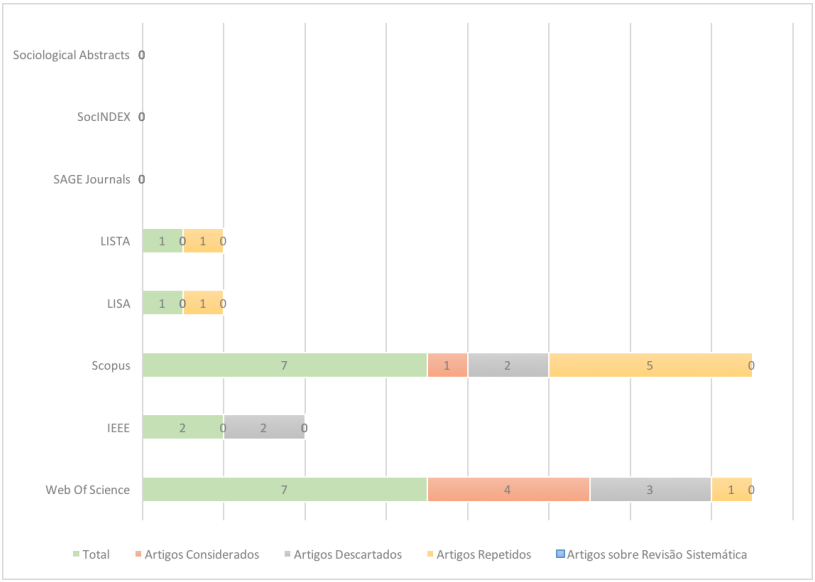
Como é possível observar, das 44 pesquisas realizadas, apenas 8 retornaram resultados. Usando os critérios de inclusão e exclusão mencionados na Seção anterior, na IEEE foram encontrados 2 artigos e ambos foram descartados no processo de tratamento e limpeza dos dados mencionado na Seção anterior, por não tratar de Reconhecimento de Padrões e métricas aplicados a Redes Sociais Científicas. Na *Web Of Science*, foram encontrados 7 artigos, sendo que destes, 4 foram selecionados durante o tratamento e limpeza dos dados, 3 não foram

considerados pertinentes e um deles era repetido (o artigo repetido foi descartado pois a busca em outras bases de dados retornaram o mesmo artigo).

Na *Scopus*, também foram encontrados 7 artigos, dos quais um deles foi considerado pertinente, dois desses artigos foram descartados e 5 artigos foram repetidos na busca em outras bases de dados. Neste caso, para a *Scopus*, alguns artigos selecionados para análise também vieram repetidos de outras bases de dados. As bases de dados LISA e LISTA apresentaram apenas 1 resultado pertinente, que foi repetido em outras bases de dados. As bases *SocIndex*, SAGE *Journals* Online e *Sociological Abstracts* não retornaram resultados pertinentes e nenhuma das bases de dados utilizadas encontraram documentos que apresentem Revisão Sistemática envolvendo Redes Sociais Científicas e métricas com Reconhecimento de Padrões. Além disso, um artigo adicional que foi encontrado em testes usando o termo Redes Sociais foi adicionado para a pesquisa[12].

Resumidamente, apenas 5 documentos foram selecionados como pertinentes para uma posterior análise qualitativa, utilizando os critérios de inclusão e exclusão predefinidos e apresentados acima. A Figura 5 resume os resultados encontrados nas bases de dados após a limpeza e tratamento dos dados.

Figura 5 - Resultado dos dados coletados nas bases de dados.



Fonte: elaboração própria

Com base nessas informações, pode-se considerar que estudos de Reconhecimento de Padrões e Redes Sociais Científicas é um campo de pesquisa bastante novo que apresenta uma

[12] O artigo foi adicionado com base em pesquisas anteriores à qualificação desta dissertação que encontraram resultados envolvendo o uso de uma Rede Social Científica chamada *Scientia.net*. O motivo para a inclusão desse artigo foi justamente devido a aplicações envolvendo Reconhecimento de Padrões na Rede Social Científica mencionada.

agenda de pesquisa urgente para a área de ciência da informação e computação. Ao observar o gráfico acima, percebe-se que, em relação a Reconhecimento de Padrões, há pouquíssimos estudos nesta área sobre Redes Sociais Científicas.

A próxima Seção apresentará a discussão da RSL, mostrando o estado da arte do campo. Essa discussão faz parte da análise qualitativa da Revisão Sistemática de Literatura.

2.3.2 Discussão da RSL

Como visto na Seção anterior onde foram discutidos os resultados da RSL, o campo de estudo de Reconhecimento de Padrões em Redes Sociais Científicas ainda é recente e a escassez de trabalhos publicados indica a necessidade de uma agenda de pesquisa.

As pesquisas realizadas nas bases de dados retornaram ao final 5 artigos que envolvem o uso de Reconhecimento de Padrões em Redes Sociais Científicas. O primeiro artigo a ser citado, encontrado nas bases de dados LISA, LISTA, *Web Of Science* e *Scopus*, publicado na *Revista Española de Documentacíon Científica* (Vol. 38, Nº 4), do autor José Luis Ortega (pesquisasor do *Centro de Ciencias Humanas y Sociales*, Espanha), com o título "*Diferencias y evolución del impacto académico en los perfiles de Google Scholar Citations: Una aplicación de árboles de decisión"* (2015), trabalha com o uso de Árvores de Decisão para analisar a produção e o impacto de mais de 3000 perfis coletados do *Google Scholar Citations* com o objetivo de identificar os segmentos (classificados por gênero, postos acadêmicos e disciplinas) com maior êxito em termos de impacto científico. As Árvores de Decisão também ajudaram a identificar e explorar os atributos que caracterizam o impacto científico por meio da divisão dos perfis durante o processo.

O autor realizou a coleta de dados no *Google Scholar Citations* por meio de um *Script* que rastreou o máximo de perfis possíveis para que pudesse ser possível identificar os dados de cada perfil e seus indicadores bibliométricos. O autor coletou 5 amostras trimestrais que compreendem os períodos de dezembro de 2011 a dezembro de 2012, além de uma amostra anual em dezembro de 2013. Em posse das amostras, Ortega (2015) fez a seleção dos perfis e foram encontrados um total de 12.480 perfis, que estavam presentes nas 3 amostras. Após a seleção, houve um processo de limpeza e normalização, resultando em apenas 3.034 perfis que puderam ser classificados simultaneamente de acordo com as categorias gênero, posto acadêmico e área de investigação.

Em posse da quantidade total de perfis, Ortega (2015) realizou a etapa de classificação nas Árvores de Decisão, que se baseou em dois indicadores para definir a evolução dos pesquisadores no tempo. Esses indicadores foram chamados de cit./art. (quantidade total de

citações recebidas por cada autor dividido pelo número de artigos) e índice/h[13]. Para que os indicadores pudessem ser interpretados pela Árvore de Decisão, foi necessário transformá-los de contínuos a ordinários. Essas variáveis foram, deste modo, ordenadas e agrupadas em quartis. O quartil um correspondeu a 25% dos perfis com pontuações bibliométricas mais altas, enquanto que o quartil quatro agrupou os 25% com piores resultados. Com base nisso, os resultados foram sendo agrupados nos quartis conforme a porcentagem resultante após classificação nas Árvores de Decisão.

Os resultados obtidos por Ortega (2015) neste experimento foram divididos em duas partes: atividade atual (avaliar como estão as atividades dos pesquisadores atualmente) e evolução das atividades (o desenvolvimento das atividades no tempo), ambas usando as três categorias citadas: gênero, posto acadêmico e área de investigação. Para a atividade atual, na categoria posto acadêmico, 35% dos professores titulares estão dispostos no Q1, 54% dos bolsistas de pós-doutorado estão no Q4, mostrando resultado acadêmico mais baixo. Na área de investigação, Professores Titulares de Ciências da vida contem 47.7% das investigações em Q1, enquanto que Ciências Físicas - Artes e Humanidades - Ciências Sociais[14] detêm 44.3% em Q4. Isso significa que professores titulares de Ciências da Vida têm mais impacto que professores titulares em Ciências Físicas – Artes e Humanidades – Ciências Sociais na categoria área de investigação. Segundo a pesquisa, o gênero é pouco relevante apresentando diferenças apenas no caso de professores titulares de ciências físicas onde 31.5% dos homens estão em Q1 e 13.4% em Q4, enquanto que 26% das mulheres estão tanto em Q1 quanto em Q4. A Tabela 2 apresenta a distribuição Cit. Art e índice/h por quartis da categoria "atividade atual" dos pesquisadores.

Tabela 2 - Distribuição de Cit./Art. e índice h por quartis na atividade atual.

Cuartil	*Cit./Art.*	*N*	*Índice h*	*N*
Q1	30.3 - 891.18	758	28 - 234	737
Q2	14.24 - 30.2	759	15 - 27	772
Q3	6.78 - 14.22	758	8 – 14	712
Q4	0 - 6.77	759	0 – 7	813
Total		3034		3034

Fonte: ORTEGA 2015, p. 5.

Com relação à evolução das atividades dos perfis estudados percebe-se que, na categoria posto acadêmico, bolsistas de pós-doutorado (59.7%) e assistentes de investigação

[13] Segundo Correia e Mesquita (2014), o índice/h é definido pelo seguinte exemplo: "Um cientista tem índice/h se h dos seus artigos científicos (Np) tiver pelo menos h citações e os outros artigos (Np – h) tiverem ≤ h citações cada". Neste caso, segundo as autoras, somente os artigos com maior número de citações contribuem para o índice/h de determinado autor, de forma a simplificar todo o processo de avaliação bibliométrica.

[14] Todas as áreas de investigação que estiverem separadas por travessão (-) foram classificadas juntas.

(49.5%) são os que mais aumentam sua pontuação de citações por artigo publicado, enquanto que professores titulares (30.8%) é o grupo que tem mais proporção de valores no Q4, ou seja, os professores titulares têm menor pontuação de citações por artigo do que bolsistas de pós-doutorado e assistentes de investigação. Na área de investigação, os perfis com as piores pontuações são professores titulares de ciências da vida - Artes e humanidades (38.3%) e Ciências Sociais - Ciências da Saúde - Multidisciplinar (34.6%). Professores adjuntos de ciências sociais - Artes e Humanidades (44.7%) são os que mais incrementam seu impacto. A Tabela 3 apresenta a distribuição Cit./Art. e índice/h por quartis da categoria "evolução das atividades" dos pesquisadores.

Tabela 3 - Distribuição da porcentagem de crescimento de cit./Art. e índice/h agrupados por quartis.

Cuartiles	*Cit./Art.*	*N*	*Índice h*	*N*
Q1	5.87 - 133.99	759	4.8 - 51.67	760
Q2	2.99 - 5.86	759	2.83 - 4.75	775
Q3	1.03 - 2.98	758	1.61 - 2.78	741
Q4	0 - 1.02	758	0 - 1.6	758
Total		3034		3034

Fonte: ORTEGA 2015, p. 8.

Com base nas Árvores de Decisão, o autor chegou a conclusão de que o primeiro aspecto qualitativo que diferencia a atividade científica em termos de impacto é a produção acadêmica de um autor. Pesquisadores com uma carreira estabelecida têm melhor impacto científico do que os pesquisadores iniciantes. Isso também é observado em termos de disciplinas, detectando que pesquisadores de Ciências da Vida têm mais impacto do que pesquisadores em Artes e Humanidades. Do ponto de vista evolutivo, as Árvores de Decisão mostram que os jovens pesquisadores, principalmente das Ciências Humanas e Sociais, aumentam seus currículos mais rapidamente do que os professores *seniors* que apresentaram pequenos aumentos em todas as áreas estudadas. Isso pode ser interpretado como um fenômeno de crescimento em que os pesquisadores novatos tendem a aumentar seus currículos nos estágios iniciais de suas carreiras e, em seguida, permanecem estáveis em sua maturidade. O autor também considerou que a Árvore de Decisão é uma ferramenta de mineração de dados aconselhável para estudar a influência de vários aspectos qualitativos envolvidos na atividade científica em relação ao impacto e produção, já que a ferramenta possibilitou agrupar e categorizar quais os tipos de autores que descrevem um impacto científico maior, considerando o gênero, posição acadêmica e disciplina. Por fim, o autor também considera o *Google Scholar* como uma ferramenta bibliográfica apropriada por facilitar a construção de artigos por autores com indicadores bibliométricos comparáveis.

O segundo artigo pertinente encontrado na RSL, publicado em *The International Arab Journal of Information Technology* (Vol. 13, Nº 2), intitulado "*Investigation and Analysis of Research Gate User's Activities using Neural Networks*" (2016) do autor Omar Alheyasat (*Al-Balq'a Applied University*, Jordânia) e encontrado nas bases de dados *Web Of Science* e *Scopus*, trata de um estudo que envolve uma investigação das atividades de uma amostra composta por um milhão de usuários do *Research Gate* (RG) usando o método de Redes Neurais. O motivo de uso do *Research Gate*, segundo o autor, são vários: primeiro, por ser privado e construído apenas para cientistas e pesquisadores. Segundo, tem um fórum que permite aos pesquisadores trocar perguntas e respostas. Em terceiro lugar, permite que desenvolvedores e pesquisadores busquem oportunidades. Em quarto lugar, maximiza a colaboração entre pesquisadores compartilhando publicações, opiniões e conhecimentos especializados. Finalmente, *posts* em páginas de usuário foram substituídos pelo fórum de pergunta/resposta. O objetivo do trabalho é identificar, com a amostra coletada no *Research Gate* e usando as Redes Neurais, alguma correlação entre os dados dos perfis dos usuários e o número de seus seguidores. Espera-se que essa correlação seja através da publicação de trabalhos acadêmicos, além de número de citações e visualizações de determinado perfil.

Para que fosse possível a coleta de dados sobre um milhão de usuários, dois *web-crowlers*[15] foram desenvolvidos. O primeiro gera uma lista de *links* e o segundo utiliza a lista gerada para coletar os dados na rede. Os *crowlers* geram como saída arquivos que consistem em tuplas[16] e cada uma delas representa um atributo de um usuário no *Research Gate*. Estas tuplas consistem de 10 valores separados. Os valores de informação na tupla representam: 1) O número de citações de um autor (*cit.*); 2) O número de usuários do RG que visualizaram publicações de determinado usuário (*view.*); 3) o número de publicações dos usuários do RG (*pub.*); 4) O número de usuários do RG que visualizaram determinado perfil (*pro.*); 5) O fator de impacto cumulativo para todas as publicações do usuário (*im.*); 6) O campo de especialidade do usuário no RG (*maj.*); 7) O número de usuários do RG que estão seguindo determinado usuário (*fing.*); 8) O número de seguidores para determinado usuário no RG (*fer.*); 9) O número de perguntas e respostas que determinado usuário contribuiu (*q.*); 10) O número de perguntas e respostas que o usuário viu (*qisview.*). Todos esses 10 atributos preencheram o conjunto de dados que em seguida foi classificado pela Rede Neural.

---

[15] *Web-Crowler* "é um tipo de robô usado por buscadores para encontrar e indexar páginas em um site. Ele captura informações de páginas e cadastra os links encontrados, possibilitando encontrar outras páginas e mantendo sua base atualizada". Disponível em: <http://www.globalad.com.br/blog/o-que-e-crawler/>. Acesso em: 10/09/2017.

[16] Sequência de valores separados por vírgulas.

Ao alimentar o conjunto de dados na Rede Neural, obteve-se os resultados, que neste experimento foram divididos basicamente em duas partes: estatísticas de perfil e atividades dos usuários. Para as estatísticas de perfil, observou-se que o número médio de seguidores (*number of followers*) e o número médio de *links* que um usuário gera (*number of following*) são aproximadamente os mesmos. Isso significa que um usuário segue a maioria de seus seguidores. A variância entre o valor máximo e o valor médio do número total de perfis mostra que alguns usuários são mais populares do que outros no RG. A Tabela 4 apresenta a estatísticas de perfil geral dos 10 valores citados anteriormente.

Tabela 4 - Estatísticas de perfil.

Data	MAX	Avg	STD
Number of Publications	3127	12.262	36.365
Number of Profile Views	2124314	212.3	3625.4
Number of Publication Views	1000000	272	1945.13
Cumulative Impact Factor of Researchers	11069	27.58	111.72
Number of Following	1148	7.5	15.7999
Number of Followers	808	7.57	15.3483
Total Number of Citation	39601	44.16	283.36
Number of Question/Answer	744	0.282	4.1634
# of Question Views	74958	9.28	216.82

Fonte: ALHEYASAT 2016, p. 323.

Finalmente, seguindo os dados da tabela acima, a contribuição no ambiente pergunta/resposta é limitada (*number of question/answer*). O valor médio é menor do que 1 e o número máximo é 744, considerado pequeno em comparação com o número de *posts* que outros usuários em outras Redes Sociais Online escrevem. Para as atividades dos usuários, nota-se que cerca de 60% dos usuários têm uma pontuação RG 0, ou seja, que a maioria dos usuários não estão ativos nem para completar seus perfis. Mais de 95% dos usuários não têm contribuição no ambiente de pergunta/resposta. A maior contribuição de qualquer usuário neste campo nem sequer superou 900 *posts* ou comentários. Além disso, 80% dos usuários têm menos de 10 seguidores em suas redes, 75% dos usuários estão seguindo menos de 10 usuários e 98% de todos os usuários estão seguindo menos de 100 usuários.

A pesquisa conduzida por Alheyasat (2016) demonstra que existe uma alta correlação entre os usuários e seus seguidores por meio de publicação de atividades de pesquisa. Isso significa que a maioria dos usuários estão interessados em usuários com maiores habilidades de pesquisa no *Research Gate*. Porém, esta correlação é baixa quando se trata do campo de

pergunta/resposta, o que indica que este recurso no *Research Gate* é pouco usado pelos seus usuários.

O terceiro artigo, publicado na *International Conference on Inteliggent Computing* (ICIC), intitulado "*Mining Google Scholar Citations: An Exploratory Study*" (2012), dos autores Ze Huang e Bo Yuan (*Tsinghua University*, China) e encontrado na base de dados *Scopus*, trabalha com classificação de citação de várias disciplinas relacionadas a área de computação no *Google Scholar Citations* usando o algoritmo *K-Means*. As disciplinas analisadas foram: Mineração de Dados, Inteligência Artificial, Bioinformática, Recuperação da Informação, Aprendizado de Máquina e Reconhecimento de Padrões. Ao analisar essas disciplinas, o objetivo dos autores era verificar os padrões gerais de citações, a correlação entre os índices métricos[17]. os padrões de citação pessoal dos pesquisadores e a transformação dos temas de pesquisa ao longo do tempo.

Para que fosse possível analisar as disciplinas mencionadas, foi necessário realizar um agrupamento no qual foram selecionados apenas pesquisadores com mais de 200 citações em pelo menos 10 trabalhos (o que totalizou 3.539 pesquisadores). Em seguida, os documentos de cada pesquisador foram classificados com base nos números de citação em ordem decrescente. Depois, foi criado um vetor "10D" com o primeiro elemento correspondente as citações dos primeiros 10% de *papers* mais citados entre todas as publicações de determinado pesquisador. O segundo elemento correspondeu a proporção de citação dos segundos 10% *papers* e assim por diante, até completar o vetor "10D". Neste caso, os autores foram agrupando, em cada posição do vetor, a proporção de citações dos *papers* mais citados. A primeira posição do vetor agrupa os 10% superiores enquanto a última posição tem os 10% inferiores. Os autores do artigo desenvolveram este método partindo da hipótese de que alguns pesquisadores podem ter um pequeno número de documentos altamente citados, enquanto outros autores podem ter citações distribuídas de forma relativamente uniforme.

Com relação aos índices métricos mencionados, os autores afirmam que várias métricas foram projetadas e aprimoradas, como o *h-index* e o *g-index*. Para o caso do *Google Scholar*, é utilizado apenas 3 métricas, que são: o número total de citações (TC), o índice/h e o índice i10. O índice/h, que já foi mencionado no primeiro artigo discutido nesta RSL e agora mencionado no artigo de Huang e Yuan (2012), tenta abordar a produtividade e os fatores de impacto e é definido como o número máximo h, de modo que existem h *papers*

[17] É utilizada para quantificar o impacto do resultado de pesquisa de um indivíduo, apesar de estar longe de ser suficiente para comparar e avaliar o trabalho de pesquisa de forma abrangente. HUANG; YUAN (2012), p. 185 "tradução nossa".

com número de citações ≥ h. O índice i10, segundo os autores, foi introduzido em julho de 2011 e indica a quantidade de documentos acadêmicos de um autor que recebeu pelo menos 10 citações.

Os autores também calcularam o coeficiente de correlação (CC) entre os índices métricos em cada disciplina. A Tabela 5 apresenta estas correlações (com o número de autores mostrado entre parêntesis). Em todas as disciplinas, os valores de CC foram superiores a 0,7 e algumas disciplinas apresentaram valores de CC mais fortes do que outros.

Tabela 5 - Coeficiente dos índices métricos em diferentes disciplinas.

	All Publications			*Recent Publications (Since 2007)*		
Discipline	TC vs. h	TC vs. i10	h vs. i10	TC vs. h	TC vs. i10	h vs. i10
IR (511)	0.8495	0.9004	0.9389	0.8583	0.9185	0.9289
AI (1000)	0.7829	0.7738	0.9710	0.8276	0.8366	0.9636
Bio (999)	0.8019	0.7282	0.9374	0.7752	0.7205	0.9180
DM (879)	0.7862	0.7669	0.9327	0.8052	0.8225	0.9306
ML (1000)	0.7690	0.7240	0.9512	0.7604	0.7498	0.9511
PR (539)	0.8434	0.8583	0.9306	0.8404	0.8613	0.9334

Fonte: HUANG; YUAN 2012, p. 186

Como é possível observar na Tabela 5, algumas disciplinas (por exemplo, IR[18]) apresentaram valores de CC mais fortes entre o TC, índice/h e índice/i10 do que outros (a exemplo de AI[19] e ML[20]).

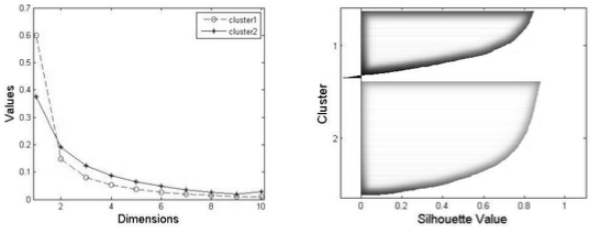
A partir da analise do *Google Scholar,* Huang e Yuan (2012) verificaram que os pesquisadores, especialmente os mais eminentes em AI, tendem a ter um número total de citações muito maior em comparação às disciplinas como Recuperação da Informação e Reconhecimento de Padrões. Entretanto, é possível que a análise do número total de citações não seja uma métrica confiável para avaliar e comparar o impacto de pesquisadores em diferentes disciplinas.

Os autores também observaram os dois centroides[21] gerados pelo *cluster k-means* após agrupamento dos dados. Os pesquisadores do *cluster* 1 tenderam a ter a maioria das citações concentradas nos poucos melhores artigos. Os 10% mais importantes contribuíram com cerca de 60% do total de citações. Os artigos dos autores do *cluster* 2 receberam citações de forma mais uniforme, tanto que é possível observar que há mais autores no *cluster* 2 (2225 ou

[18] *Information Retrieval*, sigla em inglês para Recuperação da Informação.
[19] *Artifical Inteligence*, sigla em inglês para Inteligência Artificial.
[20] *Machine Learning*, sigla em inglês para Aprendizado de Máquina.
[21] Está relacionado com o k de *k-means*, que é a quantidade de pontos centrais dos grupos. No caso, cada *cluster* tem seus pontos centrais (centroides), onde se concentram os dados. NOGARE, 2015.

62,87%) do que no *cluster* 1 (1314 ou 37.13%). A Figura 6 apresenta os centroides de ambos os *clusters* (a esquerda) e a avaliação de autores dos *clusters* (a direita).

Figura 6 - Padrão de citação nos centroides (esquerda); avaliação dos autores nos clusters (direita).



Fonte: HUANG; YUAN 2012, p. 187.

Huang e Yuan (2012) também desenvolveram uma nuvem de *tags* para tentar identificar os tópicos ou palavras-chave que evoluíram em uma disciplina. Os autores dividiram os *papers* em dois subconjuntos: antes de 2007 e desde 2007 para observar a mudança ao longo do tempo. A Figura 7 apresenta um exemplo de uso das palavras no campo de IR. Algumas palavras comuns no título como *based, approach, systems, analysis e using* foram ignoradas, já que elas aparecem frequentemente em todas as disciplinas.

Figura 7 - Nuvem de tags de títulos em IR: Acima (antes de 2007); abaixo (desde 2007).

Fonte: HUAN; YUANG 2012, p. 188

Ao comparar as duas nuvens correspondentes aos dois períodos de tempo, os autores encontraram algumas pistas interessantes. Por exemplo, o tamanho da fonte da palavra *semantic* aumentou, o que pode sugerir que a recuperação semântica tem crescido rapidamente como tópico de pesquisa principal em IR. Enquanto isso, a palavra *social* apareceu apenas na nuvem inferior, indicando que uma nova direção de pesquisa relacionada às redes sociais surgiu nos últimos anos.

Os autores apresentam, como conclusão deste artigo, que o estudo foi um dos primeiros sobre citações do *Google Scholar*. Huang e Yuan (2012) descobriram, com base no estudo, que diferentes disciplinas tinham números diferentes de citações e haviam fortes correlações entre o índice/h, índice i10 e o número total de citações. Também foi identificado dois grupos distintos de autores em termos de distribuição de citações. Eles também demonstraram a eficácia da nuvem de *tags* na descoberta de tendência de tópicos em determinado campo de pesquisa. Para um trabalho futuro, os autores pretendem coletar conjuntos de dados mais abrangentes para realizar análises mais completas. Os autores também acreditam ser interessante combinar os domínios de conhecimento com os resultados da análise de dados para fornecer mais informações sobre cada disciplina.

O quarto artigo encontrado durante a RSL nas bases IEEE e *Web Of Science*, apresentado na *Fourth International Conference on Computational Aspects of Social Networks* (2012), intitulado "*Machine Learning Algorithms Applied in Automatic Classification of Social Network Users"* (2012), dos autores Bruno Lima e Vinicius Machado (Departamento de Informática e Estatística, Universidade Federal do Piaui – UFPI), utiliza algoritmos de aprendizado de máquina na rede social *Scientia.net* para verificar qual dos algoritmos (em termos de paradigma e individual)[22] tem melhor desempenho na classificação dos usuários. Segundo Lima e Machado (2012), o *Scientia.net*[23] é uma rede social voltada para o ambiente acadêmico com conteúdo específico para cientistas que querem compartilhar suas pesquisas ou avançar seu trabalho através da interação com outros pesquisadores. Foi criada através de implementação de ferramentas que permitem a interação de seus usuários (alunos, professores e pesquisadores) com base em seus interesses comuns, e funciona através de um algoritmo de aprendizado de máquina. Além disso, o *Scientia.net* faz recomendações , fornecendo automaticamente aos seus utilizadores os itens relevantes relacionados ao seu perfil. Ou seja, é indicado para encontrar artigos, eventos e contatos de outros pesquisadores.

Como foi dito anteriormente, Lima e Machado (2012) utilizam algoritmos de aprendizado de máquina para classificação dos usuários no *Scientia.net*. Os algoritmos selecionados foram as Redes Neurais *Multilayer Perceptron, Support Vector Machine, K-Means* e *Konohen Network*. O primeiro e segundo algoritmos citados possuem paradigma de

[22] Os autores usaram 4 algoritmos de inteligência artificial. Dois deles têm o paradigma supervisionado (Redes Neurais e *Support Vector Machine*) e os outros dois têm o paradigma não supervisionado (*K-Means* e *Konohen Networks*). Neste caso, o trabalho faz uma verificação em termos de paradigma (analisando os algoritmos separados em paradigma) e individual (verificando a execução de cada um individualmente).

[23] Desenvolvida na Universidade Federal do Piauí (UFPI, Piauí, Brasil).

aprendizagem supervisionada, enquanto que o terceiro e quarto algoritmos possuem paradigma não supervisionado.

Os autores implementaram os algoritmos baseados no próprio algoritmo do *Scientia.net*, que usa a biblioteca WEKA[24]. A única exceção foi o *Konohen Network*, pois não existe implementações desse método em bibliotecas WEKA. Lima e Machado (2012) também criaram um banco de dados com base na estrutura de banco de dados do *Scientia.net*. A base de dados é composta pelo perfil acadêmico de 2000 usuários, oriundos de 20 áreas do conhecimento. O perfil acadêmico dos usuários na base é constituído por oito atributos, que são:

1. Graduação (*Undergraduate*)
2. Pós-Graduação:
    a. Mestrado (*Master*)
    b. Subárea do Mestrado (*Master Sub Area*)
    c. Doutorado (*Doctoral*)
    d. Subárea do Doutorado (*Doctoral Sub Area*)
    e. Pós-Doutorado (*Postoctoral*)
    f. Subárea do pós-doutorado (*Posdoctoral Sub Area*)
3. Área de Interesse (*Area Of Interest*)

Esta base de dados coletada do *Scientia.net* foi replicada para 9 tabelas com os mesmos registros, mudando apenas a ordem de inserção dos registros para executar os algoritmos, a fim de garantir a legitimidade dos resultados uma vez que os algoritmos são sensíveis a ordem. Os resultados de cada execução foram reunidos para atingir uma média dos resultados.

Para classificar utilizando algoritmos supervisionados, foi necessário definir as classes as quais os dados serão classificados. Para o *Multilayer Perceptron* e o *Support Vector Machine*, foram definidas 20 classes de acordo com a área de interesse dos usuários, baseadas nas áreas de conhecimento especificadas pela Capes. Para algoritmos não supervisionados, os grupos gerados após a execução foram analisados manualmente e determinou a

[24] Weka é uma coleção de algoritmos de aprendizado de máquina para tarefas de mineração de dados. Os algoritmos podem ser aplicados diretamente a um conjunto de dados ou chamados pelo seu próprio código (que deve ser desenvolvido em Java). Weka contém ferramentas para pré-processamento de dados, classificação, regressão, agrupamento, regras de associação e visualização. Também é adequado para o desenvolvimento de novos esquemas de aprendizado de máquina. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 12/09/2017 “tradução nossa”.

homogeneidade desses grupos, levando em consideração a similaridade de todos os atributos de perfis incluídos em um grupo. Assim, foi possível determinar se um usuário foi inserido por engano em um grupo.

Após determinar as classes para os algoritmos supervisionados e os grupos para algoritmos não supervisionados, cada classe e cada grupo foram analisados separadamente para determinar a porcentagem da taxa de erro. A taxa de erro média do algoritmo foi de 10 execuções. Foram também analisados os tempos médios de execução dos algoritmos, sendo o tempo médio a média dos tempos obtidos em 10 execuções de cada algoritmo. Neste caso, este é o tempo de treinamento dos algoritmos.

Todo esse processo foi necessário para que os autores encontrassem os resultados do experimento. Para testar as Redes *Multilayer Perceptron* (MLP) e *Support Vector Machine* (SVM), foi necessário dividir o banco de dados em duas partes: a base de treinamento, com 1800 usuários e a base de testes com 200 usuários. O MLP utilizado tinha 10 camadas ocultas, com uma taxa de aprendizagem de 0,1 e foi realizado com 5000 iterações, tendo um tempo de 3,58 minutos para treinar e uma taxa de erro de 1,5% com uma média de três usuários classificados incorretamente, dois da classe Odontologia e um da classe Inteligência Artificial. A Tabela 6 apresenta os resultados da Rede *Multilayer Perceptron*.

Tabela 6 - Resultados das redes Multilayer Perceptron.

Amount of Classified User by Class			
Class	**Correct**	**Incorrect**	**Error (%)**
Medicine	10	0	0
Odontology	8	2	20
Biology	10	0	0
Chemical	10	0	0
Physics	10	0	0
Mechanical Engineering	10	0	0
Math	10	0	0
Artificial Intelligence	9	1	10
Operation Research	10	0	0
Computer Architecture	10	0	0
Network	10	0	0
Database	10	0	0
Software Engineering	10	0	0
Electrical Engineering	10	0	0
Civil Engineering	10	0	0
Geography	10	0	0
History	10	0	0
Law	10	0	0
Economia	10	0	0
Literature	10	0	0
		Mean Error Rate	1,5

Fonte: LIMA; MACHADO 2012, p. 61.

Para o SVM utilizou-se uma função de núcleo de base radial e foi treinada com 1800 perfis de usuários e testada com 200 perfis, tendo um tempo de treinamento de 0,0096 minutos e uma taxa de erro de 1,0%. A Tabela 7 apresenta os resultados obtidos do SVM.

Tabela 7 - Resultados do SVM.

Amount of Classified User by Class			
Class	**Correct**	**Incorrect**	**Error (%)**
Medicine	10	0	0
Odontology	8	2	20
Biology	10	0	0
Chemical	10	0	0
Physics	10	0	0
Mechanical Engineering	10	0	0
Math	10	0	0
Artificial Intelligence	10	0	10
Operation Research	10	0	0
Computer Architecture	10	0	0
Network	10	0	0
Database	10	0	0
Software Engineering	10	0	0
Electrical Engineering	10	0	0
Civil Engineering	10	0	0
Geography	10	0	0
History	10	0	0
Law	10	0	0
Economy	10	0	0
Literature	10	0	0
		Mean Error Rate	1

Fonte: LIMA; MACHADO 2012, p. 61.

Observando a tabela, percebe-se que houve uma média de dois usuários incorretamente classificados na classe Odontologia. Os autores observaram esses usuários incorretamente classificados e descobriram que, na verdade, a classe deles deveria ser Arquitetura de Computador. A justificativa dos autores a esse fenômeno surgiu por que estes usuários incorretamente classificados têm perfis com atributos que se parecem com a maioria dos usuários classificados em Odontologia.

Para os algoritmos não supervisionados, o primeiro método testado foi a *Konohen Networks*. Para os testes, foi estabelecida uma rede de 20 neurônios, pois havia um total de 20 áreas diferentes de conhecimento, que é aproximadamente a quantidade de grupos que se desejou obter, formando uma grade com dimensões 4 x 5. O algoritmo foi executado usando 5000 iterações, com uma taxa de aprendizado inicial de 0,5 e uma largura inicial de 0,8. *Kohonen Networks* gerou um total de 18 grupos de usuários, e cada grupo formou uma área diferente do conhecimento. A exceção foram dois grupos onde a rede gerou um grupo com usuários de Direito e Economia e outro grupo de Medicina e Odontologia, devido às sub-áreas comuns desses usuários. A *Kohonen Networks* gerou grupos em um período de 5,77 minutos e uma taxa de erro média de 3,33%, pois alguns grupos apresentaram perfis totalmente diferentes aos demais participantes do grupo. A Tabela 8 apresenta os resultados obtidos em *Konohen Networks*.

Tabela 8 - Resultados do *Konohen Networks*.

Amount of Classified User by Class			
Class	**Correct**	**Incorrect**	**Error (%)**
Literature	100	0	0
Law/Economy	200	2	0,99
Geography	100	0	0
Civil Engineering	100	0	0
History	100	0	0
Software Engineering	100	7,3	6,8
Electrical Engineering	100	0	0
Network	79	15	15,96
Computer Architecture	100	24,7	19,81
Math	100	0	0
Database	69	2	2,82
Artificial Intelligence	100	1	0,99
Mechanical Engineering	99,8	0	0
Chemical	100	9,6	8,76
Operation Research	99	2,8	2,75
Physics	99,8	0,2	0,2
Biology	89	0	0
Medicine/Odontology	200	1,8	0,89
		Mean Error Rate	3,33

Fonte: LIMA; MACHADO 2012, p. 62.

Para o algoritmo *K-Means*, utilizou um total de 20 centróides, pois, assim como na *Kohonen Network*, estes são aproximadamente a quantidade de grupos que se desejou obter no final da execução. Tanto a *Kohonen Network* quanto *K-means* criaram 18 grupos com usuários de duas áreas de conhecimento. Tais grupos, um com usuários de História e Direito e o outro com usuários de Medicina e Odontologia, são semelhantes à de *Kohonen Networks*. Este algoritmo apresentou uma taxa de erro de 9,84% devido à quantidade de usuários classificados incorretamente. O tempo dessa execução foi de 0,012 minutos. A Tabela 9 apresenta os resultados obtidos em *K-means*.

Tabela 9 - Resultados do algoritmo K-Means.

Amount of Classified User by Class			
Class	**Correct**	**Incorrect**	**Error (%)**
Medicine/Odontology	200	15	5,98
Biology	73	0	0
Chemical	100	42	29,58
Physics	60	0	0
Mechanical Engineering	100	40	28,57
Math	60	0	0
Artificial Intelligence	100	10	9,09
Operation Research	100	10,1	9,17
Computer Architecture	90	20,8	18,77
Network	80	0	0
Database	41,9	0,3	0,71
Software Engineering	100	30,4	23,31
Electrical Engineering	90	36,5	28,85
Civil Engineering	100	0	0
Geography	100	0	0
History/Law	200	60	23,08
Economy	50	0	0
Literature	90	0	0
		Mean Error Rate	9,84

Fonte: LIMA; MACHADO 2012, p. 62

Com base nos dados obtidos, os autores concluíram que, pelo paradigma de aprendizagem, os algoritmos não supervisionados são mais adequados para esse tipo de problema, pois não é necessário obter uma prévia de classes como nos algoritmos supervisionados. Com isso, *Konohen Network* seria o algoritmo mais indicado para este tipo de problema, por ter o paradigma não supervisionado e ter uma taxa de erro médio menor que o algoritmo *K-Means*.

O quinto artigo, publicado na revista *Social Network Analysis Mining*[25], intitulado "*Automatic Labeling of Social Network Users Scientia.net Through The Machine Learning Supervised Application"* (2015), dos autores Bruno Lima, Vinicius Machado e Lucas Lopes (Departamento de Computação, Universidade Federal do Piauí - UFPI) e encontrado na base *Web Of Science*, é assinado pelos mesmos autores do trabalho anterior, com a colaboração de Lucas Lopes e utiliza praticamente o mesmo procedimento metodológico do trabalho mencionado anteriormente. Sua diferença está no processo de *Labeling* (rotulagem) aplicado. Este processo baseia-se no uso de um algoritmo não supervisionado para ser usado na definição de *clusters* para em seguida aplicar um algoritmo supervisionado a cada atributo de cada *cluster*. A avaliação dos algoritmos não supervisionados pode identificar quais atributos são relevantes para o problema. Os autores pretendem fazer classificação de usuários no *Scientia.net* usando o *K-Means* (algoritmo não supervisionado para a geração dos *clusters*) e as Redes Neurais *Multilayer Perceptron* (algoritmo supervisionado para o treinamento dos dados). O banco de dados utilizado também era composto por 2000 usuários de 20 áreas diferentes do conhecimento. A tabela de banco de dados também foi replicada em 9 tabelas com os mesmos registros, alterando somente a ordem de inserção.

Para gerar os resultados, foi necessário usar o algoritmo *k-means* com o comando (x, k), onde x é uma matriz que contem todos os elementos (banco de dados) e o k é o número de *clusters* a serem gerados. Para usar as Redes Neurais *Multilayer Perceptron*, foi feito um treinamento no qual 60% dos dados foram utilizados para este fim, e os demais 40% foram usados para testes.

Os parâmetros e topologia dos dois algoritmos utilizados foram adicionados aos parâmetros do modelo de discretização, onde I representou a quantidade de Redes Neurais por atributo e V é a variação relativa da maior taxa de acertos por grupo. Isso resultou em um grande número de combinações possíveis de modo que os resultados representaram uma pequena parte das combinações. A marcação é feita em função dos *clusters* gerados pelo *k-means*. Ainda assim, os marcadores são específicos e podem ser diferentes para cada execução, dependendo do agrupamento.

O segundo *cluster* gerado pelo algoritmo *k-means* foi o único que apresentou um dos atributos de uma subárea. Pelo fato desses atributos terem informações específicas, eles não tendem a ter alta relevância. No entanto, foi sugerido no grupo 2 uma classificação correta, uma vez que o grupo continha apenas 12 elementos, sendo assim um grupo bem específico.

[25] Publicado online dia 31 de julho de 2015.

Além disso, ao observar os outros grupos gerados, a maioria (55%) dos *clusters* teve uma taxa de acerto de 100%. Ao calcular a média das execuções das taxas de acerto, obtêm uma média de 93.357%, o que é um resultado considerado satisfatório para os autores.

A conclusão deste trabalho foi a mesma apresentada do artigo anteriormente citado, ou seja, o algoritmo não supervisionado torna-se o mais adequado para a classificação dos usuários no *Scientia.net*. Isso se deve ao fato dos algoritmos supervisionados exigirem o conhecimento antecipado das classes nas quais os usuários serão classificados para a implementação do treinamento. Foi também destacado o modelo de rotulagem, onde foi trabalhado o reconhecimento das classes por meio dos algoritmos não supervisionados. Além disso, os resultados foram considerados satisfatórios, levando em consideração que a maioria dos *clusters* avaliados na base foram rotulados com alta taxa de sucesso, com média de mais de 90%. Por fim, é possível ter uma melhora na classificação dos usuários no *Scientia.net*, levando em consideração o trabalho anterior (Lima e Machado, 2012), onde os resultados obtidos indicam uma melhora na classificação usando *Konohen Network*.

Com base na revisão de literatura encontrada e aqui apresentada, é possível pontuar várias questões. Analisando o ano de publicação dos artigos, por exemplo, pode-se verificar que o artigo mais antigo encontrado na Revisão Sistemática foi no ano de 2012. Isso pode indicar que estudos de Redes Sociais Científicas com Reconhecimento de Padrões começaram a ser apresentados neste período. Algo interessante que pode ser comentado está no artigo de Huang e Yuan (2012), onde eles afirmam na conclusão que o estudo deles foi um dos primeiros relacionados a análise de citação usando o algoritmo *K-Means* no *Google Scholar*.

Analisando os artigos apresentados nesta RSL, pode-se verificar os seguintes casos: 1) uso de árvores de decisão para analisar a produção e o impacto de perfis no *Google Scholar* (ORTEGA, 2015); 2) uso das Redes Neurais para identificar correlações entre os dados de perfil dos usuários e seus seguidores no *Research Gate* (ALHEYASAT, 2016); 3) uso do *K-Means* para verificar padrões de citações, a correlação entre índices métricos, os padrões de citação pessoal dos pesquisadores e a transformação dos temas de pesquisa ao longo do tempo no *Google Scholar* (HUANG; YUAN, 2012); 4) uso de algoritmos de Aprendizado de Máquina na rede social *Scientia.net* para verificar qual deles tem melhor desempenho na classificação dos usuários (LIMA; MACHADO, 2012); 5) classificação de usuários no *Scientia.net* usando o *K-Means* e as Redes Neurais *Multilayer Perceptron* (LIMA; MACHADO; LOPES, 2015).

Avaliando os artigos pelo lado qualitativo, pode-se verificar alguns aspectos. Para melhor organização, cada ponto de vista a ser analisado apresentará o ponto de vista geral

(caso necessário) e os pontos de vista específicos de cada artigo, sendo estes enumerados de acordo com a ordem que foram discutidos no começo desta Seção. A seguir, será apresentada a análise qualitativa dos artigos.

2.3.3 Análise Qualitativa da RSL

Do ponto de vista teórico, em geral, os artigos não realizam uma discussão teórica no sentido de debate de ideias, mas eles apontam autores que os embasaram para desenvolver o trabalho. O referencial teórico encontrado nos trabalhos possuem um foco maior em Computação, trazendo conceitos vindos do Aprendizado de Máquina e os classificadores (*K-Means*, Redes Neurais, Árvores de Decisão, etc). Algumas exceções a serem consideradas estão no artigo de Ortega (2015) e de Huang e Yuan (2012). O artigo de Ortega (2015) apresenta algumas definições de índice/h, que é um índice métrico muito utilizado na área de Ciência da Informação. O artigo de Huang e Yuan (2012) trabalha com um conjunto de índices métricos, como o índice/h e o índice/i10, e os conceitua no trabalho.

Do ponto de vista dos dados usados nos artigos, pode-se citar: 1) Ortega (2015) usou 5 amostras trimestrais compreendidas no período de dezembro de 2011, dezembro de 2012 e uma amostra anual de dezembro de 2013. Mais de 12.000 perfis foram selecionados nessas amostras e, após o processo de limpeza e normalização, 3.034 perfis foram utilizados. Talvez a quantidade de dados poderia ser maior se o autor utilizasse mais dados, pois foi usado apenas uma amostra, que foi selecionada e depois normalizada. 2) Alheyasat (2016) usou uma amostra de 1 milhão de usuários e isso pode ser considerado significativo para o problema, mesmo sendo uma amostra por conveniência, pois a quantidade de dados é maior que todos os artigos discutidos nesta Seção. O autor poderia ter considerado verificar se a amostra é representativa estatisticamente levando em consideração o universo dos dados no *Research Gate* (Rede Social Científica que o autor coletou os dados) e também ao objetivo proposto (identificar a correlação entre os dados de perfil dos usuários e seus seguidores). 3) Huang e Yuan (2012) apresentaram um total de 3.539 pesquisadores, usando um critério de seleção com apenas pesquisadores com mais de 200 citações em pelo menos 10 trabalhos. Levando em consideração seu critério, a quantidade de dados pode ser válida, pois os autores limitaram os dados neste ponto. 4) Lima e Machado (2012) e Lima, Machado e Lopes (2015) apresentaram o universo de 2000 usuários. Como o *Scientia.net* tinha poucos usuários se for comparar com outras Redes Sociais Científicas, a quantidade de dados foi considerada grande levando em conta as limitações do próprio *Scientia.net* em número de usuários.

Resumindo a avaliação dos dados coletados nos artigos, os trabalhos de Lima e Machado (2012), Lima, Machado e Lopes (2015) e de Huang e Yuan (2012) apresentaram dados mais completos, enquanto o de Ortega (2015) pode ser considerado incompleto por ter uma amostra muito pequena (porém, que não inviabiliza sua pesquisa) e o trabalho de Alheyasat (2016) é considerado completo, apesar de ser uma amostra de usuários.

Analisando os métodos usados nos artigos encontrados nesta RSL, pode-se informar o seguinte: 1) O método proposto por Ortega (2015) realiza a coleta de dados no *Google Scholar* e, após isso, faz a seleção dos dados coletando amostras trimestrais em dezembro de 2011 a dezembro de 2012, além de uma anual em dezembro de 2013 e coletam um total de 3.034 perfis que podem ser classificados simultaneamente de acordo com as categorias gênero, posto acadêmico e área de investigação; 2) O artigo de Alheyasat (2016) propõe como método o uso de 2 *web-crowlers* para a coleta de 1 milhão de usuários no *Research Gate*. Um gerava uma lista de *links* e o outro usava a lista para coletar dados e estes eram organizados em tuplas. Foi feito o uso das Redes Neurais para investigar a correlação entre os dados dos perfis dos usuários e o número de seus seguidores; 3) O artigo de Huang e Yuan (2012) tem, como proposta metodológica, uma coleta de dados no *Google Scholar* e foi realizado um agrupamento no qual foram selecionados apenas pesquisadores com mais de 200 citações em pelo menos 10 trabalhos, e isso totalizou 3.539 pesquisadores. Após isso, houve a classificação dos documentos de cada pesquisador organizados em ordem decrescente baseados no número de citação. Depois, foi criado um vetor "10D" e, em cada posição do vetor, foi alocada a proporção de citações dos *papers* mais citados. A primeira posição do vetor agrupa os 10% superiores enquanto que a última posição tem os 10% inferiores; 4) O artigo de Lima e Machado (2012) apresenta um método baseado na coleta de 2000 usuários do *Scientia.net*, divididos em 20 áreas do conhecimento com o objetivo de coletar seus perfis acadêmicos. A base de dados coletada foi replicada 9 vezes com os mesmos registros, mudando a ordem de inserção para executar os algoritmos, a fim de garantir a legitimidade dos resultados já que eles são sensíveis a ordem. Os resultados de cada execução foram reunidos para atingir uma média dos resultados. Além disso, foi necessário definir classes para que os algoritmos supervisionados (Redes Neurais *Multilayer Perceptron* e *Support Vector Machine*) pudessem classificar os dados e para os algoritmos não supervisionados (*Konohen Network* e *K-Means*) foi gerado grupos que foram analisados manualmente para verificar a similaridade de todos os atributos de perfis incluídos em cada grupo; 5) O artigo de Lima, Machado e Lopes (2015) utiliza praticamente o mesmo procedimento metodológico informado no quarto tópico, diferenciando apenas no tratamento dos algoritmos

supervisionados e não supervisionados que, neste caso, foram usados em conjunto. O algoritmo não supervisionado foi utilizado para gerar grupos e formar *clusters*, enquanto que o algoritmo supervisionado foi usado para treinamento dos dados para obter os resultados.

Com base no resumo dos métodos apresentados, nota-se que foi utilizado métodos para classificar usuários de acordo com determinada categoria (ORTEGA, 2015); investigar a correlação entre os dados de perfis de um usuário e o número de seus seguidores (ALHEYASAT, 2016); verificar padrões gerais de citações e sua relação entre métricas de índices, os padrões de citação pessoal dos pesquisadores e a transformação dos temas de pesquisa ao longo do tempo (HUANG; YUAN, 2012); usar classificadores divididos por paradigma para classificar usuários (LIMA; MACHADO, 2012); usar classificadores com paradigmas diferentes, unidos para classificar usuários (LIMA; MACHADO; LOPES, 2015).

Neste sentido, ainda há ausência de métodos que avaliem comparativamente o comportamento dos cientistas em diferentes países, ou que utilize dados de mais de uma Rede Social Científica com um único classificador ou com vários classificadores para comparar resultados. Faltam também estudos que avaliem a popularidade de publicações em uma Rede Social Científica ou diversas comparativamente, etc. Considera-se que ainda existe uma infinidade de métodos que podem ser usados usando Redes Sociais Científicas e Reconhecimento de Padrões.

Levando em consideração o ponto de vista da discussão dos resultados apresentados nos artigos, nota-se que todos falham em uma discussão crítica dos resultados. Ambos os artigos apresentam como conclusão uma análise descritiva dos resultados encontrados, sem apresentar uma discussão teórica ou crítica que aponte claramente para possíveis contribuições ao campo de estudo.

Analisando as limitações dos artigos, pode-se observar que, no geral, todos os artigos apresentados possuem fortes limitações, principalmente metodológicas. Entretanto, é importante esclarecer que as limitações não inviabilizam ou deslegitimam os artigos, já que as limitações são necessárias para delimitar uma pesquisa.

Como limitação, pode ser encontrado o seguinte: 1). O artigo de Ortega (2015), nos resultados, trabalha com áreas como ciências da vida – artes e humanidades e ciências sociais – ciências da saúde – multidisciplinar, o que indica que estas áreas foram classificadas juntas na árvore de decisão. Trabalhando com elas separadamente, talvez outros resultados poderiam ser encontrados. Além disso, o autor usa, durante a coleta de dados, somente os dados obtidos na categoria gênero, posição acadêmica e disciplina, não inserindo mais atributos para alimentar as Árvores de Decisão. O autor também não usa outros classificadores além da

árvore de decisão para verificar diferenças nos resultados e usa somente o *Google Scholar* para coleta de dados. 2). No artigo de Alheyasat (2016), foi coletada uma amostra de um milhão de usuários no *Research Gate*. Não se sabe o motivo de não ter coletado este universo de usuários. O autor usa somente as Redes Neurais para gerar resultados e a Rede Social Científica *Research Gate* para identificar correlações entre os dados dos perfis dos usuários e o número de seus seguidores. 3). O artigo de Huang e Yuan (2012), tem como limitação o uso do algoritmo *k-means* para classificar citações no *Google Scholar* usando as áreas de computação. Pode ser que, com o uso de outros classificadores, novos resultados sejam encontrados, tendo a possibilidade de comparar os resultados encontrados no *k-means*. Os autores também usam somente o *Google Scholar*, podendo utilizar outras redes sociais científicas para ter mais resultados a comparar. O método apresentado neste artigo também é bastante rigoroso, tendo como critério de seleção somente autores com mais de 200 citações em pelo menos 10 trabalhos. 4). O artigo de Lima e Machado (2012) utiliza um total de 4 classificadores (Redes Neurais, *Support Vector Machine*, *K-Means* e *Konohen Network*) separados em 2 paradigmas (o primeiro e segundo classificadores são supervisionados, enquanto o terceiro e o quarto são não-supervisionados). Eles utilizam o universo de 2000 usuários da Rede Social Científica *Scientia.net*. Os resultados do artigo estão focados apenas no desempenho dos algoritmos na classificação dos usuários da rede, tanto individualmente quanto divididos pelo paradigma. Outra limitação encontrada neste trabalho está em relação a Rede Social Científica *Scientia.net*, pois os autores poderiam usar o mesmo método em outras Redes Sociais Científicas para encontrar outros resultados e compará-los. 5). O artigo de Lima, Machado e Lopes (2015) utiliza uma técnica de rotulagem, por meio do uso de um algoritmo não supervisionado (*K-Means*) com outro supervisionado (Redes Neurais) para gerar grupos e classificar usuários no *Scientia.net*. O artigo poderia usar mais modelos de algoritmos não supervisionados e supervisionados para verificar os resultados e compará-los com o modelo aplicado neste trabalho. O uso da Rede Social Científica *Scientia.net* também pode ser considerado uma limitação, pois os autores usam somente ela, sem trabalhar com dados coletados de outras Redes Sociais Científicas para obter resultados diferenciados e compará-los.

Quanto as limitações desta Revisão Sistemática, tentamos utilizar várias bases de dados para coletar o máximo de documentos possíveis que estejam relacionados ao tema Redes Sociais Científicas e Reconhecimento de padrões. Do total de artigos, apenas 5 foram pertinentes o que pode significar uma necessidade de pesquisa urgente neste campo, porém, a revisão sistemática apresentada baseou-se em critérios rigorosos de seleção, deixando de lado,

por exemplo, estudos de Redes Sociais Comuns (*Facebook*, *Google Plus*, entre outros). É possível que existam muitos estudos que envolvem o uso de Reconhecimento de Padrões nestas redes, com métodos que possivelmente podem ter aplicabilidade em Redes Sociais Científicas. Considera-se que esta RSL também não é capaz de avaliar a relevância dos artigos encontrados e das pesquisas feitas, apesar de considerar que os artigos encontrados são os pioneiros nos estudos de Redes Sociais Científicas com Reconhecimento de Padrões.

Por fim, conclui-se neste Capítulo que as Redes Sociais Científicas trazem consequências para a visibilidade e a invisibilidade dos artigos e dos próprios cientistas. Isso ocorre devido a essas Redes se basearem em algoritmos que classificam artigos quanto a popularidade, utilizando uma série de fatores como, por exemplo, quantidade de vezes que um artigo foi lido ou até mesmo verificando o número de seguidores do autor daquele artigo. Um estudo interessante que foi discutido nesta RSL é o trabalho de Alheyasat (2015), que chegou a conclusão de que existe uma alta correlação entre os usuários e seus seguidores por meio de publicação de atividades de pesquisa, indicando que muitos usuários se interessam por outros usuários com maiores habilidades de pesquisa. Esse fato faz com que o autor ganhe certa visibilidade ao publicar seus trabalhos em uma Rede Social Científica e os algoritmos desenvolvidos para definir a popularidade de um artigo nessas redes pode seguir um rumo parecido (é possível que outras variáveis existam além dessas aqui discutidas). Esta dissertação tenta, como um dos objetivos, identificar essas variáveis que indicam que determinado artigo (aqui chamado de *paper*) é popular usando a Rede Social Científica *Mendeley*.

A visibilidade e invisibilidade também podem causar impactos em diversos fatores, como a produção, difusão, divulgação e uso da ciência. Muitos trabalhos acadêmicos publicados nessas redes são pouco lidos, tendo assim pouca popularidade, dificultando a divulgação do trabalho para que novos estudos de determinado tópico de pesquisa venham a surgir.

O Capítulo 3, que será apresentado a seguir, mostrará as definições de aprendizado de máquina e reconhecimento de padrões importantes para a compreensão dos métodos que serão aplicados nessa dissertação.

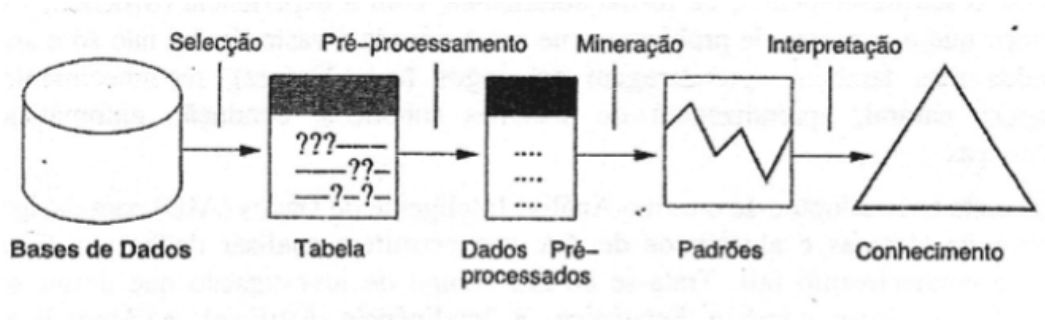
3 RECONHECIMENTO DE PADRÕES E APRENDIZADO DE MÁQUINA: DESENVOLVENDO UMA REVISÃO DE LITERATURA DOS MÉTODOS

O presente Capítulo visa apresentar os conceitos de aprendizado de máquina e reconhecimento de padrões considerados importantes para a compreensão dos procedimentos metodológicos do trabalho. Neste capítulo serão vistas definições de reconhecimento de padrões, o aprendizado de máquina e o paradigma de aprendizagem supervisionada, o método de amostragem *holdout*, as medidas de erro com base em problemas de classificação e o algoritmo Naive Bayes.

3.1 Reconhecimento de Padrões

Para que o conceito de Reconhecimento de Padrões seja apresentado de forma mais clara, segue a Figura 8, onde ilustra o processo de Descoberta de Conhecimento em Bases de Dados (DCBD).

Figura 8 - Processo DCBD.



Fonte: ROCHA *et*. *al*, 2008

Ao observar a Figura 8, é possível perceber uma série de processos que consiste na Descoberta de Conhecimento em Bases de Dados. Segundo Rocha *et*. *al*. (2008), esse processo consiste em transformar dados brutos em conhecimento de alto nível, tendo algumas etapas de desenvolvimento, que são:

- **Seleção** de um Conjunto de Dados sobre um domínio de aplicação e os objetivos do processo DCBD;
- **Pré-processamento** com validação da integridade, limpeza dos dados (tratamento de ruído e informações incompletas) e o uso de técnicas para a redução do número de variáveis;
- **Mineração** de dados (MD), que inclui a adaptação do objetivo de DCBD a métodos

de Análise Inteligente de Dados (AID) (ex. classificação), a escolha do algoritmo de AID e a procura dos padrões de conhecimento;

- **Interpretação**, **verificação** e **avaliação** dos resultados obtidos com os modelos de MD;
- **Utilização e gestão** do conhecimento adquirido.

Também é possível notar, ao observar a Figura 8, que a detecção de padrões está entre a mineração e interpretação. É neste ponto que um algoritmo de AID é utilizado para a classificação dos dados coletados nas etapas de seleção e pré-processamento.

3.2 Aprendizado de Máquina

O aprendizado de máquina (AM) tem varias definições vindas de vários autores. Pode-se citar definições como a de Rocha et. al (2008), que conceitua aprendizado de máquina como a capacidade de um programa de computador aprender por meio de técnicas desenvolvidas. Essas técnicas são os próprios classificadores de aprendizado de máquina, como redes bayesianas ou as redes neurais, que carregam paradigmas e definições que moldam o desenvolvimento desses algoritmos.

Outra definição de aprendizado de máquina pode ser definida por Mitchell (1997), que afirma que o AM se preocupa em construir programas de computador que melhorem automaticamente com a experiência. Faceli et. al (2011), afirma que o aprendizado de máquina é a capacidade de ferramentas computacionais criarem por si próprias hipóteses ou funções que possam resolver determinado problema, por meio da experiência durante o aprendizado automático.

Como é possível observar, os autores citados têm um conceito em comum. Nesse sentido, o aprendizado de máquina pode ser considerado como a capacidade de um programa de computador aprender por meio da experiência através de um conjunto de dados. Esse algoritmo de aprendizado de máquina deve ter a capacidade de induzir hipóteses que resolvam um determinado problema.

Os algoritmos de aprendizado de máquina apresentam um caráter indutivo, ou seja, eles têm a capacidade de induzir uma hipótese através de um conjunto de dados. Segundo Faceli et. al (2011), um algoritmo de AM, por meio de um conjunto de exemplos de treino, deve gerar modelos ou hipóteses que tenham a capacidade de relacionar atributos de entrada de um determinado objeto de um exemplo de treino[26] ao valor de seu atributo de saída. Faceli et. al (2011) também afirma que, para que os algoritmos de AM possam gerar uma hipótese,

[26] Exemplos de treino são os conjuntos de dados que serão alimentados por um algoritmo de AM.

os exemplos de teste[27] devem pertencer ao mesmo problema estudado e não devem fazer parte do conjunto de exemplos de treino.

Os algoritmos de AM apresentam paradigmas de aprendizagem. A próxima Seção apresenta a definição do paradigma de aprendizado supervisionado, que será importante para a compreensão do método usado nesta dissertação.

3.3 Aprendizagem Supervisionada (AS)

Segundo Bishop (2006), a aprendizagem supervisionada são dados em que o conjunto de exemplos de treinamento contém as classes de saída juntamente com os atributos de entrada correspondentes. Conduta e Magrin (2010) definem que no aprendizado supervisionado o objetivo é induzir conceitos a partir de exemplos pré-classificados, ou seja, os exemplos devem conter uma classe de saída conhecida e estruturada.

Rocha *et. al.* (2008) definem a aprendizagem supervisionada através da presença de um "professor", ou seja, cada exemplo apresentado no conjunto de dados contem uma resposta correta (que seria a classe de saída). Neste caso, cada exemplo deve conter seus atributos de entrada e as classes de saída correspondentes.

Conduta e Magrin (2010) categorizam as classes de saída em dois tipos: caso as classes tenham valores discretos, o problema categoriza-se como classificação. Se as classes apresentam valores contínuos, categoriza-se como regressão. A próxima Seção apresenta as medidas de erro aplicadas a problemas de classificação. Também é apresentado o método de amostragem *Holdout*, utilizado para gerar exemplos de treino para classificação em um algoritmo de AM. O conhecimento desses conceitos é importante para que se tenha uma compreensão do método que será apresentado no próximo capítulo.

3.4 Medidas de Erro Para Problemas de Classificação e o Método *Holdout*

As medidas de erro e os métodos de amostragem são de grande importância para verificar a confiabilidade de um classificador e também para apresentar um método que permita uma classificação mais adequada. Em problemas de classificação, uma técnica comum é aplicar a Matriz de Confusão e o PECC (porcentagem de exemplos corretamente classificados).

A Matriz de Confusão baseia-se em mapear os exemplos contidos no treino para verificar quantos foram previstos. A matriz de confusão consiste em uma tabela com valores

[27] Exemplos de teste são os dados utilizados para testes de indução em um algoritmo de AM.

distribuídos (representando o total de exemplos treinados por um classificador) e espera-se uma maior concentração de valores em sua diagonal (onde sempre apresentará a relação negativo-negativo e positivo-positivo) identificando os valores previstos pelo classificador. Esses valores previstos são os exemplos que foram classificados corretamente por um algoritmo de aprendizado de máquina. A Tabela 10 apresenta um modelo simplificado de uma matriz de confusão.

Tabela 10 - Exemplo de Matriz de Confusão 2x2.

	Negativo	**Positivo**
Negativo	**A**	B
Positivo	C	**D**

Fonte: Adaptado de ROCHA *et. al.* 2008

A matriz de confusão é o primeiro passo para definir um modelo de avaliação que, neste caso, é o PECC. Segundo Rocha *et. al.* (2008) o PECC "passa a contar o número de exemplos para os quais o valor previsto para a classe coincide com o valor real (valores da diagonal da matriz)". O PECC geralmente é normalizado em termos de porcentagem, dividindo pelo número total de exemplos. O cálculo a seguir, que foi definido por Rocha *et. al.* (2008), apresenta um breve modelo de cálculo do PECC, baseando-se na matriz de confusão apresentada na figura acima:

$$\mathrm{PECC} = \frac{\mathrm{a} + \mathrm{d}}{\mathrm{a} + \mathrm{b} + \mathrm{c} + \mathrm{d}} * 100\% \qquad (\mathbf{3.1})$$

Neste caso, percebe-se que o PECC apresenta, basicamente, a porcentagem dos exemplos de treino que foram classificados corretamente por um algoritmo de AM baseando-se na matriz de confusão.

Como foi visto anteriormente, os algoritmos de AM devem ter a capacidade de indução. Com isso, buscam-se meios para transformar alguns exemplos de treino em exemplos de teste. Com isso, surgem os métodos de amostragem e o mais popular entre eles é o *Holdout*. Rocha et. al (2008) confirmam que este método é considerado o mais popular e "baseia-se em uma divisão dos dados iniciais do problema em exemplos de treino, para uso do algoritmo de aprendizagem, e exemplo de teste para estimar o erro cometido". Rocha et. al (2008) também afirmam que esse método possui como principal vantagem a simplicidade no

desenvolvimento e a rapidez na execução. Porém, ele pode ter algumas desvantagens uma vez que é sensível à forma como é feita a divisão dos exemplos de treino em exemplos de teste.

3.5 Algoritmo Naive Bayes

O algoritmo Naive Bayes é considerado um dos algoritmos mais simples, porém completo, para classificação de dados. Seu conceito baseia-se na teoria das probabilidades que, segundo Rocha *et. al.* (2008) é considerada uma das principais fontes técnicas para classificação de bases de dados.

A ideia do algoritmo baseia-se em calcular as probabilidades associadas a um exemplo para cada classe de saída, com base no uso de frequências de co-ocorrência de cada atributo do conjunto de dados de treino.

Rocha *et. al.* (2008) citam um breve exemplo de utilização do algoritmo Naive Bayes baseado em um problema de jogar ao ar livre levando em consideração as questões climáticas. A base de dados de treino deste problema é apresentado no Quadro 5.

Quadro 5 - Exemplos de treino do problema de jogar ao ar livre.

Aparência	Temperatura(ºF)	Umidade	Vento	Jogar
Limpo	Quente	Alta	Fraco	Não
Limpo	Quente	Alta	Forte	Não
Nublado	Quente	Alta	Fraco	Sim
Chuvoso	Temperada	Alta	Fraco	Sim
Chuvoso	Fria	Normal	Fraco	Sim
Chuvoso	Fria	Normal	Forte	Não
Nublado	Fria	Normal	Forte	Sim
Limpo	Temperada	Alta	Fraco	Não
Limpo	Fria	Normal	Fraco	Sim
Chuvoso	Temperada	Normal	Fraco	Sim
Limpo	Temperada	Normal	Forte	Sim
Nublado	Temperada	Alta	Forte	Sim
Nublado	Quente	Normal	Fraco	Sim
Chuvoso	Temperada	Alta	Forte	Não

Fonte: Adaptado de ROCHA *et al*. 2008, p. 38.

Com base nos exemplos de treino apresentados no Quadro 5, o algoritmo Naive Bayes é alimentado. Esse processo começa com uma contagem, calculando o número de vezes que ocorre cada par atributo, valor para cada possível valor da classe. A Tabela 11 apresenta a contagem finalizada

Tabela 11 - Contagem da ocorrência de pares de atributo como exemplo de cada uma das classes.

Aparência			**Temperatura**			**Umidade**			**Vento**		
	Sim	Não		Sim	Não		Sim	Não		Sim	Não
Limpo	2	3	Quente	2	2	Alta	3	4	Fraco	6	2
Nublado	4	0	Temperada	4	2	Normal	6	1	Forte	3	3
Chuvoso	3	2	Fria	3	1						

Fonte: Adaptado de (ROCHA *et. al.*, 2008, p. 132)

Com base na frequência de cada atributo calculado pelo algoritmo Naive Bayes, é gerada as frequências relativas, dividindo as frequências apresentadas na Tabela 11 pelo número de exemplos da respectiva classe. A Tabela 12 apresenta as frequências relativas do problema.

Tabela 12 - Tabela de Frequências Relativas.

Aparência			**Temperatura**			**Umidade**			**Vento**		
	Sim	Não		Sim	Não		Sim	Não		Sim	Não
Limpo	2/9	3/5	Quente	2/9	2/5	Alta	3/9	4/5	Fraco	6/9	2/5
Nublado	4/9	0/5	Temperada	4/9	2/5	Normal	6/9	1/5	Forte	3/9	3/5
Chuvoso	3/9	2/5	Fria	3/9	1/5						

Fonte: Adaptado de ROCHA *et. al*. 2008, p. 132.

A partir desse momento, o algoritmo Naive Bayes já está treinado. Agora, deve-se testar a capacidade de indução do algoritmo aplicando um exemplo de teste. Supõe-se que o exemplo a ser testado terá as seguintes configurações:

“aparência” = “chuvoso”, “temperatura” = “fria”, “umidade” = “alta”, “vento” = “forte”.

A classificação deste exemplo passa por calcular para cada classe o valor de uma função de pertença L (*likelihood*). O *likelihood* é obtido multiplicando as frequências relativas de cada atributo presente no exemplo de teste, associando a cada classe de saída presente na base de dados.

Para o exemplo dado, tem-se os seguintes valores:

L(“sim”) = 3/9*3/9*3/9*3/9*9/14 = 0,00794

L(“não”) = 3/5*1/5*4/5*3/5*5/14 = 0,01371

Escolhendo o valor mais alto para a função L, a classe atribuída a este exemplo seria “não”. Pode ainda, se pretendido, atribuir uma probabilidade a cada uma das classes anteriores, normalizando os valores de L de modo que a sua soma seja 1. Assim, a probabilidade (P) de cada uma das classes será:

P(“sim”) = L(“sim”)/ (L(“sim”) + L(“não”)) = 0,367 (37%)

P(“não”) = L(“não”)/ (L(“sim”) + L(“não”)) = 0,633 (63%)

Com esses procedimentos, é possível realizar a classificação da base com o algoritmo Naive Bayes.

O Capítulo 4, que será apresentado a seguir, mostrará os materiais e métodos desta dissertação, exibindo todo o processo desde a coleta até a obtenção dos dados formatados para serem executados no algoritmo Naive Bayes que apresenta o paradigma de aprendizagem supervisionada.

4 MATERIAIS E MÉTODOS: DESCOBERTA DE CONHECIMENTO NA BASE DE DADOS MENDELEY

O presente capítulo apresenta o procedimento metodológico utilizado no processo de coleta e seleção para gerar os dados necessários para a obtenção dos resultados, bem como as técnicas que foram utilizadas no processo de tratamento dos dados. Neste caso, apresenta-se a coleta e seleção dos dados, o pré-processamento e a mineração, considerados fundamentais para obtenção do conjunto de exemplos de treino que mais tarde será alimentado pelo algoritmo Naive Bayes.

4.1 Coleta e Seleção dos Dados

A coleta dos dados foi realizada por meio de um algoritmo desenvolvido com base no Mendeley API. A plataforma Mendeley permite a coleta de dados, desde que haja uma autenticação entre o usuário e o servidor do Mendeley. Por isso, o algoritmo desenvolvido promove a autenticação e inicia a coleta automática. Esta coleta baseia-se em usar os métodos da API do Mendeley[28] para realizar consultas para retornar dados referentes aos documentos, como título, tipo do documento, ano de publicação, resumo, palavras-chave, entre outros. Toda a coleta é gerada por uso de links que são criados conforme instrução disponível no site do Mendeley para desenvolvedores. No caso desta dissertação, foi utilizado o método *search catalog*[29]. Ele permite uma coleta geral de documentos, sendo necessária a utilização de um parâmetro específico, que pode ser *title* (título do documento), *author* (autor de algum documento), *source* (nome da publicação) ou *abstract* (resumo do documento). Neste caso, utilizamos o *source*, colocando como parâmetro as palavras *Proceedings* e *Journal*, por acreditar que elas podem retornar uma quantidade considerável de documentos. Com isso, dividimos *Proceedings* e *Journal* em quatro categorias de dados. O Quadro 6 apresenta as categorias utilizadas, juntamente com o link utilizado para consulta.

[28] Disponível em: <http://dev.mendeley.com/>. Acesso em: 24/01/2018.
[29] Disponível em: <http://dev.mendeley.com/methods/#catalog-search>. Acesso em: 24/01/2018.

Quadro 6 - Categorias de dados com seus respectivos links de consulta.

Categorias de dados	Links de Consulta
Proceedings_Open	https://api.mendeley.com/search/catalog?source=proceedings&open_access=true&view=all&limit=100&access_token=
Proceedings	https://api.mendeley.com/search/catalog?source=proceedings&view=all&limit=100&access_token=
Journal_Open	https://api.mendeley.com/search/catalog?source=journal&open_access=true&view=all&limit=100&access_token=
Journal	https://api.mendeley.com/search/catalog?source=journal&view=all&limit=100&access_token=

Fonte: elaboração própria

Observando o Quadro 6, é possível verificar que há um parâmetro chamado *access_token* sem valor. É neste parâmetro que é colocado um *token* para acessar os dados da API do Mendeley. Sem um *token*, a consulta não retornará um resultado. Um *token* de acesso pode ser obtido de duas formas: 1) criando uma aplicação que permita a comunicação com o servidor do Mendeley para solicitar um *token*; 2) por meio de uma aplicação demonstrativa do Mendeley que pode ser encontrada no site para desenvolvedores[30]. A segunda opção é a mais simples de todas, pois é necessário apenas um cadastro no Mendeley para receber o *token* de acesso. É importante informar que os *tokens* têm validade de uma hora para expirar. Por isso que os links disponibilizados no Quadro 6 não apresentam *tokens*.

Outro fator importante a ser comentado no Quadro 6 é em relação às categorias. Observando os links das categorias *Proceedings_Open* e *Journal_Open*, é possível encontrar um parâmetro chamado *open_access*. Ele é responsável por retornar apenas documentos que tenham acesso aberto no Mendeley. Por padrão, este parâmetro é *false*, ou seja, se ele não for usado durante a consulta, todos os documentos retornados não serão de acesso aberto.

Outra questão importante a ser comentada no Quadro 6 é que, o fato das pesquisas terem como principal parâmetro as palavras *Proceedings* e *Journal*, não significa dizer que os documentos encontrados serão somente *Proceedings* e *Journal*. É preciso notar que estas palavras estão dando valor ao parâmetro *source* (nome da publicação), ou seja, se a fonte de publicação de qualquer documento apresentar a palavra *Proceedings* ou *Journal*, o documento associado a essa fonte de publicação será retornado nas consultas. O parâmetro que define o tipo de documento é o *Type*, que será visto em breve como um dos parâmetros utilizados nesta dissertação durante o processo de seleção e tratamento dos dados.

Por fim, os links apresentados no Quadro 6 retornaram uma série de documentos. O universo desses documentos será apresentado na Tabela 13, juntamente com as categorias de dados.

[30] Disponível em: <https://mendeley-show-me-access-tokens.herokuapp.com/>. Acesso em: 24/01/2018

Tabela 13 - Categorias de dados com o total de documentos encontrados.

Categorias de dados	Universo	Data da última verificação
Proceedings Open	3.416	26/07/2017
Proceedings	1.696.118	26/07/2017
Journal Open	815.794	26/07/2017
Journal	13.575.936	26/07/2017

Fonte: elaboração própria

Se for somado o universo de todas as categorias da Tabela 13, obtêm-se um total de 16.091.264 documentos. Em posse desses dados, foi necessário realizar uma série de tratamentos que garantisse que os dados seriam selecionados adequadamente para atender aos critérios estabelecidos pelo algoritmo Naive Bayes. O algoritmo usa um modelo de base de dados da *UCI Machine Learning Repository*[31]. Este site é um repositório de bases de dados de Aprendizado de Máquina desenvolvido pela *University Of California Irvine*. O modelo de bases de dados da UCI se estabelece da seguinte maneira:

- Para atributos nominais: Escrever o tipo de dado e suas características de forma nominal, ordenando-os de menor para o maior dependendo da análise dos dados. Ex: temperatura fraca, forte.
- Para atributos numéricos: Escrever o tipo de dado e suas características de forma numérica, inserindo uma faixa de valor começando pelo menor valor e terminando pelo alto valor, precedido da palavra NUM. Ex: temperatura NUM 0 100.
- Para os exemplos: Escrever cada característica de um exemplo em sequência, a fim de trata-lo como um elemento do conjunto de dados.

O Quadro 7 apresenta os atributos da base de dados que foram escolhidos para o processo de seleção e também as características possíveis de cada um dos dados.

[31] Disponível em: <http://archive.ics.uci.edu/ml/index.php>. Acesso em: 25/12/2017

Quadro 7 - Atributos selecionados na base de dados e suas características.

Atributo	Descrição	Características possíveis
Title	Título do documento	Muito_ruim, ruim, bom, muito_bom, excelente (será comentado a seguir)
Type	Tipo do documento	Depende da base de dados
Source	Lugar de publicação do documento	Depende da base de dados
Year	Ano de publicação do documento	Até 1999: artigo_classico; De 2000 a 2007: artigo_para_revisão_de_literatura; De 2008 a 2011: artigo_de_referência; De 2012 a 2015: estado_da_arte; De 2016 a 2017: saindo_do_forno.
Keywords	Palavras-chave do documento	Muito_ruim, ruim, bom, muito_bom, excelente (será comentado a seguir)
Authors	Autores que desenvolveram o documento	Muito_ruim, ruim, bom, muito_bom, excelente (será comentado a seguir)
Month	Mês de publicação do documento	Nome por extenso do mês correspondente no documento. Os meses são ordenados do menor para o maior nos atributos (Mês com menor recorrencia de publicações para o de maior recorrencia na base)
Abstract	Resumo do documento	Muito_ruim, ruim, bom, muito_bom, excelente (será comentado a seguir)
Reader Count	Quantidade de leitores do documento. Também é o atributo que define as classes de saída da base de dados	Não_popular, pouco_popular, popular, muito_popular, extremamente_popular (será comentado a seguir).

Fonte: elaboração própria

Observando o Quadro 7, é possível notar os atributos que foram selecionados para gerar os resultados. Em geral, esses atributos foram escolhidos devido à suposição de que eles podem auxiliar como indicadores, a fim de identificar um artigo popular. As características possíveis foram pensadas como estratégia, para que o algoritmo Naive Bayes tenha a capacidade de identificar e aprender com os dados que serão usados nele após todo o processo de tratamento dos dados.

Conforme Quadro 7, apresenta-se a justificativa de utilização de cada um dos atributos.

Title: Foi escolhido por que se acredita que o tema de pesquisa de um autor pode influenciar na popularidade de um artigo. Temas atuais ou mais comentados podem ter mais chances de serem vistos. A escolha das características desse atributo, conforme Quadro 7, foi apenas para nomear títulos que apresentassem determinada pontuação. Neste caso, o "muito_ruim" tem pontuações baixas, enquanto o "excelente" tem pontuações altas. Essas pontuações variam dependendo da categoria de dados, pois são baseados nos contadores de frequência para

palavras de um título na base de dados inteira. Esse processo de execução, assim como dos demais atributos serão vistos mais à frente.

Type: Foi escolhido pois o tipo do documento pode ajudar na visibilidade do assunto. Artigos de revistas têm visibilidade diferente de um livro, por exemplo. Quanto às características possíveis, os tipos de documento dependerão da base de dados.

Source: Acredita-se que o nome da publicação pode influenciar na popularidade de um documento. Artigos publicados em revistas ou conferências renomadas podem ter mais chances de serem lidos. Suas características dependerão da base de dados.

Year: O ano de publicação serve para descobrir o motivo de determinado documento ter sido popular. Artigos clássicos podem ter mais popularidade possivelmente por serem mais lidos ou artigos publicados recentemente podem ganhar popularidade devido ao tema ser bastante comentado. As características do ano foram pensadas visando sua publicação. Artigos clássicos apresentam pontuações menores do que documentos recentes.

Keywords: As palavras-chave podem influenciar no acesso a um documento, principalmente se forem palavras frequentes Neste caso, elas podem ajudar um documento a ter mais visibilidade. Suas características foram pensadas pelo mesmo motivo do atributo *title.*

Authors: O nome do autor também pode ser um indicador forte para que um documento seja popular. Cientistas já conhecidos podem ter um grande volume de acessos em um documento em menor tempo. Suas características foram pensadas pelo mesmo motivo do atributo *title.*

Month: O mês de publicação é importante para saber a frequência de publicações dos documentos em determinados meses do ano. Essas frequências podem ser interessantes para descobrir os meses em que a maioria das revistas realizam suas publicações. Suas características foram pensadas para dar mais valor em meses que tenham uma publicação mais recente.

Abstract: O resumo pode possibilitar a leitura de um documento, uma vez que ele descreve um trabalho. Nesse sentido, pode existir a possibilidade de um resumo ajudar na visibilidade de um documento. Suas características foram pensadas pelo mesmo motivo do atributo *title*.

Reader Count: Esse atributo é particular do Mendeley. Ele retorna o contador de leitores de um documento. Neste caso, a quantidade de leitores pode ser um grande indicador de popularidade (sendo aqui considerado como acesso). Apesar do *reader count* apresentar características com nomes diferentes do atributo *title*, sua ideia foi pensada da mesma forma que neste atributo. As características de todos os atributos listados servem apenas para que o algoritmo Naive Bayes tenha a capacidade de reconhecer os dados para poder classificá-los.

Para que um documento da base de dados seja selecionado, todos os atributos presentes no Quadro 8 devem, obrigatoriamente, estarem preenchidos. Caso contrário, determinado documento não poderá ser selecionado para iniciar o processo de classificação. Com a seleção concluída, cada categoria de dados apresentou um subconjunto. A Tabela 14 apresenta novamente as categorias de dados com o universo de documentos, juntamente com o subconjunto e a porcentagem de cada subconjunto em relação ao universo de documentos.

Tabela 14 - Categorias de dados com universo, subconjunto e porcentagem do subconjunto em relação ao universo.

Categorias de dados	Universo	Subconjunto	Amostra %
Proceedings Open	3.416	361	10,56%
Proceedings	1.696.118	74.264	4,37%
Journal Open	815.794	168.286	20,63%
Journal	13.575.936	3.806.870	28,04%

Fonte: elaboração própria

Observando a Tabela 14, é possível perceber que, além das porcentagens dos subconjuntos de cada categoria não serem tão próximas, elas também são pequenas em relação ao universo. Uma das possibilidades para esta situação pode ser devido ao atributo *month*, que no Mendeley é considerado um *aditional field* (campo adicional). Ou seja, é possível que muitos autores, ao cadastrarem um documento no Mendeley, não coloquem o mês de publicação. Porém, para este trabalho, o atributo mês é fundamental para saber a frequência das publicações durante os meses do ano.

Com os dados selecionados, é necessária a remoção de ruídos que podem estar presentes na amostra como documentos repetidos, por exemplo. A próxima Seção apresenta a etapa de pré-processamento, que tem a função de trabalhar na limpeza dos dados.

4.2 Pré-Processamento

A fase de pré-processamento é algo importante no desenvolvimento de Reconhecimento de Padrões porque, com esse processo, é possível identificar e remover dados repetidos ou qualquer ruído presente nos dados que pode influenciar no processo de aprendizagem dos algoritmos de Inteligência Artificial. No caso dos documentos coletados no Mendeley, além de documentos repetidos, foram encontrados dados com meses irregulares (que utilizaram um mês fora do padrão) e até mesmo anos irregulares (documentos com data de publicação acima do ano vigente da coleta de dados ou até mesmo com datas de dois ou três séculos atrás). Esses erros geralmente são produzidos por falha humana, ou seja, quando um autor de determinado documento publica no Mendeley informando dados errados. O

tratamento para esse tipo de problema foi a remoção desses documentos. A Tabela 15 apresenta a quantidade de documentos que foram repetidos, com meses irregulares e anos irregulares para cada categoria de dados e o total de documentos que foram usados após a remoção.

Tabela 15 - Total de documentos usados após o pré-processamento de dados.

Categorias de dados	Documentos Repetidos	Docs. Com anos irregulares	Docs. Com meses irregulares	Total
Proceedings_Open	2	0	0	359
Proceedings	3.519	93	27	70.615
Journal_Open	1.803	18	15	166.450
Journal	452.821	1.354	1.282	3.351.413

Fonte: elaboração própria

Com a remoção dos documentos, a amostra de cada categoria de dados teve uma redução. A Tabela 16 apresenta as categorias de dados com o universo de documentos, juntamente com a amostra e a porcentagem de cada amostra em relação ao universo de documentos após a remoção.

Tabela 16 - Categorias de dados com universo, amostra e porcentagem da amostra em relação ao universo após remoção de documentos.

Categorias de dados	Universo	Amostra	Amostra %
Proceedings_Open	3.416	359	10,51%
Proceedings	1.696.118	70.615	4,16%
Journal_Open	815.794	166.450	20,40%
Journal	13.575.936	3.351.413	24,68%

Fonte: elaboração própria

Observando a Tabela 16, percebe-se que não houve uma grande redução nos dados, principalmente comparando as porcentagens apresentadas nesta tabela com a Tabela 14, tendo como uma pequena exceção à categoria *Journal*, com redução de, aproximadamente, 4%. Com isso, a Tabela 16 apresenta o subconjunto que será utilizado nos demais processos até gerar os arquivos com os dados transformados para classificação no algoritmo Naive Bayes conforme foi apresentado no Quadro 7. A próxima Seção apresenta a mineração dos dados, etapa fundamental para adaptar os dados ao classificador Naive Bayes.

4.3 Mineração dos dados

A etapa de Mineração de dados é responsável por tratar os dados selecionados na etapa de pré-processamento para serem adaptados ao classificador Naive Bayes. Para isso, a fase de mineração de dados foi dividida em duas etapas, ambas de forma automática por meio de algoritmos.

Na primeira etapa, foi realizada uma pré-adaptação onde foram selecionados somente os atributos que teriam processamentos repetidos. Essa repetição leva em consideração, por exemplo, contagem de frequências de palavras na base de dados (*title, keywords, authors e abstract*), ordenação das palavras para posterior remoção de palavras comuns, como preposições (*title* e *abstract*) e soma imediata das frequências após contagem de palavras (*authors*).

Levando em consideração as etapas de pré-adaptação informadas, somente *title, keywords, authors* e *abstract* tinham procedimentos repetidos. Os demais atributos da base de dados eram exclusivos e não havia necessidade de estarem presentes na pré-adaptação. Esse procedimento foi realizado pensando-se na possibilidade de criar categorias de classes de saída para apresentar nos resultados. Em caso de necessidade de apresentar mais categorias de classes, não seria necessário repetir sempre os processos de pré-adaptação para cada categoria de classe de saída uma vez que a base de dados é imutável para cada categoria de dados (ver o Quadro 7 para saber quais são as categorias de dados usadas).

Para a segunda etapa, foi realizado o processo de adaptação final respeitando a individualidade de cada atributo. O Quadro 8 apresenta os atributos e os procedimentos de adaptação para cada um.

Quadro 8 - Procedimentos de adaptação para cada atributo.

Atributo	Procedimentos de Adaptação
Title	Pré-adaptação: Contagem de frequências para cada palavra, ordenação de palavras (decrescente)[32] Adaptação final: Soma das frequências de cada palavra em um título, discretização
Type	Pré-adaptação: - Adaptação final: Contagem de frequências para cada type, ordenação de palavras (crescente)[33]
Source	Pré-adaptação: - Adaptação final: Contagem de frequências para cada source, ordenação de palavras (crescente)
Year	Pré-adaptação: - Adaptação final: Identificação de cada ano e verificação de sua classe de acordo com os critérios estabelecidos na Tabela 11
Keywords	Pré-adaptação: Contagem de frequências para cada palavra-chave de um documento, soma das frequências de todas as palavras-chave de um documento. Adaptação final: discretização.
Authors	Pré-adaptação: Contagem de frequências para cada autor de um documento, soma das frequências de todos os autores de um documento. Adaptação final: discretização.
Month	Pré-adaptação: - Adaptação final: Distribuição de frequência de cada mês encontrado, ordenação dos meses (crescente), escrever por extenso de acordo com o número do mês. Ex: 1 – janeiro.
Abstract	Pré-adaptação: Contagem de frequências para cada palavra, ordenação de palavras (decrescente) Adaptação final: Soma das frequências de cada palavra em um abstract, discretização.
Reader Count	Pré-adaptação: - Adaptação final: Discretização.

Fonte: elaboração própria

Observando o Quadro 8, existe um processo em que muitos atributos passam durante a adaptação final, chamado de discretização. Esse processo baseia-se em estabelecer faixas de valores para atributos numéricos, uma vez que o algoritmo Naive Bayes não suporta atributos deste tipo. As faixas de valores são uma forma de adaptar os dados para o algoritmo Naive Bayes.

Como exemplo, é possível supor a discretização em *title*. Observando o Quadro 7, é possível perceber que *title* pode ter até cinco valores possíveis: "muito_ruim, ruim, bom, muito_bom, excelente". Ao fazer a distribuição de frequência de todas as palavras em *title* somadas, supõe-se que o menor valor encontrado de um título encontrado na base de dados é 1 e o maior é 25. Ao iniciar o processo de discretização, a nomeação dos dados será da seguinte maneira:

[32] A ordenação decrescente é importante para que ocorra a remoção de palavras desnecessárias, como preposições ou pronomes. Geralmente, essas palavras sempre têm mais ocorrência. Por isso foi decidido ordenar de forma decrescente para facilitar a remoção.

[33] A ordenação crescente é necessária, pois o algoritmo Naive Bayes trabalha com um reconhecimento de atributos nominais em sequencia crescente. Nesse sentido, as primeiras palavras da ordem são as de menor recorrência, enquanto que as últimas são as mais recorrentes da base.

- De 1 a 5: muito_ruim
- De 6 a 10: ruim
- De 11 a 15: bom
- De 16 a 20: muito_bom
- De 21 a 25: Excelente

Este foi o primeiro modelo de discretização usado para classificar os documentos quanto à popularidade. Este modelo de discretização apresentou alguns problemas de distribuição na classe de saída, uma vez que, com distribuições iguais, a maioria dos documentos eram alocados para a classe de saída "não_popular", e pouquíssimos documentos tinham alocações nas demais classes. Com isso, houve a ideia de distribuir a discretização em porcentagens para que seja possível ajustar os parâmetros de distribuição sempre que possível para que as classes de saída tivessem uma distribuição maior ou mais equilibrada.

Antes de apresentar a distribuição das classes de saída após a aplicação da discretização em porcentagem, é necessário mostrar a configuração das categorias de dados. Em geral, as quatro categorias que estão sendo mostradas desde o início deste capítulo (*Proceedings_Open*, *Proceedings*, *Journal_Open* e *Journal*) foram divididas em 3 subcategorias chamadas de: "cinco classes de saída", "três classes de saída" e "duas classes de saída". A alteração nas classes de saída é apenas para verificar se o aumento ou diminuição nas classes interfere no desempenho do algoritmo Naive Bayes. O Quadro 9 apresenta a distribuição das classes de saída para as 3 subcategorias apresentadas.

Quadro 9 - Subcategorias com as suas respectivas classes possíveis.

Subcategorias	Classes possíveis
Cinco Classes de Saída	Não_popular, pouco_popular, popular, muito_popular, extremamente_popular
Três Classes de Saída	Não_popular, popular, extremamente_popular
Duas Classes de Saída	Não_popular, extremamente_popular

Fonte: elaboração própria

Com isso, o modelo de discretização em porcentagem será apresentado a seguir com base nas 3 subcategorias apresentadas no Quadro 9 para cada categoria de dados. Esse modelo consiste em distribuir em forma de porcentagem cada documento a uma respectiva classe de saída baseando-se na quantidade de leitores (esta que nomeia as classes de saída em questão). Os valores em porcentagens foram obtidos buscando na base de dados o documento com maior quantidade de leitores. Após isso, para cada documento, é realizado um cálculo que converte o valor numérico em porcentagem para, em seguida, alocar na faixa de valor

correspondente. O cálculo para transformar os valores numéricos em porcentagem se deu da seguinte maneira:

$$D = \frac{(V * 100)}{M} \qquad (\mathbf{4.1})$$

Onde:

D: Valor discretizado em porcentagem

V: O valor numérico que deseja discretizar

M: O maior valor da base de dados.

Em termos práticos, o cálculo apresentado acima funciona como uma regra de três, onde o valor contido em M representa 100% e o valor em V está para D, que é o valor de discretização em porcentagem que se deseja calcular. Se aplicar o mesmo exemplo apresentado anteriormente (fazer a discretização em *title*, onde o menor valor encontrado na base de dados é 1 e o maior é 25), supõe-se que determinado exemplo a ser discretizado tenha *title* com pontuação 15. Neste caso, tem-se os seguintes valores conforme a legenda acima: V=15, M=25. Sendo assim, o cálculo de discretização em porcentagem se dará da seguinte maneira:

$$D = \frac{(15 * 100)}{25}$$

Com isso, o valor de D para este caso será 60%. Sua alocação em uma característica na base de dados dependerá da classe de saída da própria. Isso será visto adiante, apresentando a distribuição das porcentagens aplicadas nesta dissertação.

Começando com a categoria *Proceedings_Open*, a distribuição das porcentagens na discretização se estabeleceu da seguinte maneira na subcategoria “cinco classes de saída”, conforme Tabela 17.

Tabela 17 - Distribuição da discretização na categoria *Proceedings_Open* e subcategoria "cinco classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 1,6%	0 a 1,04	179	1
Pouco_Popular	>1,6% a 4%	>1,04 a 2,6	77	2
Popular	>4% a 7%	>2,6 a 4,55	60	3,36
Muito_Popular	>7% a 12%	>4,55 a 6,5	22	5,54
Extremamente_Popular	>12% a 100%	>6,5 a 65	21	17,9

Fonte: elaboração própria

Observando a Tabela 17, percebe-se que há a coluna chamada frequência média de popularidade. Ela se baseia na média entre a quantidade de leitores de cada classe de saída com o total de documentos. Esta coluna traz a média para saber onde se concentra a maior parte de contador de leitores para cada classe de saída. A coluna discretização (%) foi obtida por meio de testes exaustivos na base de dados até encontrar um ajuste que permita o equilíbrio de distribuição ou então uma distribuição descrescente (não_popular apresentando maior quantidade de documentos, seguido por pouco_popular com a segunda maior quantidade e assim segue), conforme apresentada na coluna total de documentos.

A Tabela 18 apresenta a distribuição para a subcategoria "três classes de saída" em *Proceedings_Open*.

Tabela 18 - Distribuição da discretização na categoria *Proceedings_Open* e subcategoria "três classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 1,6%	0 a 1,04	179	1
Popular	>1,6% a 6%	>1,04 a 3,09	115	2,33
Extremamente_Popular	>6% a 100%	>3,09 a 65	65	9.01

Fonte: elaboração própria

Para finalizar a categoria *Proceedings_Open*, é apresentada a Tabela 19 mostrando a distribuição da discretização para a subcategoria "duas classes de saída".

Tabela 19 - Distribuição da discretização na categoria *Proceedings_Open* e subcategoria "duas classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 4%	0 a 2,6	253	1,3
Extremamente_Popular	>4% a 100%	>2,6 a 65	103	6,79

Fonte: elaboração própria

A categoria *Proceedings* também passou pelos mesmos procedimentos de *Proceedings_Open*. A Tabela 20 apresenta a distribuição para a subcategoria "cinco classes de saída" em *Proceedings*.

Tabela 20 - Distribuição da discretização na categoria *Proceedings* e subcategoria "cinco classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,5%	0 a 10,38	51.795	3,49
Pouco_Popular	>0,5% a 2%	>10,38 a 41,54	13.647	20,29
Popular	>2% a 5%	>41,54 a 103,85	3.728	63,55
Muito_Popular	>5% a 8%	>103,85 a 166,16	840	129,19
Extremamente_Popular	>8% a 100%	>166,16 a 2077	605	307,77

Fonte: elaboração própria

A Tabela 21 apresenta a distribuição para a subcategoria "três classes de saída" em *Proceedings*.

Tabela 21 - Distribuição da discretização na categoria *Proceedings* e subcategoria "três classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,5%	0 a 10,38	51.795	3,49
Popular	>0,5% a 2%	>10,38 a 41,54	13.647	20,29
Extremamente_Popular	>2% a 100%	>41,54 a 2077	5.173	102,77

Fonte: elaboração própria

Finalizando a categoria *Proceedings*, é apresentada a Tabela 22 com a distribuição para a subcategoria "duas classes de saída".

Tabela 22 - Distribuição da discretização na categoria *Proceedings* e subcategoria "duas classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,5%	0 a 10,38	51.795	3,49
Extremamente_Popular	>0,5% a 100%	>10,38 a 2077	18.820	42,96

Fonte: elaboração própria

A categoria *Journal_Open* também passou pelo mesmo tratamento das demais categorias. A Tabela 23 apresenta a distribuição de *Journal_Open* para a subcategoria "cinco classes de saída".

Tabela 23 - Distribuição da discretização na categoria *Journal_Open* e subcategoria "cinco classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,035%	0 a 3,2	60.211	1,82
Pouco_Popular	>0,035% a 0,08%	>3,2 a 7,4	40.069	5,30
Popular	>0,08% a 0,15%	>7,4 a 13,9	28.263	10,11
Muito_Popular	>0,15% a 0,3%	>13,9 a 27,9	23.033	18,97
Extremamente_Popular	>0,3% a 100%	>27,9 a 9326	14.184	59,51

Fonte: elaboração própria

A Tabela 24 apresenta a distribuição para a subcategoria "três classes de saída" em *Journal_Open*.

Tabela 24 - Distribuição da discretização na categoria *Journal_Open* e subcategoria "três classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,06%	0 a 5,59	83.448	2,55
Popular	>0,08% a 0,3%	>5,59 a 27,9	68.128	12,20
Extremamente_Popular	>0,3% a 100%	>27,9 a 9326	14.184	59,51

Fonte: elaboração própria

Finalizando as subcategorias de *Journal_Open*, é apresentada a Tabela 25 com a subcategoria "duas classes de saída"

Tabela 25 - Distribuição da discretização na categoria *Journal_Open* e subcategoria "duas classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,1%	0 a 9,3	112.316	3,77
Extremamente_Popular	>0,1% a 100%	>9,3 a 9326	54.134	27,82

Fonte: elaboração própria

Assim como em *Journal_Open*, a categoria *Journal* também passou pelo mesmo procedimento. A Tabela 26 apresenta a distribuição de *Journal* para a subcategoria "cinco classes de saída".

Tabela 26 - Distribuição da discretização na categoria Journal e subcategoria "cinco classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,17%	0 a 16,9	1.516.506	2,21
Pouco_Popular	>0,17% a 0,5%	>16,9 a 49,79	960.451	30,97
Popular	>0,5% a 0,9%	>49,79 a 89,60	433.574	55,82
Muito_Popular	>0,9% a 2%	>89,60 a 199,12	328.187	108,53
Extremamente_Popular	>2% a 100%	>199,12 a 9956	112.695	207,18

Fonte: elaboração própria

A Tabela 27 apresenta a distribuição para a subcategoria "três classes de saída" em *Journal*.

Tabela 27 - Distribuição da discretização na categoria *Journal* e subcategoria "três classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0,25%	0 a 24,89	1.702.893	2,98
Popular	>0,25% a 0,8%	>24,89 a 1991,2	1.015.640	30,74
Extremamente_Popular	>0,8% a 100%	>1991,2 a 9956	832.880	1993,7

Fonte: elaboração própria

Finalizando as subcategorias de *Journal*, é apresentada a Tabela 28 com a subcategoria "duas classes de saída".

Tabela 28 - Distribuição da discretização na categoria Journal e subcategoria "duas classes de saída".

Classes de Saída	Discretização (%)	Discretização (qtd)	Total de Documentos	Frequência Média de Popularidade
Não_Popular	0% a 0.5%	0 a 49,78	2.286.075	2,98
Extremamente_Popular	>0.5% a 100%	>49,78 a 9956	1.065.338	55,40

Fonte: elaboração própria

Com base nas tabelas apresentadas é possível perceber que, apesar da distribuição das classes de saída não estarem tão equilibradas, elas estão ordenadas do maior para o menor de acordo com a classe de menor popularidade para a de maior popularidade. Acredita-se que os dados configurados nesse sentido podem refletir a realidade das bases de dados em geral, em que a maioria dos documentos não apresenta tantos leitores (por isso a classe "não_popular" em todas as distribuições de faixas na discretização apresentou maior quantidade de documentos). A distribuição em porcentagem ajudou a melhorar todas as classes de saída, visto que, se fosse a uma distribuição igualitária, a classe "não_popular" iria ter a grande maioria dos documentos, correndo o risco das demais classes apresentarem pouquíssimos ou até mesmo nenhum valor. Todas as faixas de porcentagens apresentadas nas tabelas foram obtidas com base em testes exaustivos até encontrar uma faixa que mantenha os dados equilibrados ou com uma ordem de grandeza decrescente de distribuição (a classe "não_popular" obtém mais documentos que a classe "pouco_popular", que por sua vez tem mais documentos que a classe "popular" e assim segue).

Com o procedimento de discretização, é possível transformar atributos numéricos em nominais, fazendo com que seja possível o reconhecimento desses atributos pelo algoritmo Naive Bayes. Todos os processos de discretização, pré-adaptação e adaptação final são feitos automaticamente por meio de algoritmos que foram desenvolvidos para esta finalidade.

A mineração, assim como os demais tratamentos mencionados neste capítulo, foram essenciais para gerar os resultados da dissertação. Isso será apresentado no próximo capítulo, onde será visto como o algoritmo Naive Bayes irá se comportar ao fazer induções nas bases de dados coletadas e tratadas.

5 RESULTADOS

Neste Capítulo serão apresentados todos os resultados obtidos com o algoritmo Naive Bayes a partir do conjunto de exemplos de treino. Esse conjunto baseou-se nas categorias de dados (*Proceedings_Open, Proceedings, Journal_Open e Journal*), todas divididas em 3 subcategorias (cinco classes de saída, três classes de saída e duas classes de saída). Após a avaliação técnica dos resultados obtidos, será feita uma discussão, com o objetivo de tentar identificar atributos que ajudem a reconhecer artigos populares no Mendeley.

5.1 Aplicação do Método de Amostragem e Exemplos de Treino

Como foi visto no Capítulo 4, foi usado o método *Holdout* para realizar os testes de indução no algoritmo Naive Bayes. Esse método baseou-se em coletar aleatoriamente um exemplo de treino para cada classe de saída com o intuito de virar exemplo de teste para o algoritmo Naive Bayes iniciar o processo de indução.

A seleção aleatória feita pelo *Holdout* diferenciou para cada subcategoria. Isso porque as subcategorias apresentam quantidades de classes de saída diferentes (ver Tabela 4.8 no Capítulo 4 onde são apresentadas as classes de saída possíveis para cada subcategoria). Neste caso, para a subcategoria “cinco classes de saída”, o método selecionou aleatoriamente 5 exemplos de treino para cada classe. Na subcategoria “três classes de saída” foram selecionadas apenas 3 exemplos e na subcategoria “duas classes de saída” selecionou-se 2 exemplos. A Tabela 29 apresenta o total de exemplos de teste contidos em cada subcategoria após o *Holdout* para cada categoria de dados.

Tabela 29 - Categorias de dados com as respectivas subcategorias e exemplos de treino antes e depois do *Holdout*.

Categoria de dados	Subcategorias	Qtd. De exemplos de treino antes do Holdout	Qtd. De exemplos de treino após Holdout
Proceedings_Open	Cinco classes de saída	359	354
	Três classes de saída		356
	Duas classes de saída		357
Proceedings	Cinco classes de saída	70.615	70.610
	Três classes de saída		70.612
	Duas classes de saída		70.613
Journal_Open	Cinco classes de saída	166.450	166.445
	Três classes de saída		166.447
	Duas classes de saída		166.448
Journal	Cinco classes de saída	3.351.413	3.351.408
	Três classes de saída		3.351.410
	Duas classes de saída		3.351.411

Fonte: elaboração própria

Como é possível observar na Tabela 29, a quantidade de exemplos usados para treino não varia tanto para cada categoria de dados individualmente, ou seja, a redução dos exemplos de treino é mínima. A partir de agora, serão vistos os resultados em todas as categorias de dados.

5.2 Resultados da categoria *Proceedings_Open*

Nesta Seção serão apresentados todos os resultados da categoria *Proceedings_Open*. Na subcategoria "cinco classes de saída", como medidas de erro, apresenta-se a matriz de confusão e o PECC, na Tabela 30.

Tabela 30 - Matriz de Confusão e PECC para a subcategoria "cinco classes de saída" em *proceedings_open*.

	Negativo	**Positivo**	**Negativo**	**Positivo**	**Negativo**
Negativo	**131**	21	10	12	4
Positivo	46	**24**	6	0	1
Negativo	27	6	**17**	6	3
Positivo	7	2	1	**9**	2
Negativo	6	3	3	0	**8**
PECC 53%					

Fonte: elaboração própria

Na matriz de confusão apresentada na Tabela 30, todos os elementos localizados na diagonal da matriz (em negrito) representam a quantidade de exemplos que foram corretamente classificados pelo algoritmo Naive Bayes. No total, foram 189 exemplos, que representam cerca de 53% da base de dados total (mostrado pelo PECC). Essa porcentagem é considerada pequena para classificação de dados, uma vez que a classificação seria praticamente um booleano, tendo altas chances de erro.

Após o cálculo das medidas de erro, apresenta-se, no Quadro 10, os atributos utilizados para classificação no algoritmo Naive Bayes como exemplos de teste.

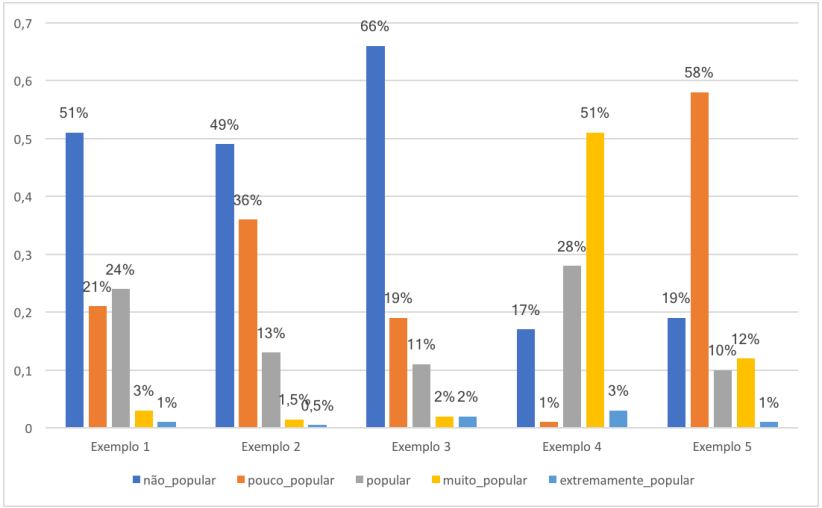
Quadro 10 - Exemplos de teste utilizados na subcategoria "cinco classes de saída" em *Proceedings_Open*.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Ruim	Journal	proceedings_of_the_indian_academy_of_sciences___chemical_sciences	Artigo_classico	Muito_ruim	Muito_ruim	fevereiro	ruim
2	Ruim	Journal	proceedings_of_the_indian_academy_of_sciences___chemical_sciences	Artigo_classico	Muito_ruim	Muito_ruim	abril	ruim
3	Muito_ruim	Conference_proceedings	Bmc_proceedings	Artigo_de_referencia	Muito_ruim	Ruim	Janeiro	ruim
4	Bom	journal	Bmc_proceedings	Artigo_de_referencia	Muito_bom	Muito_ruim	Junho	ruim
5	Ruim	Journal	Bmc_proceedings	Artigo_para_revisao_de_literatura	Muito_ruim	Ruim	Dezembro	ruim

Fonte: elaboração própria

Algo importante ao observar no Quadro 10 é o número do exemplo. Neste caso, o exemplo 1 apresenta uma configuração de documento "não_popular", o exemplo 2 tem a classe "pouco_popular", o exemplo 3 é "popular", o exemplo 4 é "muito_popular" e o exemplo 5 é "extremamente_popular". É importante informar que, para o algoritmo Naive Bayes, as classes dos exemplos de teste são desconsideradas, pois o algoritmo faz uma estimativa com base no treinamento da base de dados. Porém, os resultados desta dissertação levarão em consideração as classes dos exemplos de teste em casos onde o PECC é baixo, a fim de trazer uma confiabilidade maior nesses casos. Os resultados dos exemplos de teste serão apresentados a seguir, na Figura 9.

Figura 9 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria "cinco classes de saída" em *Proceedings_Open*.



Fonte: elaboração própria

Observando a Figura 9, é possível perceber que o algoritmo previu os exemplos 1, 2 e 3 como “não_popular”, enquanto que o exemplo 4 apresentou a classe “muito_popular” e o exemplo 5 a classe “pouco_popular”.

A seguir, a Tabela 31 apresenta a matriz de confusão e PECC para a subcategoria “três classes de saída”.

Tabela 31 - Matriz de Confusão e PECC para a subcategoria “três classes de saída” em *proceedings_open*.

	Negativo	**Positivo**	**Negativo**
Negativo	**124**	20	34
Positivo	63	**32**	19
Negativo	20	7	**37**
PECC 54%			

Fonte: elaboração própria

Para esta subcategoria, o PECC apresentou ser um pouco maior que na subcategoria “cinco classes de saída”, com diferença de apenas 1%. Para esta subcategoria, foram classificados corretamente 193 exemplos de treino (54% da base de dados conforme PECC).

O Quadro 11 apresenta os atributos utilizados para classificação na subcategoria “três classes de saída”.

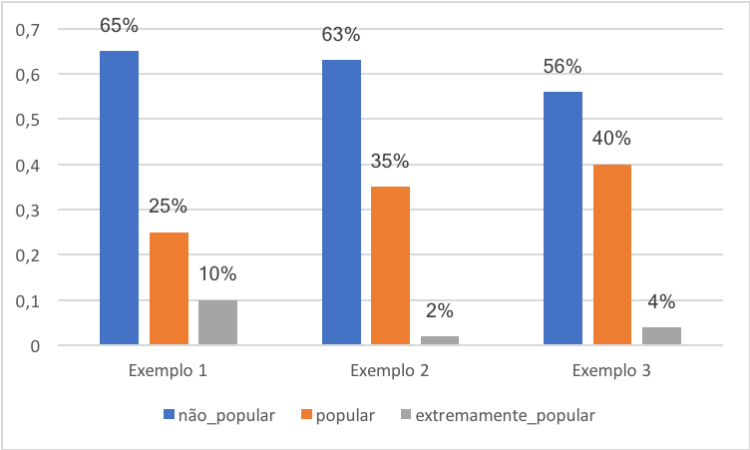
Quadro 11 - Exemplos de teste utilizados na subcategoria “três classes de saída” em *Proceedings_Open*.

Exemplos	**Title**	**Type**	**Source**	**Year**	**Keywords**	**Authors**	**Month**	**Abstract**
1	Muito_ruim	Conference_proceedings	proceedings_of_national_aviation_university	Artigo_de_referencia	Muito_ruim	bom	março	Muito_ruim
2	bom	Journal	proceedings_of_the_indian_academy_of_sciences___chemical_sciences	Artigo_classico	Muito_ruim	Muito_ruim	dezembro	Muito_ruim
3	bom	journal	proceedings_of_the_indian_academy_of_sciences___chemical_sciences	Artigo_classico	Muito_ruim	Muito_ruim	abril	bom

Fonte: elaboração própria

O Quadro 11 apresenta os 3 exemplos utilizados para classificação no algoritmo Naive Bayes. O exemplo 1 tem como saída a classe “não_popular”, o exemplo 2 tem a classe “popular” e o exemplo 3 é da classe “extremamente_popular”. A Figura 10 apresenta os resultados dos exemplos de teste após execução no algoritmo Naive Bayes.

Figura 10 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria "três classes de saída" em *Proceedings_Open*.



Fonte: elaboração própria

Observando a Figura 10, é possível perceber que o algoritmo previu todos os exemplos de teste como "não_popular".

Por fim, será apresentada a subcategoria "duas classes de saída" do *Proceedings_Open*. A Tabela 32 apresenta a matriz de confusão e PECC para esta subcategoria.

Tabela 32 - Matriz de Confusão e PECC para a subcategoria "duas classes de saída" em *proceedings_open*.

	Negativo	**Positivo**
Negativo	**214**	41
Positivo	56	**43**
PECC 73%		

Fonte: elaboração própria

Para esta subcategoria, é possível perceber que houve um aumento significativo no PECC se for comparar com as subcategorias "três classes de saída" e "cinco classes de saída" em *Proceedings_Open*. O algoritmo ainda apresenta uma margem de erro, porém está menor para este caso. Para esta subcategoria, foram classificados corretamente 257 exemplos de treino (73% da base de dados conforme PECC apresentado).

A seguir, será apresentado o Quadro 12, com os exemplos de teste utilizados nesta subcategoria.

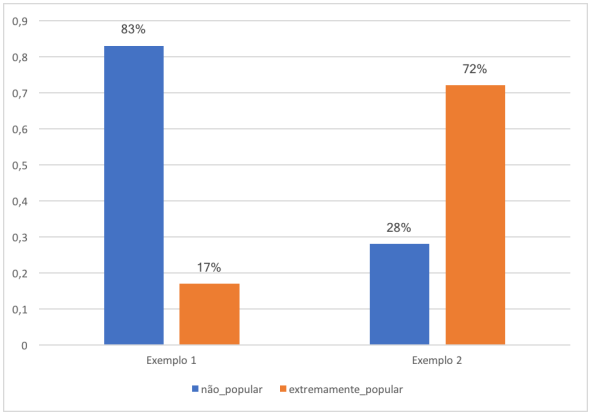
Quadro 12 - Exemplos de teste utilizados na subcategoria "duas classes de saída" em Proceedings_Open

Exemplos	**Title**	**Type**	**Source**	**Year**	**Keywords**	**Authors**	**Month**	**Abstract**
1	excelente	journal	proceedings_of_the_indian_academy_of_sciences___chemical_sciences	Artigo_classico	Muito_ruim	Muito_ruim	maio	Muito_ruim
2	Muito_ruim	Journal	Bmc_proceedings	Artigo_de_referência	Muito_ruim	Muito_ruim	novembro	Muito_ruim

Fonte: elaboração própria

No Quadro 12, o exemplo 1 contém a classe “não_popular” e o exemplo 2 tem a classe “extremamente_popular”. A Figura 11 apresenta o resultado após execução no algoritmo Naive Bayes.

Figura 11 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria “três classes de saída” em *Proceedings_Open*.



Fonte: elaboração própria

Ao observar a Figura 11, percebe-se que o algoritmo previu o exemplo 1 como “não_popular” e o exemplo 2 como “extremamente_popular”.

5.3 Resultados da categoria *Proceedings*

Nesta Seção será visto todos os resultados da categoria *Proceedings*. Começando pela subcategoria “cinco classes de saída” será apresentada, na Tabela 35, a matriz de confusão e PECC.

Tabela 33 - Matriz de Confusão e PECC para a subcategoria “cinco classes de saída” em *proceedings*.

	Negativo	**Positivo**	**Negativo**	**Positivo**	**Negativo**
Negativo	**50.009**	1.572	212	1	1
Positivo	11.242	**2.192**	213	0	0
Negativo	2.961	549	**216**	1	1
Positivo	666	99	65	**9**	1
Negativo	493	71	37	0	**4**
		PECC	74%		

Fonte: elaboração própria

Observando a Tabela 33, percebe-se que o PECC está mais alto se for comparar com o *Proceedings_Open* na mesma subcategoria. Acredita-se que o motivo para isso acontecer vem devido a quantidade de dados, que é bem maior neste caso do que em *Proceedings_Open*. Uma quantidade grande de exemplos de treino é interessante, pois assim o algoritmo terá mais dados para aprender, podendo fazer induções com maior precisão. Nesta subcategoria foram classificados corretamente 52.430 exemplos (74% da base de dados segundo PECC).

O Quadro 13 apresenta os exemplos de teste utilizados para a subcategoria “cinco classes de saída”.

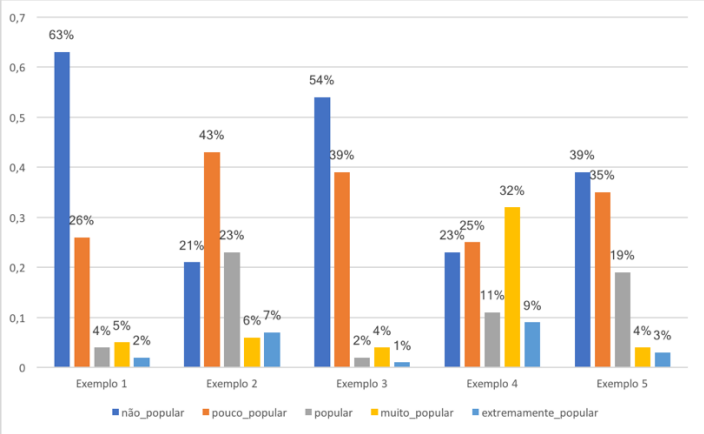
Quadro 13 - Exemplos de teste utilizados na subcategoria “cinco classes de saída” em *Proceedings*.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Muito_ ruim	Journal	proceedings_ of_the_nation al_academy_ of_sciences	Artigo_ classico	Muito_ruim	Muito_ru im	março	Muito_ruim
2	Muito_ Ruim	Journal	proceedings_ of_the_nation al_academy_ of_sciences	Artigo_ para_re visao_d e_litera tura	Muito_ruim	Muito_ru im	março	Muito_ruim
3	Muito_ ruim	Journal	proceedings_ of_the_nation al_academy_ of_sciences_ of_the_united _states_of_a merica	Artigo_ classico	Muito_ruim	Muito_R uim	dezembro	Muito_ruim
4	Muito_ ruim	journal	proceedings_ biological_sci ences___the_ royal_society	Artigo_de eferência	Muito_ruim	Muito_ru im	fevereiro	Muito_ruim
5	ruim	Journal	proceedings_ of_the_nation al_academy_ of_sciences_ of_the_united _states_of_a merica	Estado _da_art e	Muito_ruim	Muito_ru im	junho	Muito_ruim

Fonte: elaboração própria

No Quadro 13, a ordem dos exemplos segue da mesma maneira que em *Proceedings_Open*, ou seja, o exemplo 1 é da classe “não_popular”, o exemplo 2 pertence a classe “pouco_popular”, o exemplo 3 contem a classe “popular”, o exemplo 4 tem a classe “muito_popular” e o exemplo 5 é da classe “extremamente_popular”. A Figura 12 apresenta os resultados desses exemplos após a execução no algoritmo Naive Bayes.

Figura 12 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria "cinco classes de saída" em *Proceedings*.



Fonte: elaboração própria

Observando a Figura 12, percebe-se que o algoritmo Naive Bayes previu os exemplos 1, 3 e 5 como "não_popular", o exemplo 2 como "pouco_popular" e o exemplo 4 como "muito_popular".

Em seguida, será mostrada a subcategoria "três classes de saída" para *Proceedings*. A Tabela 34 apresenta a matriz de confusão e o PECC para esta subcategoria.

Tabela 34 - Matriz de Confusão e PECC para a subcategoria "três classes de saída" em *proceedings*.

	Negativo	**Positivo**	**Negativo**
Negativo	**50.217**	1.018	560
Positivo	11.296	**1.767**	584
Negativo	3.978	558	**637**
PECC 75%			

Fonte: elaboração própria

Observando a Tabela 34 e comparando com os resultados da subcategoria "cinco classes de saída" de *Proceedings*, percebe-se que não há grandes mudanças no PECC, tendo uma diferença de apenas 1%. Porém, quanto maior o PECC, maior será a capacidade do algoritmo Naive Bayes no processo de indução. Para este caso, foram classificados corretamente 52.621 (75% da base de dados segundo o PECC).

O Quadro 14 apresenta os exemplos utilizados na subcategoria "três classes de saída" para *Proceedings*.

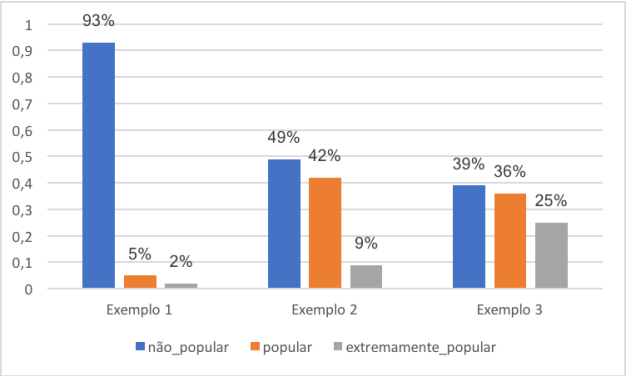
Quadro 14 - Exemplos de teste utilizados na subcategoria "três classes de saída" em *Proceedings*.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Muito_ruim	Journal	aip_conference_proceedings	Artigo_de_referencia	Muito_ruim	Muito_ruim	novembro	Muito_ruim
2	Muito_ruim	Journal	proceedings_of_the_national_academy_of_sciences_of_the_united_states_of_america	Artigo_classico	Muito_ruim	Muito_ruim	março	Muito_ruim
3	bom	journal	proceedings_of_the_national_academy_of_sciences_of_the_united_states_of_america	Artigo_classico	Muito_ruim	Muito_ruim	março	Muito_ruim

Fonte: elaboração própria

Com base na numeração dos exemplos apresentados no Quadro 14, tem-se as seguintes classes de saída: o exemplo 1 contém a classe "não_popular", o exemplo 2 tem a classe "popular" e o exemplo 3 tem a classe "extremamente_popular". A Figura 13 apresenta os resultados gerados pelo algoritmo Naive Bayes nesses 3 exemplos de teste.

Figura 13 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria "três classes de saída" em *Proceedings*.



Fonte: elaboração própria

Como é possível observar na Figura 13, o algoritmo Naive Bayes previu todos os exemplos de teste como "não_popular".

Para a subcategoria "duas classes de saída", apresenta-se a Tabela 35 com a matriz de confusão e PECC.

Tabela 35 - Matriz de Confusão e PECC para a subcategoria "duas classes de saída" em *proceedings*.

	Negativo	Positivo
Negativo	**42.404**	9.391
Positivo	7.196	**11.624**
PECC	77%	

Fonte: elaboração própria

Com base na matriz de confusão e PECC apresentados na Tabela 35, percebe-se que a subcategoria “duas classes de saída” apresentou uma taxa de erro menor se for comparar com as demais subcategorias em *proceedings*. Para esta subcategoria foram classificados corretamente 54.058 exemplos (77% conforme PECC).

O Quadro 15 apresenta os exemplos de teste utilizados para classificação no algoritmo Naive Bayes.

Quadro 15 - Exemplos de teste utilizados na subcategoria “duas classes de saída” em *Proceedings*.

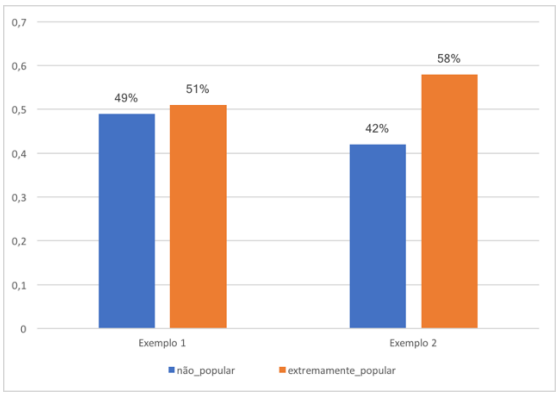
Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Muito_ruim	journal	proceedings_of_the_national_academy_of_sciences_of_the_united_states_of_america	Artigo_classico	Muito_ruim	Muito_ruim	abril	Muito_ruim
2	Muito_ruim	Journal	proceedings_of_the_national_academy_of_sciences_of_the_united_states_of_america	Artigo_de_referência	Muito_ruim	Muito_ruim	dezembro	Muito_ruim

Fonte: elaboração própria

No Quadro 15, os exemplos de teste coletados a base continham as seguintes classes de saída: o exemplo 1 tem a classe “não_popular” e o exemplo 2 é da classe “extremamente_popular”.

A Figura 14 apresenta os resultados dos exemplos de teste após execução no algoritmo Naive Bayes.

Figura 14 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria “duas classes de saída” em *Proceedings*.



Fonte: elaboração própria

Observando a Figura 14, percebe-se que o algoritmo Naive Bayes previu ambos os exemplos como “extremamente_popular”. O Exemplo 1 foi classificado como “extremamente_popular” com uma diferença pequena da classe “não_popular” de apenas 2%. A próxima Seção apresentará os resultados da categoria *journal_open.*

5.4 Resultados da categoria *journal_Open*

Nesta Seção serão vistos todos os resultados da categoria *journal_open.* Começando pela categoria "cinco classes de saída", apresenta-se a Tabela 36 com a matriz de confusão e o PECC.

Tabela 36 - Matriz de Confusão e PECC para a subcategoria "cinco classes de saída" em *journal_open*.

	Negativo	**Positivo**	**Negativo**	**Positivo**	**Negativo**
Negativo	**46.353**	7.926	2.440	2.239	1.252
Positivo	22.770	**9.780**	3.476	2.985	1.057
Negativo	11.547	6.536	**4.379**	4.221	1.579
Positivo	6.189	4.015	3.458	**6.077**	3.293
Negativo	2.323	1.268	1.322	3.821	**6.139**
			PECC 44%		

Fonte: elaboração própria

Observando a Tabela 36, percebe-se que o PECC é baixo se for comparado com as demais categorias na subcategoria "cinco classes de saída". Para esta subcategoria foram classificados corretamente 72.728 exemplos (44% da base de dados segundo PECC).

O Quadro 16 apresenta os exemplos de teste que foram utilizados no algoritmo Naive Bayes para testar a indução.

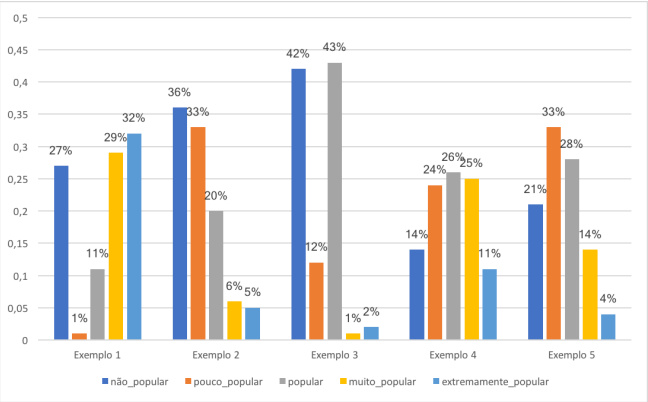
Quadro 16 - Exemplos de teste utilizados na subcategoria "cinco classes de saída" em *journal_open*.

Exemplos	**Title**	**Type**	**Source**	**Year**	**Keywords**	**Authors**	**Month**	**Abstract**
1	Excelente	Journal	international_journal_of_clinical_and_health_psychology	Artigo_de_referência	Excelente	Muito_ruim	janeiro	Excelente
2	Excelente	Journal	korean_journal_of_radiology	Artigo_para_revisão_de_literatura	Excelente	Muito_ruim	abril	Excelente
3	Muito_ruim	Journal	world_journal_of_diabetes	Artigo_classico	Muito_ruim	Muito_Ruim	dezembro	Muito_ruim
4	Muito_ruim	journal	proceedings_biological_sciences___the_royal_society	Estado_da_art	Excelente	Ruim	outubro	Excelente
5	Excelente	Journal	journal_of_biomedical_science	Artigo_para_revisão_de_literatura	bom	ruim	janeiro	Excelente

Fonte: elaboração própria

O Quadro 16 está organizada da mesma maneira que as demais. Nesse sentido, tem-se as seguintes classes que foram coletadas após a aplicação do *Holdout*: o exemplo 1 tem a classe "não_popular", o exemplo 2 tem a classe "pouco_popular", o exemplo 3 tem a classe "popular", o exemplo 4 contem a classe "muito_popular" e o exemplo 5 apresenta a classe "extremamente_popular". A Figura 15 apresenta os resultados encontrados no algoritmo após a execução dos exemplos de teste.

Figura 15 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria "cinco classes de saída" em *journal_open*.



Fonte: elaboração própria

Observando a Figura 15, percebe-se que o algoritmo classificou o exemplo 1 como "extremamente_popular", o exemplo 2 como "não_popular", os exemplos 3 e 4 como "popular" e o exemplo 5 foi classificado como "pouco_popular",

A seguir, apresenta-se os resultados da subcategoria "três classes de saída". A Tabela 37 apresenta a matriz de confusão e o PECC desta subcategoria.

Tabela 37 - Matriz de Confusão e PECC para a subcategoria "três classes de saída" em *journal_open*.

	Negativo	**Positivo**	**Negativo**
Negativo	**59.313**	23.540	594
Positivo	22.859	**43.238**	2.030
Negativo	1.528	9.802	**3.543**
	PECC 64%		

Fonte: elaboração própria

Comparando com o PECC da subcategoria anterior, percebe-se que esta subcategoria apresentou um PECC maior. Neste caso, foram classificados corretamente 106.094 (64% da base de dados segundo PECC).

O Quadro 17 apresenta os exemplos de teste utilizados para esta subcategoria.

Quadro 17 - Exemplos de teste utilizados na subcategoria "três classes de saída" em *journal_open*.

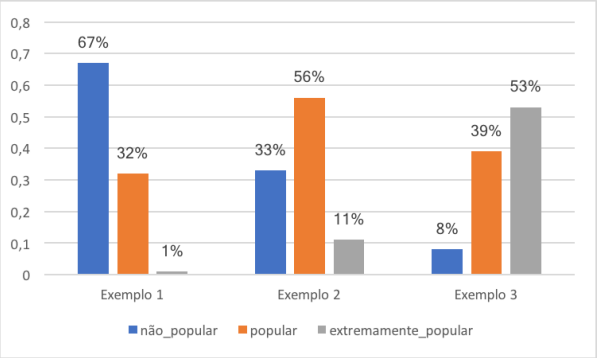
Exemplos	**Title**	**Type**	**Source**	**Year**	**Keywords**	**Authors**	**Month**	**Abstract**
1	Excelente	Journal	american_journal_of_environmental_sciences	Saindo_do_forno	Muito_ruim	Muito_ruim	maio	Excelente
2	Excelente	Journal	world_journal_of_gastroenterology	Artigo_de_referência	Bom	Muito_ruim	maio	Excelente
3	Excelente	generic	international_journal_of_environmental_research_and_public_health	Artigo_de_referência	Excelente	Muito_ruim	dezembro	Excelente

Fonte: elaboração própria

Para os exemplos de teste apresentados no Quadro 17, tem-se as seguintes classes: o exemplo 1 tem a classe "não_popular", o exemplo 2 contem a classe "popular" e o exemplo 3

é da classe “extremamente_popular”. A Figura 16 apresenta os resultados obtidos após a execução dos exemplos de teste no algoritmo Naive Bayes.

Figura 16 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria “três classes de saída” em *journal_open*.



Fonte: elaboração própria

Observando a Figura 16, percebe-se que o algoritmo classificou o exemplo 1 como “não_popular”, o exemplo 2 como “popular” e o exemplo 3 como “extremamente_popular”.

Por fim, será apresentado os resultado da subcategoria “duas classes de saída” em *journal_open*. A Tabela 38 apresenta a matriz de confusão e o PECC para esta subcategoria.

Tabela 38 - Matriz de Confusão e PECC para a subcategoria “duas classes de saída” em *journal_open*.

	Negativo	**Positivo**
Negativo	**98.316**	13.989
Positivo	25.643	**28.490**
PECC 76%		

Fonte: elaboração própria

Observando a Tabela 38 e realizando o somatório dos elementos na diagonal da matriz, tem-se 126.803 exemplos classificados corretamente (76% segundo PECC). Este PECC, comparando com as demais subcategorias de *journal_open*, apresenta-se maior. Quanto maior o PECC, mais capacidade o algoritmo terá em classificar exemplos com mais precisão.

O Quadro 18 apresenta os exemplos de teste utilizados para a subcategoria “duas classes de saída” em *journal_open*.

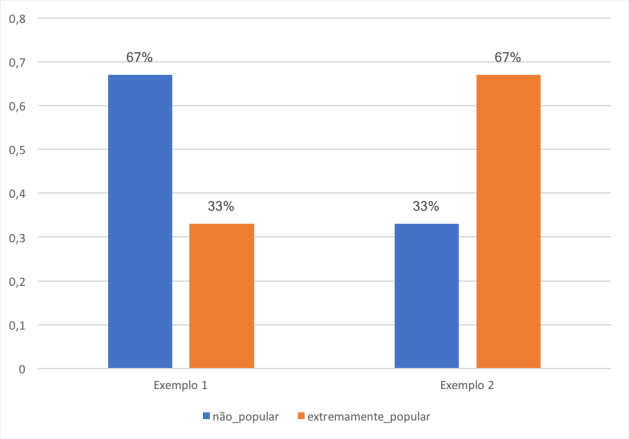
Quadro 18 - Exemplos de teste utilizados na subcategoria “duas classes de saída” em *journal_open*.

Exemplos	**Title**	**Type**	**Source**	**Year**	**Keywords**	**Authors**	**Month**	**Abstract**
1	Excelente	journal	international_journal_of_molecular_sciences	Saindo_do_forno	Excelente	Muito_ruim	julho	Excelente
2	Excelente	Journal	international_journal_of_environmental_research_and_public_health	Artigo_de_referência	Muito_ruim	Muito_ruim	janeiro	Excelente

Fonte: elaboração própria

No Quadro 18 tem-se as seguintes classes: o exemplo 1 contem a classe "não_popular" e o exemplo 2 tem a classe "extremamente_popular". A Figura 17 apresenta os resultados obtidos após a execução dos exemplos de teste no algoritmo Naive Bayes.

Figura 17 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria "duas classes de saída" em *journal_open*.



Fonte: elaboração própria

Observando a Figura 17, percebe-se que classificou o exemplo 1 como "não_popular" e o exemplo 2 como "extremamente_popular". A próxima Seção apresenta os resultados da categoria *Journal*.

5.5 Resultados da Categoria *Journal*

Esta Seção apresenta os resultados obtidos para a categoria *Journal*. Começando pela subcategoria "cinco classes de saída", apresenta-se a matriz de confusão e o PECC na Tabela 39.

Tabela 39 - Matriz de Confusão e PECC para a subcategoria "cinco classes de saída" em *journal*.

	Negativo	**Positivo**	**Negativo**	**Positivo**	**Negativo**
Negativo	**683.043**	160.138	42.382	390.240	240.703
Positivo	257.658	**334.004**	204.221	154.326	10.242
Negativo	4.636	16.810	**405.076**	6.645	407
Positivo	52.020	49.866	33.831	**111.331**	81.139
Negativo	31.018	22.390	11.047	6.531	**41.709**
PECC 47%					

Fonte: elaboração própria

Observando a Tabela 39, percebe-se que o PECC é baixo, porém não tanto quanto o *Journal_Open* para esta mesma subcategoria. Para este caso foram classificados corretamente 1.575.163 exemplos (47% da base de dados segundo PECC).

O Quadro 19 apresenta os exemplos de teste que foram utilizados no algoritmo Naive Bayes para testar a indução.

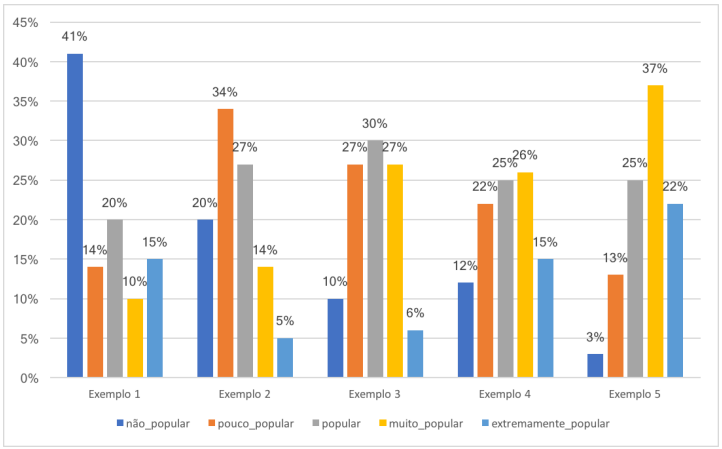
Quadro 19 - Exemplos de teste utilizados na subcategoria “cinco classes de saída” em *journal*.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Ruim	Journal	journal_of_bacteriology	Artigo_classico	Muito_ruim	Muito_ruim	outuro	Muito_ruim
2	Muito_ruim	Journal	faseb_journal	Estado_da_arte	Muito_ruim	Muito_ruim	dezembro	Muito_ruim
3	Ruim	Journal	journal_of_antimicrobial_chemotherapy	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_Ruim	março	Muito_ruim
4	Ruim	journal	journal_of_theoretical_biology	Artigo_para_evisaoao__c literatura	Muito_ruim	Muito_ruim	novembro	Muito_ruim
5	Ruim	Journal	european_journal_of_neuroscience	Artigo_de_referência	Muito_ruim	Muito_ruim	junho	ruim

Fonte: elaboração própria

O Quadro 19 está organizada da mesma maneira que as demais. Nesse sentido, tem-se as seguintes classes que foram coletadas após a aplicação do *holdout*: o exemplo 1 tem a classe “não_popular”, o exemplo 2 tem a classe “pouco_popular”, o exemplo 3 tem a classe “popular”, o exemplo 4 contem a classe “muito_popular” e o exemplo 5 apresenta a classe “extremamente_popular”. A Figura 18 apresenta os resultados encontrados no algoritmo após a execução dos exemplos de teste.

Figura 18 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria “cinco classes de saída” em *journal*.



Fonte: elaboração própria

Observando a Figura 18, percebe-se que o algoritmo classificou o exemplo 1 como “não_popular”, o exemplo 2 como “pouco_popular”, o exemplo 3 como “popular” e os exemplos 4 e 5 como “muito_popular”,

A seguir, apresenta-se os resultados da subcategoria “três classes de saída”. A Tabela 40 apresenta a matriz de confusão e o PECC desta subcategoria.

Tabela 40 - Matriz de Confusão e PECC para a subcategoria "três classes de saída" em *journal*.

	Negativo	**Positivo**	**Negativo**
Negativo	**1.060.913**	420.061	221.919
Positivo	154.063	**739.921**	121.656
Negativo	189.649	416.461	**226.770**
PECC	60,5%		

Fonte: elaboração própria

Comparando com o PECC da subcategoria anterior, percebe-se que esta subcategoria apresentou um PECC menor. Neste caso, foram classificados corretamente 2.027.604 (60,5% da base de dados segundo PECC).

O Quadro 20 apresenta os exemplos de teste utilizados para esta subcategoria.

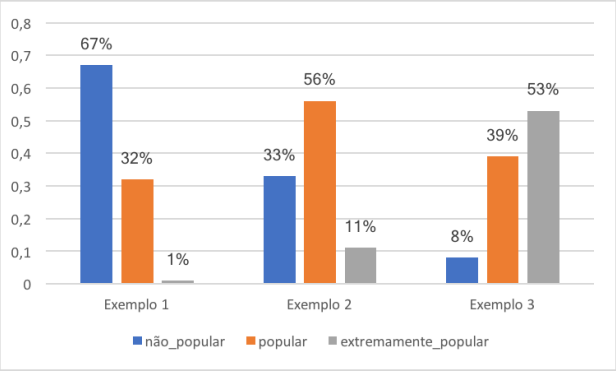
Quadro 20 - Exemplos de teste utilizados na subcategoria "três classes de saída" em *journal*.

Exemplos	**Title**	**Type**	**Source**	**Year**	**Keywords**	**Authors**	**Month**	**Abstract**
1	Muito_ruim	Journal	aha_journal	Estado_da_arte	Muito_ruim	Muito_ruim	setembro	Muito_ruim
2	Muito_ruim	Journal	journal_of_molecular_evolution	Artigo_de_referência	Muito_ruim	Muito_ruim	maio	Muito_ruim
3	Muito_ruim	generic	journal_of_operations_management	Artigo_classico	Muito_ruim	Muito_ruim	novembro	Muito_ruim

Fonte: elaboração própria

Para os exemplos de teste apresentados na Quadro 20, tem-se as seguintes classes: o exemplo 1 tem a classe "não_popular", o exemplo 2 contem a classe "popular" e o exemplo 3 é da classe "extremamente_popular". A Figura 19 apresenta os resultados obtidos após a execução dos exemplos de teste no algoritmo Naive Bayes.

Figura 19 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria "três classes de saída" em *journal*.



Fonte: elaboração própria

Observando a Figura 19, percebe-se que o algoritmo classificou o exemplo 1 como "não_popular", o exemplo 2 como "popular" e o exemplo 3 como "extremamente_popular".

Por fim, será apresentado os resultado da subcategoria "duas classes de saída" em *journal*. A Tabela 41 apresenta a matriz de confusão e o PECC para esta subcategoria.

Tabela 41 - Matriz de Confusão e PECC para a subcategoria “duas classes de saída” em *journal*.

	Negativo	**Positivo**
Negativo	**2.040.763**	245.312
Positivo	525.513	**539.825**
PECC	77%	

Fonte: elaboração própria

Observando a Tabela 41 e realizando o somatório dos elementos na diagonal da matriz, tem-se 2.580.588 exemplos classificados corretamente (76% segundo PECC).

O Quadro 21 apresenta os exemplos de teste utilizados para a subcategoria “duas classes de saída” em *journal*.

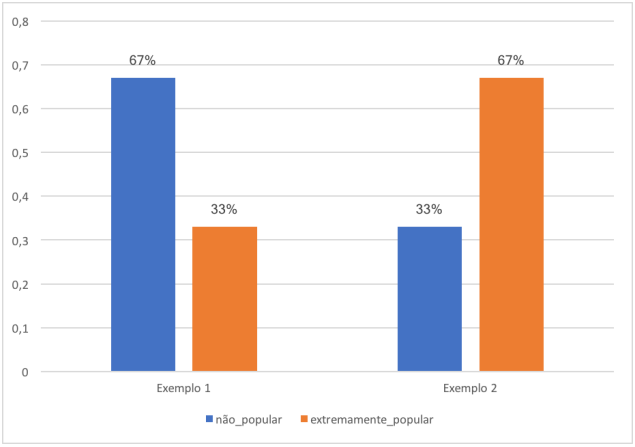
Quadro 21 - Exemplos de teste utilizados na subcategoria “duas classes de saída” em *journal*.

Exemplos	**Title**	**Type**	**Source**	**Year**	**Keywords**	**Authors**	**Month**	**Abstract**
1	Muito_ruim	Journal	netherlands_journal_of_medicine	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_ruim	outubro	Muito_ruim
2	Muito_ruim	Journal	embo_journal	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_ruim	março	Muito_ruim

Fonte: elaboração própria

No Quadro 21, tem-se as seguintes classes: o exemplo 1 contem a classe “não_popular” e o exemplo 2 tem a classe “extremamente_popular”. A Figura 20 apresenta os resultados obtidos após a execução dos exemplos de teste no algoritmo Naive Bayes.

Figura 20 - Resultado dos exemplos de teste após execução no algoritmo Naive Bayes para a subcategoria “duas classes de saída” em *journal*.



Fonte: elaboração própria

Observando a Figura 20, percebe-se que classificou o exemplo 1 como “não_popular” e o exemplo 2 como “extremamente_popular”. Todas as categorias de dados apresentaram resultados melhores nesta subcategoria. A próxima Seção apresenta a discussão dos resultados obtidos nas categorias de dados.

5.6 Discussão dos Resultados

Nesta Seção será feita uma discussão sobre os resultados. Além disso, serão relatadas várias questões referentes a todo o processo que levou aos resultados gerados nesta dissertação, como as limitações do trabalho e a análise qualitativa dos resultados.

Começando pelos resultados deste capítulo, percebe-se que, em todas as categorias de dados, a subcategoria "duas classes de saída" foi a que obteve um melhor PECC (73% para *proceedings_open*, 77% para *proceedings*, 76% para *journal_open* e 77% para *journal*). Portanto, esta subcategoria em todos os casos é considerada a ideal, caso se utilize o mesmo procedimento metodológico desta dissertação aplicando no Mendeley. Acredita-se que esta subcategoria se saiu melhor devido aos dados apresentarem um certo equilíbrio de distribuição nas classes de saída, levando em consideração que todas as categorias de dados, em geral, apresentaram a maioria dos documentos com a classe "não_popular" indicando que, no Mendeley, a maioria dos documentos apresentam uma quantidade pequena de leitores.

Outro fator que pode ter levado a este resultado é a própria base de dados. Em *Proceedings_Open*, a base foi menor em comparação com as demais categorias de dados. Tanto a subcategoria "cinco classes de saída" quanto a "três classes de saída" apresentaram PECC baixos neste caso (53% e 54% respectivamente). Porém, a subcategoria "três classes de saída" apresenta comportamento diferente nas demais categorias de dados (75% em *proceedings*, 64% em *journal_open* e 60,5% em *journal*). De qualquer maneira, o fator da quantidade de documentos na base de dados pode não ser o principal motivo que levou a um PECC baixo, levando em consideração que o PECC para a subcategoria "cinco classes de saída" em *Proceedings* foi de 74% (considerada alta em relação as demais categorias). A questão da distribuição das classes de saída é o principal problema, pois a maioria dos documentos encontrados têm poucos leitores, fazendo com que as classes que representam maior quantidade de leitores tenham uma distribuição menor de documentos. Quando se utiliza muitas classes de saída para esses casos, a distribuição dos documentos fica ainda mais desequilibrada. Existe a possibilidade desse fenômeno ocasionar uma má classificação dos exemplos de treino pelo algoritmo Naive Bayes. Por isso que a subcategoria "duas classes de saída" apresentou melhores PECC, pois a quantidade de classes é menor, ocasionando uma distribuição mais equilibrada.

Com isso, será discutida a questão dos problemas e limitações deste trabalho antes de realizar a análise qualitativa dos resultados. Para começar, pode-se citar a questão do método de amostragem *Holdout*. Este método foi escolhido devido, principalmente, a sua

simplicidade e rapidez, além dele atender ao procedimento metodológico (selecionar aleatoriamente 1 exemplo de treino de cada classe de saída para virar exemplo de teste). Porém, somente o uso do *Holdout* pode ser considerado uma limitação, levando em consideração que existem outros métodos de amostragem que são mais completos (ou até melhores) que o *Holdout*, podendo adaptar-se ao método em questão. Além disso, seria interessante verificar se o uso de outros métodos de amostragem influenciaria no resultado final que é a classificação dos *papers* no algoritmo Naive Bayes.

Uma limitação do trabalho que pode ser considerada está em relação a somente o uso do algoritmo Naive Bayes. O uso de outros algoritmos de Reconhecimento de Padrões poderia ser interessante para poder comparar os resultados obtidos em cada um. Porém, somente o uso do algoritmo Naive Bayes não inviabiliza a pesquisa.

No caso do Mendeley, não se considera uma limitação somente o uso desta plataforma para classificação de documentos, uma vez que existe um atributo considerado importante e que tornou a classe de saída (devido ao algoritmo Naive Bayes apresentar o paradigma de aprendizagem supervisionada) que é a quantidade de leitores. Com base nos estudos de diversos autores apresentados no Capítulo 2 dessa dissertação, é possível chegar a conclusão de que o Mendeley aparenta ser a única plataforma que apresenta um contador de leitores para documentos, podendo trazer estudos baseados em aspectos de leitura. Conforme mencionado no Capítulo 2, segundo Mohamadi *et. al.* (2015), o Mendeley aparenta ser a única opção para revelar aspectos de leitura de artigos de pesquisa. Como era extremamente necessário obter a quantidade de leitores para esta dissertação, o Mendeley de fato aparenta ser a única plataforma a possibilitar a obtenção desses dados de maneira simples. A partir de agora, será apresentada a análise qualitativa dos resultados apresentados neste capítulo.

5.7 Análise Qualitativa dos Resultados

Como foi possível verificar nos resultados, foram apresentados, ao todo, 4 categorias divididas em 3 subcategorias cada. As categorias de dados são: *proceedings*, *proceedings_open*, *journal_open* e *journal*. As subcategorias que foram apresentadas em cada uma são: “cinco classes de saída”, “três classes de saída” e “duas classes de saída”.

Levando em consideração somente as subcategorias e fazendo um cruzamento de resultados somente nelas (ignorando as categorias de dados neste caso) e realizando um somatório das classes de saída usando somente os exemplos que foram classificados corretamente pelo algoritmo Naive Bayes, têm-se o seguinte resultado: na subcategoria “cinco classes de saída”, 3 exemplos foram classificados como “não_popular”, 2 exemplos como

“pouco_popular”, 2 exemplos como “popular”, 3 exemplos como “muito_popular” e nenhum exemplo foi classificado como “extremamente_popular”. Na subcategoria “três classes de saída”, 4 exemplos foram classificados como “não_popular”, 1 exemplo foi classificado como “popular” e 2 exemplos foram “extremamente_popular”. Para a subcategoria “duas classes de saída”, 3 exemplos foram classificados como “não_popular” e 4 exemplos foram classificados como “extremamente_popular”.

Considerando a subcategoria “cinco classes de saída” na classe “não_popular”, tem-se as seguintes características, conforme Quadro 22.

Quadro 22 - Exemplos de teste classificados como “não_popular” na subcategoria “cinco classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	ruim	journal	proceedings_of_the_indian_academic_of_sciences_chemical_sciences	Artigo_classico	Muito_ruim	Muito_ruim	fevereiro	ruim
2	Muito_ruim	Journal	proceedings_of_the_national_academy_of_sciences	Artigo_classico	Muito_ruim	Muito_ruim	março	Muito_ruim
3	Ruim	Journal	journal_of_bacteriology	Artigo_classico	Muito_ruim	Muito_ruim	Outubro	Muito_ruim

Fonte: elaboração própria

Levando em consideração que o algoritmo Naive Bayes apresenta a capacidade de indução, qualquer exemplo que apresente as características dos 3 exemplos encontrados na classe “não_popular” terão esta classe citada. A seguir, serão abordados possíveis lógicas nos atributos de entrada que podem ser indicadores para saber o motivo desses documentos serem classificados como “não_popular”. A princípio, será feita uma avaliação individual por subcategorias e, após isso, serão avaliados de forma geral os documentos.

Com os exemplos e suas características apresentadas, é possível analisar o seguinte: para os atributos *title* e *abstract*, percebe-se que as classes foram “muito_ruim” e “ruim”, indicando que as palavras que foram utilizadas nesses documentos não continham altas pontuações (por isso que apresentam esses nomes). Com isso, pode existir a possibilidade de um documento ser “não_popular” caso o título e o resumo não apresentem pontuações altas de frequência, sendo que elas são baseadas nas próprias palavras utilizadas nos artigos. Algo interessante a ser notado é que os artigos foram classificados como artigos clássicos, podendo indicar que nem sempre os artigos clássicos são os mais lidos. Os autores em todos os 3 documentos foram classificados como “muito_ruim”, trazendo um indicativo de que os autores, em comparação com a base coletada, não são os mais populares. O mês de publicação encontrado foram em fevereiro, março e outubro, indicando que esses meses podem apresentar uma publicação frequente.

Para a classe “pouco_popular” foram classificados corretamente 2 exemplos. O Quadro 23 apresenta este exemplo que foi classificado pelo algoritmo Naive Bayes.

Quadro 23 - Exemplo de teste classificado como “pouco_popular” na subcategoria “cinco classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Muito_ruim	journal	proceedings_of_the_national_academy_of_sciences	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_ruim	março	Muito_ruim
2	Muito_ruim	Journal	Faseb_journal	Estado_da_arte	Muito_ruim	Muito_ruim	Dezembro	Muito_ruim

Fonte: elaboração própria

Com base nos atributos apresentados para este caso, é possível analisar o seguinte: tanto *title* quanto *abstract* e *keywords* apresentaram a classe “muito_ruim”, indicando que o documento pode ser pouco popular por apresentar baixas frequências nestes atributos. O ano de publicação obteve como classes “artigo_para_revisão_de_literatura” e “estado_da_arte”, apresentando publicações relativamente recentes. O mês de publicação é março e dezembro, indicando que são possíveis meses frequentes para publicação de artigos.

Para a classe “popular”, também foram classificados corretamente 2 exemplos. O Quadro 24 apresenta os exemplos que foram usados no algoritmo Naive Bayes.

Quadro 24 - Exemplo de teste classificado como “popular” na subcategoria “cinco classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Muito_ruim	journal	world_journal_of_diabetes	Artigo_classico	Muito_ruim	Muito_ruim	dezembro	Muito_ruim
2	Ruim	Journal	journal_of_antimicrobial_chemotherapy	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_ruim	março	Muito_ruim

Fonte: elaboração própria

Com base nos atributos apresentados, é possível analisar o seguinte: *keywords*, *authors* e *abstract* foram classificados como “muito_ruim”, podendo indicar que não são esses atributos que estão influenciando mais fortemente para que o documento tenha sido classificado pelo algoritmo Naive Bayes como “popular”. É possível que o ano de publicação, que foi classificado como “artigo_classico” e “artigo_para_revisão_de_literatura” tenha uma influência, visto que *title*, *keywords* e *abstract* apresentam frequências baixas conforme sua classificação. Porém, o fato de ser “artigo_clássico” e “artigo_para_revisão_de_literatura” pode ser o principal fator para a classificação ser “popular”, visto que documentos dessas naturezas geralmente tem uma quantidade de leitura considerável. Os meses de publicação foram dezembro e março, trazendo um indicativo de frequência de publicação nesses meses.

Para a classe “muito_popular” foram classificados corretamente 3 exemplos. O Quadro 25 apresenta esses exemplos que foram utilizados.

Quadro 25 - Exemplos de teste classificados como “muito_popular” na subcategoria “cinco classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	bom	journal	bmc_proceedings	Artigo_de_referência	Muito_bom	Muito_ruim	junho	Ruim
2	Ruim	Journal	journal_of_theoretical_biology	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_ruim	Novembro	Muito_ruim
3	Muito_ruim	Journal	Proceedings_biological_sciences_the_royal_society	Artigo_de_referência	Muito_ruim	Muito_ruim	Fevereiro	Muito_ruim

Fonte: elaboração própria

Com base nos atributos dos exemplos de teste apresentados, é possível fazer a seguinte análise: o *title* do primeiro exemplo foi classificado como “bom”, indicando que o documento apresenta uma pontuação boa em relação a base de dados, enquanto que o atributo *title* do segundo e terceiro exemplos apresentaram a classificação “ruim”, indicando uma pontuação de frequência menor. O atributo *keywords* apresentou a classe “muito_bom” no primeiro exemplo, fortalecendo ainda mais sua pontuação na base de dados. Acredita-se que, para este caso, o motivo do documento ser classificado como “muito_popular” veio, possivelmente, aos atributos *title* e *keywords*. Aparentemente, o autor não tem grande influência para a popularidade dos documentos, visto que ele foi classificado como “muito_ruim”. Os meses de publicação foram junho, novembro e fevereiro, indicando que estes meses podem ser frequentes para publicação de artigos. Para o segundo exemplo, acredita-se que o ano de publicação está trazendo a maior influência, levando em consideração que, no exemplo anterior, um documento foi classificado como “popular” e apresentou o ano de publicação como “artigo_para_revisão_de_literatura”, reforçando que documentos desse tipo podem ter uma quantidade considerável de leitores em alguns casos. O mesmo pode ser dito para o terceiro exemplo, com o ano classificado como “artigo_de_referência”.

Para a subcategoria “três classes de saída” da classe “não_popular”, foram classificados 4 exemplos. O Quadro 26 apresenta os exemplos de teste que foram usados no algoritmo Naive Bayes.

Quadro 26 - Exemplos de teste classificado como “não_popular” na subcategoria “três classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Muito_ruim	Conference_proceedings	proceedings_of_the_national_aviation_university	Artigo_de_referência	Muito_ruim	bom	março	Muito_ruim
2	Muito_ruim	Journal	Api_conference_proceedings	Artigo_de_referência	Muito_ruim	Muito_ruim	Novembro	Muito_ruim
3	Excelente	Journal	american_journal_of_environmental_sciences	Saindo_do_forno	Muito_ruim	Muito_ruim	Maio	Excelente
4	Muito_ruim	Journal	Aha_journal	Estado_da_arte	Muito_ruim	Muito_ruim	Setembro	Muito_ruim

Fonte: elaboração própria

Baseando-se nos atributos apresentados, é possível perceber o seguinte. Para os exemplos 1 e 4, o motivo dele ter sido classificado como “não_popular” pode ser devido aos atributos *title* e *keyword* e *abstract*, que foram classificados como “muito_ruim”. Os meses de março e setembro podem ser indicadores de frequência de publicação nesses casos. No caso do exemplo 2, o motivo da classificação ser “não_popular” também pode ter vindo dos atributos *title*, *keywords*, *authors* e *abstract*, por apresentarem a classe “muito_ruim”. *Authors* também obteve essa mesma classe. O mês de publicação foi novembro, podendo ser outro possível indicador de frequência de publicação (os meses serão analisados em breve, quando for apresentada a análise geral dos exemplos classificados). O exemplo 3 apresenta certa particularidade, pois *title* e *abstract* apresentam a classe “excelente”, indicando que as palavras presentes nesses atributos contém alta frequência na base de dados. Acredita-se que o atributo que mais influenciou para que o documento seja “não_popular” foi o *year*, que contem a classe “saindo_do_forno”. Esta classe apresenta apenas documentos publicados em 2016 e 2017, ou seja, são recentes. Possivelmente a maioria desses documentos contém uma quantidade de leitores mínima, principalmente se for comparar com outros documentos mais antigos. Com isso, pode existir possibilidades de documentos classificados como “saindo_do_forno” apresentarem a classe “não_popular”.

Para a classe “popular”, 1 exemplo foi classificado. O Quadro 27 apresenta este exemplo.

Quadro 27 - Exemplo de teste classificado como “popular” na subcategoria “três classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Excelente	journal	world_journal_of_gastroenterology	Artigo_de_referência	bom	Muito_ruim	maio	Excelente

Fonte: elaboração própria

Com base nos atributos deste documento, é possível fazer a seguinte analise: este documento pode ter sido classificado como “popular” devido aos atributos *title*, *keywords* e *abstract*, que apresentam as classes “excelente”, “bom” e “excelente” respectivamente. *Authors* apresenta a classe “muito_ruim” e isso já foi visto na subcategoria “cinco classes de saída” na mesma classe. Neste caso, pode ser uma confirmação de que a popularidade de um autor nem sempre tem influência na popularidade de um documento.

Para a classe “extremamente_popular”, foram classificados pelo algoritmo Naive Bayes 2 exemplos. O Quadro 28 apresenta as características destes exemplos.

Quadro 28 - Exemplo de teste classificado como “extremamente_popular” na subcategoria “três classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Excelente	journal	world_journal_of_gastroenterology	Artigo_de_referência	bom	Muito_ruim	maio	Excelente
2	Muito_ruim	Journal	journal_of_operations_managemen t	Artigo_classico	Muito_ruim	Muito_ruim	Novembro	Muito_ruim

Fonte: elaboração própria

Para este caso, é possível fazer a seguinte análise: o motivo do documento ter sido classificado como “extremamente_popular” no exemplo 1 pode ser devido ao *title* e *abstract* apresentam a classe “excelente”, indicando que as palavras contidas nesses atributos são de máxima frequência na base de dados. *Authors* como “muito_ruim” pode ser mais um indicativo de que este atributo não tem grande influência na popularidade de um documento no Mendeley. O mês de publicação é maio e na subcategoria “popular” ele também foi indicado. Isso significa que maio pode ser um mês frequente para publicação de artigos. O segundo exemplo apresenta um comportamento diferente, pois sua classificação foi “muito_ruim” na maioria dos casos. Porém, o fato dele ser um “artigo_classico” pode influenciar. Em exemplos anteriores, foi mostrado que documentos desse tipo podem chegar a classe “não_popular”. Porém, podem existir casos em que documentos com essa característica sejam classificados como “extremamente_popular”, principalmente se este artigo clássico for altamente recomendado para citações em trabalhos.

Para a subcategoria “duas classes de saída”, na classe “não_popular”, apresenta-se o Quadro 29 com os exemplos de teste classificados no algoritmo Naive Bayes.

Quadro 29 - Exemplos de teste classificados como “não_popular” na subcategoria “duas classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Excelente	journal	proceedings_of_the_indian_academic_of_sciences_chemical_sciences	Artigo_classico	Muito_ruim	Muito_ruim	maio	Muito_ruim
2	Excelente	journal	international_journal_of_molecular_sciences	Saindo_do_forno	Excelente	Muito_ruim	Julho	Excelente
3	Muito_ruim	Journal	netherlands_journal_of_medicine	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_ruim	Outubro	Muito_ruim

Fonte: elaboração própria

Observando os atributos dos exemplos do Quadro 29, é possível analisar o seguinte: Para o primeiro exemplo, o motivo de sua classificação ser “não_popular” pelo algoritmo Naive Bayes pode ser devido aos atributos *keywords* e *abstract*, que apresentam a classe “muito_ruim”. No segundo exemplo, existe o mesmo caso particular do exemplo 3 do Quadro 26. O documento apresentado neste exemplo pode ser recente, pois a coleta de dados foi finalizada dia 26 de julho de 2017, conforme visto no Capítulo 4. Nesse sentido, este documento pode ter sido publicado em um período de 1 ano ou 1 mês, levando em consideração a data de finalização da coleta de dados fazendo com que, de fato, o documento não tenha sido muito lido se for comparar com os demais documentos da base de dados. O terceiro exemplo apresentou um comportamento semelhante de outros exemplos que já foram analisados aqui. Neste caso, *title*, *keywords*, *authors* e *abstract* apresentam a classe “muito_ruim”, indicando pontuações mínimas na base de dados.

Para a classe “extremamente_popular”, é apresentada o Quadro 30 com os exemplos que foram classificados pelo algoritmo Naive Bayes.

Quadro 30 - Exemplos de teste classificados como “extremamente_popular” na subcategoria “duas classes de saída”.

Exemplos	Title	Type	Source	Year	Keywords	Authors	Month	Abstract
1	Muito_ruim	journal	Bmc_proceedings	Artigo_de_referência	Muito_ruim	Muito_ruim	novembro	Muito_ruim
2	Muito_ruim	journal	proceedings_of_the_national_academy_of_sciences_of_the_united_states_of_america	Artigo_de_referência	Muito_ruim	Muito_ruim	Dezembro	Muito_ruim
3	Excelente	journal	international_journal_of_environmental_research_and_public_health	Artigo_de_referência	Muito_ruim	Muito_ruim	Janeiro	Excelente
4	Muito_ruim	Journal	Embo_journal	Artigo_para_revisão_de_literatura	Muito_ruim	Muito_ruim	Março	Muito_ruim

Fonte: elaboração própria

Observando o Quadro 30, é possível analisar o seguinte: o motivo para o primeiro exemplo ter sido classificado como “excelente” pode ser devido ao atributo *year*, que é “artigo_de_referência”. Os demais atributos não aparentam trazer indicadores para um documento “extremamente_popular”, podendo ter certa exceção a *authors*, levando em consideração que alguns documentos foram classificados como “popular” ou “extremamente_popular” com o autor na classe “muito_ruim”. Isso pode demonstrar que a popularidade de um autor não influencia na popularidade do documento. O segundo exemplo é semelhante ao primeiro. Supõe-se também que o *year* como “artigo_de_referência” pode estar influenciando para que o documento seja “extremamente_popular”. O terceiro exemplo é diferente, apresentando *title* e *abstract* como “excelente” e isso pode ser um indício para que o documento tenha sido classificado como “extremamente_popular”. O *year* também pode estar influenciando neste caso. O quarto exemplo apresenta comportamento diferente dos demais. Porém, o fato dele ser “artigo_para_revisão_de_literatura” pode influenciar de alguma forma, pois esse artigo pode ser altamente referenciado em trabalhos, sendo possível aumentar a contagem de leitores neste caso.

A partir de agora, será feita uma análise geral dos exemplos de teste, a fim de identificar algum padrão com base nos atributos de entrada. Começando pela classe “não_popular”, percebe-se que *title*, *abstract*, *keywords* e *authors* mantém, na maioria das vezes, a classe “muito_ruim”. O *title* apresentou um comportamento diferenciado, visto que 3 exemplos foram classificados com o *title* como “excelente” na classe “não_popular” contra 3 exemplos que contém a característica “muito_ruim” na mesma classe. A diferença é que os 3 exemplos que contém a classe “excelente” em *title* apresenta, no mínimo, *keywords*, *authors*

ou *abstract* como “muito_ruim”. Pode-se dizer, neste caso, que esses atributos, de maneira geral, contém pontuações mínimas na base de dados, por isso a classificação como “não_popular”.

Para a classe “pouco_popular”, percebe-se um comportamento semelhante com o “não_popular”, tendo *title*, *keywords*, *authors* e *abstract* com a classe “muito_ruim”, trazendo indicação de que esses documentos têm pontuações mínimas de frequência desses atributos em relação a base de dados.

Quanto a classe “popular”, há grandes variações nos atributos, prevalecendo as classes “muito_ruim”, “bom” e “excelente”. Nos exemplos apresentados, *title* e *abstract* apresentaram casos com as classes “muito_ruim” e “excelente”. *Keywords* variou entre “muito_ruim” e “bom”. Isso pode ser um indicativo de que a classe “popular” sempre apresentará variações nos atributos de entrada: enquanto um está “muito_ruim”, outro atributo pode estar com a classe “bom” ou “excelente”. No caso do *authors*, houve a prevalência da classe “muito_ruim”, sendo um indicativo de que um documento popular não necessita de um autor popular.

Para a classe “muito_popular” houve variação somente em alguns atributos: *title* apresenta as classes “bom” e “ruim” e *abstract* apresenta a classe “ruim” e “muito_ruim”. Os demais atributos apresentam classes semelhantes. No caso da classe “muito_popular”, não há uma grande variação nos atributos, podendo ser possível que o ano de publicação seja o principal influenciador para que os documentos classificados tenham a classe “muito_popular”, já que ambos contem as classes “artigo_para_revisão_de_literatura” e “artigo_de_referência”.

Na classe “extremamente_popular”, da mesma forma que em “popular”, houve variações nos atributos de entrada. Porém, há uma certa prevalência da classe “muito_ruim” principalmente em *keywords* e *authors*. Isso pode confirmar que palavras-chave com mínimas pontuações na base de dados ou a não popularidade de um autor não é indicativo para que um documento seja classificado como “extremamente_popular”.

No caso dos meses dos exemplos classificados, observa-se que na classe “não_popular”, é indicado os meses de março, maio e outubro. Na classe “pouco_popular”, apresenta-se os meses de março e dezembro. Para a classe “popular”, indica-se dezembro. Na classe “muito_popular” é indicado os meses de junho e novembro e a classe “extremamente_popular” apresenta o mês de dezembro. No ano de publicação, para a classe “não_popular” há uma prevalência maior nas classes “artigo_classico”, “artigo_de_referência” e “saindo_do_forno”. Na classe “pouco_popular”, apresentam as

classes “artigo_para_revisão_de_literatura” e “estado_da_arte”. Para a classe “popular”, apresentam as classes “artigo_para_revisão_de_literatura” e “artigo_classico”. Na classe “muito_popular” apresentam as classes “artigo_de_referência” e “artigo_para_revisão_de_literatura”. Para a classe “extremamente_popular”, há a prevalência da classe “artigo_de_referência”. Apesar da característica “artigo_de_referência” aparecer em todas as classes de saída, sua maior prevalência aparece em “extremamente_popular”. Percebe-se que alguns anos classificados como “artigo_classico” podem estar tanto na classe “não_popular” quanto “popular”, ou seja, este atributo apresenta algumas variações que dependem dos demais atributos para estabelecer sua classe de saída. Para o caso do “não_popular”, há uma influência maior nas pontuações de *title*, *keywords* e *abstract* do que os demais atributos. No caso da classe “popular”, a variação é maior devido a esta classe ser considerada uma intermediária entre “não_popular” e “extremamente_popular”. Portanto, todos os seus atributos apresentam variações cujas características são todas as que já foram discutidas nesta análise geral.

Com isso, é possível perceber alguns padrões de comportamento dos cientistas no Mendeley. Uma delas é a questão dos *authors*. Pelo fato deste atributo ser classificado como “muito_ruim” inclusive para a classe “extremamente_popular”, pode ser um sinal de que os cientistas no Mendeley não estão preocupados com o currículo do autor, e sim com os trabalhos publicados. Os cientistas do Mendeley aparentam se interessar por documentos publicados entre os anos 2000 a 2007, por isso que a classe “artigo_de_referência” apresentou certa frequência em “extremamente_popular”. É importante informar que a frequência aqui apresentada está relacionada principalmente com o acesso. Neste caso, se os documentos mais frequentes foram entre os anos de 2000 a 2007, significa dizer que a frequência de acesso nesses documentos é maior que nos demais.

O próximo Capítulo apresenta as considerações finais desta dissertação e as propostas para continuação deste trabalho.

6 CONSIDERAÇÕES FINAIS

Esta dissertação apresentou um trabalho exploratório que envolveu classificação de documentos usando a base de dados do Mendeley e o algoritmo Naive Bayes. Com base nos resultados quantitativos e qualitativos apresentados, é possível afirmar que a subcategoria "duas classes de saída" apresentou melhores resultados, levando em consideração que o PECC foi maior em comparação com as demais subcategorias e os exemplos de treino apresentaram melhor distribuição, deixando a subcategoria mais equilibrada. Dos exemplos classificados no algoritmo Naive Bayes, percebe-se que os meses de maio, março e dezembro são frequentes na base de dados, indicando que são possíveis meses para publicação.

As 4 categorias apresentadas nesta dissertação (*Proceedings_open, Proceedings, Journal_Open e Journal*) tem como características o acesso dos documentos. Neste caso, duas dessas categorias apresentam documentos com acesso aberto (Proceedings_Open e *Journal_Open*) e duas sem acesso aberto (*Proceedings* e *Journal*). Nos resultados apresentados, não houve diferenças entre as categorias, mas a experiência foi válida para a exploração do trabalho.

A pesquisa que foi feita nesta dissertação demanda por mais trabalhos, uma vez que pode existir a possibilidade dela responder mais questões que venham surgir no futuro. Embora a pesquisa tenha tentado identificar o comportamento dos cientistas no Mendeley, acredita-se que é necessário uma pesquisa maior, a fim de identificar mais padrões que se relacionem com o comportamento. Ao menos, o reconhecimento de padrões no Mendeley é algo possível, visto que houve a classificação dos dados coletados.

O trabalho apresentou diversos limites que foram superados no decorrer do desenvolvimento. O primeiro deles foi entender como funciona a arquitetura do Mendeley e como desenvolver um algoritmo capaz de coletar os dados. Durante o desenvolvimento, o Mendeley modificou sua autenticação oauth para a versão 2.0, fazendo com que os algoritmos que foram desenvolvidos durante a autenticação 1.0 ficassem inviáveis de serem executados na plataforma. Além disso, a Elsevier comprou o Mendeley e isso causou uma preocupação, visto que poderia ter possibilidades da revista "trancar" os dados dos documentos do Mendeley, impossibilitando a coleta dos dados.

Após longos períodos de desenvolvimento, finalmente o primeiro algoritmo funcional para coleta de dados foi concluído. A partir desse ponto, os dados começaram a ser coletados por volta de fevereiro de 2017 até julho de 2017. Durante a fase da coleta, começou-se a pensar e desenvolver formas de adaptar os dados coletados para que o algoritmo Naive Bayes

tivesse capacidade de reconhecer e classifica-los. O resultado final disso tudo foi visto no Capítulo 4 desta dissertação.

Durante o desenvolvimento do método de tratamento e adaptação dos dados, diversos obstáculos foram superados. O primeiro deles foi que a primeira versão dos algoritmos apresentaram problemas de desempenho no computador que estava realizando o processo. Grande parte das etapas causavam *Overflow* (quando a memória RAM está cheia e novos elementos tentam ser inseridos nela) no sistema. Com isso, diversas versões foram feitas a fim de otimizar o processo para que este tipo de problema seja evitado. Uma solução que apresentou um desempenho ótimo foi a distribuição dos processos em lotes. Neste caso, o algoritmo processa parte dos documentos até concluir e, logo em seguida, executa outro lote de documentos até terminar o processo.

Como foi visto no Capítulo 4, a etapa de pré-adaptação foi consequência de outro limite no trabalho que foi alcançado. A primeira versão do algoritmo apresentava somente uma etapa de adaptação, fazendo com que o processo levasse um tempo considerável, visto que todos os tratamentos de pré-adaptação vistos no trabalho executavam novamente no mínimo três vezes. Com isso, para otimizar o tempo de execução, foi desenvolvido um novo algoritmo separando todos os processos repetidos na adaptação final, a fim de executá-los somente uma vez. Isso causou uma grande redução de tempo de execução dos documentos.

Apesar de todas as técnicas de otimização e tratamento apresentadas, acredita-se que o algoritmo ainda pode ser melhorado. Um problema encontrado é que os algoritmos são independentes, ou seja, todos são executáveis. Ao terminar um processo, é necessário executar manualmente o processo seguinte. Neste caso, antes de realizar novas pesquisas, esse problema deverá ser resolvido para que a execução do método seja mais facilitada.

Dependendo da quantidade de dados a serem trabalhados no método, ainda existe uma possibilidade de demora no tratamento principalmente se for usado apenas uma máquina para fazer todo o processo. Neste caso, recomenda-se várias máquinas para resolver os métodos em paralelo, a fim de otimizar o tempo e também de testar e tratar os dados no algoritmo Naive Bayes com mais rapidez.

Como forma de continuação do trabalho futuramente, pensou-se em várias questões levando em consideração as limitações deste trabalho. Neste caso, é possível citar:

1 - A inclusão de outros classificadores de aprendizado de máquina com o paradigma de aprendizagem supervisionada ou não supervisionada.

2 - A inclusão de outros métodos de amostragem para verificar se a alteração no método pode influenciar no resultado final.

3 - Fazer testes mais exaustivos na base de dados, uma vez que os resultados deste trabalho apresentaram poucos testes, já que os dados levam tempo para serem processados durante o tratamento. No caso, providenciar um ambiente onde seja possível o teste de vários resultados simultaneamente.

4 - Verificar outras formas de avaliar a quantidade de leitores no Mendeley, levando em consideração que, durante a pesquisa, foi possível encontrar o atributo *reader_count* distribuído de diversas formas. A exemplo, é possível citar a quantidade de leitores por país, por disciplina, por áreas de interesse, entre outros.

É possível que os resultados de um trabalho futuro envolvendo este tema seja mais rápido, visto que todos os algoritmos desenvolvidos para tratar esses dados estão prontos, cabendo apenas a execução deles para gerar o conjunto de dados a ser classificado por um algoritmo de AM. Os algoritmos usados para a coleta e tratamento dos dados estão no apêndice deste trabalho. Como foi visto anteriormente, existe possibilidade de melhora nesses algoritmos. Porém, do jeito que estão, é possível realizar todos os procedimentos metodológicos deste trabalho.

REFERÊNCIAS

ALHEYASAT, O. Investigation and analysis of research gate user's activities using neural networks. **The International Arab Journal of Information Technology** v. 13, n. 2, p. 320-325, mar. 2016.

ANDRADE, D.; VERGUEIRO, W. **Aquisição de materiais de informação**. Brasília, DF: Briquet de Lemos, 1996.

BIOLCHINI, J, *et. al*. Scientific research ontology to support systematic review in software engineering. **Advanced Engineering Informatics**. v. 21, n. 2, p. 133–151, abr. 2007.

BISHOP, C. M. **Pattern recognition and maching learning**. Cambridge: Springer Science+Business Media, 2006.

BJÖRNEBORN, **L. Small-world link structures across an academic web space: a library and information science approach**, 2004. Tese (doutorado). Royal School of Library and Information Science.

BUTLER, J, *et. al*. A. The evolution of current research impact metrics: from bibliometrics to altmetrics? **Clinical Spine Surgery**. v. 30, n. 5, p. 226-228, Jun. 2017.

CONDUTA, Bruno C.; MAGRIN, Diego H. **Aprendizagem de máquina**. Universidade Federal de Limeira, Campinas: 2010.

CORREIA, A.; MESQUITA, A. **Mestrados e doutoramentos: estratégias para a elaboração de trabalhos científicos: o desafio da excelência**. 2. ed. Porto: Vida Econômica, 2014.

DAMAYANTI, M.; SUKMAAGI, A.; SUHANDIAH, S. Analysis strategy visibility and activity at the website stikom.edu in terms of increasing ranked webometrics. **Jurnal Sistem informasi & Komputer Akuntansi**, v. 5, n.10, 2016.

FACELI, K, *et. al*. **Inteligência artificial: uma abordagem de aprendizado de máquina**. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2008.

FENNER, Martin. **Altmetrics and other novel measures or scientific impact**: opening science: the evolving guide on how the internet is changing research, collaboration and scholarly publishing. New York: Springer Cham Heidelberg, 2014.

GOUVEIA, F. C. Altmetria: métricas de produção científica para além das citações. **Liinc em Revista**, v. 9, n. 1, p. 214-227, 2013.

_____; LANG, P. Da **Webometria a Altmetria: uma jornada por uma ciência emergente**. In: Fronteiras da ciência da informação. Brasília, DF: IBICT, 2013, p. 172-195.

HUANG, Z; YUAN, B. Mining google scholar citations: an exploratory study. In: **International Conference Ingeligent Computing Technology** – n. 8, China, 2012, p. 182–

189.

KITCHENHAM, B. A., DYBA˚, T., JORGENSON, M. Evidence-based software engineering. In: **Proceedings of ICSE**, p. 273–281. May, 2004.

LEUNG, Xi Y.; SUN, JIE.; BAI, BILLY. Bibliometrics of social media research: a co-citation and co-word analysis. **International Journal of Hospitality Management**, v. 66, p. 35-45. sept. 2017.

LIBRARY, INFORMATION SCIENCE AND TECHNOLOGY - LISA. **Subjects Include**. Disponível em: <https://www.ebsco.com/products/research-databases/library-information-science-and-technology-abstracts>. Acesso em: 06 set. 2017.

LIMA, B; MACHADO, V. Machine learning algorithms applied in automatic classification of social network users. In: CONGRESSO TECNOLÓGICO TI E TELECON, 4., **anais...** 2011.

_____; LOPES, L; MACHADO, V. **Automatic labeling of social network users Scientia.net through the machine learning supervised application**: social network analysis and mining. July, 2015.

LUNDEN, I. Confirmed: elsevier has bought Mendeley for $69M : $100M to expand its open, social location data efforts. Disponível em: <https://techcrunch.com/2013/04/08/confirmed-elsevier-has-bought-mendeley-for-69m-100m-to-expand-open-social-education-data-efforts/ >. Acesso em: 29 jan. 2018.

MITCHELL, T. M. **Machine learning**. USA: McGraw-Hill, 1997.

MOHAMMADI, E.; THELWALL, M. Mendeley readership altmetrics for the social Sciences and humanities: research evaluation and knowledge flows. **Journal of The Association Science and Technology**, v. 65, n. 8, p. 1627-1638, Mar. 2014.

_____; THELWALL, M. KOUSHA, K. Can Mendeley bookmarks reflect readership?: a survey of user motivations. **Journal of The Association for Information Science and Technology**, v. 67, n. 5, p. 1198-1209, Mar. 2015.

_____; et. al. Who reads research articles? an altmetrics analysis of Mendeley user categories. **Journal of Association for Information Science and Technology**, v. 66, n. 9, p. 1832-1846, apr. 2015.

NOGARE, D. **Entendendo como funciona o algoritmo K-Means**, 2015. Disponível em: <http://www.diegonogare.net/2015/08/entendendo-como-funciona-o-algoritmo-de-cluster-k-means/>. Acesso em: 10 set. 2017.

NOORDEN, R. V. **Online collaboration: scientists and the social network**. Disponível em: <http://www.nature.com/news/online-collaboration-scientists-and-the-social-network-1.15711>. Acesso em: 12 fev. 2017.

ORTEGA, J. Differences and evolution of scholarly impact in Google Sholar Citations profiles: an application of decision trees. **Revista Española de Documentación Científica**, v. 38, n. 4, 2015.

PATTILLO, G. **Fast facts**: Mendeley: Mendeley research networks. 1, 2009.

CAPES. **Acervo**. Disponível em: <https://www.periodicos.capes.gov.br/?option=com_pcollection&mn=70&smn=79>. Acesso em: 06 set. 2017.

PRIEM, J.; et. al. *Altmetrics:* a manifesto. 2010. Disponível em: <http://altmetrics.org/manifesto>. Acesso em: 22 out. 2017.

REDAÇÃO GLOBAL AD. **O que é Crawler?**. Disponível em: <http://www.globalad.com.br/blog/o-que-e-crawler>. Acesso em: 10 set. 2017.

ROCHA, M.; CORTEZ, P.; NEVES, J. **Análise inteligente de dados: algoritmos e implementação em java**. Lisboa: FCA – Editora de Informática, 2008.

RUSSO, G.; et. al. Mendeley: an easy way to manage, share, and synchronize papers and citations. **Plastic and Reconstructive Surgery**, v. 131, n. 6, 2013.

SCHNEIDER, M. et. al. Feasibility of common bibliometrics in evaluating translational science. **Journal of Clinical and Translational Science**, v. 1, n. 1, p. 45-52, feb. 2017.

SUGIMOTO, C.; et. al. Scholarly use of social media and altmetrics: a review of the literature. **Journal of The Association For Information Science and Technology**, v. 68, n. 9, p. 2037-2062, sept. 2017.

THELWALL, M.; SUD, P. Mendeley readership counts: an investigation of temporal and disciplinary differences. **Journal of The Association Sciente and Technology**, v. 67, n. 12, p. 3036-3050, dic. 2016.

_____; WILSON, P. Mendeley readership altmetrics for medical articles: an analysis of 45 fields. **Journal of The Association for Information Science and Technology**, v. 67, n. 8, p. 1962-1972, aug. 2016.

UNIVERSITY OF WAIKATO. **Weka 3:** Data Mining Software in Java. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 12 set. 2017.

VALIENTE, C; MEDONZA, J. ARENCIBIA-JORGE, R. A review of altmetrics as an emerging discipline for research evaluation. **The Association of Learned & Professional Society Publishers**, v. 29, n. 4, p. 229-238, oct. 2016.

VERMA, M. BRAHMA, K. A webometric analysis of national libraries' websites in South Asia. **Annals of Library and Information Studies (ALIS)**, v. 64, n. 2, p. 116-124, june.2017.

APÊNDICE

Apêndice A – Algoritmo para coleta de dados no Mendeley (requer uso do app do Mendeley para autenticação)

```
public class ColetarDadosMendeley {
   public static String RESOURCE_URL;
   public static void main(String[] args) throws MalformedURLException, IOException {
      DateTimeFormatter fmt = DateTimeFormatter.ofPattern("HH:mm:ss");
      int i, numFileDocuments = 0, numResourceFile = 0, mendeleyCount = 1, z = 0, control = 0, contador = 0;
      String arq = "";
      Scanner scan = new Scanner(System.in);
      Config cf = new Config();
      System.out.println("Servlet Mendeley");
      System.out.println("Escolha uma das opções: ");
      System.out.println("1 - Primeira Consulta");
      System.out.println("? - Recuperação de consulta");
      int op = scan.nextInt();
      if (op == 1) {
         RESOURCE_URL = Config.START_RESOURCE_URL;
      } else if (op != 1) {
         File file = new File("../MendeleyDocs/src/mendeley/fileresource/" + "LastURL_" + numResourceFile +
".txt");
         BufferedReader bf = new BufferedReader(new FileReader(file));
         arq = bf.readLine();
         while (arq != null) {
            if (control == 0) {
               RESOURCE_URL = arq;
            } else if (control == 1) {
               numFileDocuments = Integer.parseInt(arq);
            } else if (control == 2) {
               numResourceFile = Integer.parseInt(arq);
            } else if (control == 3) {
               mendeleyCount = Integer.parseInt(arq);
            } else if (control == 4) {
               z = Integer.parseInt(arq);
            }
            arq = bf.readLine();
            control++;
         }
         arq = "";
         String resourceUrl = Config.START_RESOURCE_URL.replace(":account-id", Config.ACCOUNT_ID);
         URLConnection resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
         resource_cxn.addRequestProperty("Authorization", "Bearer " + Config.ACCESS_TOKEN);
         Map<String, List<String>> test = resource_cxn.getHeaderFields();
         for (Map.Entry<String, List<String>> entry : test.entrySet()) {
            System.out.println(entry.getKey() + " : " + entry.getValue());
            if (entry.getKey() != null) {
               if (entry.getKey().equals("Mendeley-Count")) {
                  List<String> mendeleyC = entry.getValue();
                  mendeleyCount = Integer.parseInt(mendeleyC.get(0));
               }
            }
         }
         System.out.println("Gravando dados, aguarde...");
      }
```

```
        while (z <= mendeleyCount) {
            i = 0;
            while (i != Config.COUNT_DOCS) {
                String resourceUrl = RESOURCE_URL.replace(":account-id", Config.ACCOUNT_ID);
                URLConnection resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                resource_cxn.addRequestProperty("Authorization", "Bearer " + Config.ACCESS_TOKEN);
                Map<String, List<String>> test = resource_cxn.getHeaderFields();
                //aqui nesse if, fazer um while para sair apenas se ContentLength for diferente de zero
                //e persistir no envio para o mendeley até isso acontecer
                //Se ocorrer do token expirar aqui, vai acontecer o erro 401
                if (resource_cxn.getContentLength() == 0) {
                    // System.out.println("Erro 504. Tentando solucionar o problema...")
                    while (resource_cxn.getContentLength() == 0) {
                        resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                        resource_cxn.addRequestProperty("Authorization", "Bearer " + Config.ACCESS_TOKEN);
                        test = resource_cxn.getHeaderFields();
                        //código novo aqui
                        for (Map.Entry<String, List<String>> entry3 : test.entrySet()) {
                            List<String> codeHttp = entry3.getValue();
                            String code = codeHttp.get(0);
                            String[] tokens = code.split(" ");
                            for (int m = 0; m < tokens.length; m++) {
                                if (tokens[m].equals("401")) {
                                    System.out.println("Token expirou, insira um novo
token.....");
                                    Config.ACCESS_TOKEN = scan.next();
                                    LocalDateTime timePoint = LocalDateTime.now();
                                    System.out.println("Token inserido as " + timePoint.format(fmt));
                                    System.out.println("Gravando dados, aguarde...");
                                    resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                                    resource_cxn.addRequestProperty("Authorization", "Bearer " +
Config.ACCESS_TOKEN);
                                    test = resource_cxn.getHeaderFields();
                                    int controlador = 0;
                                    for (Map.Entry<String, List<String>> entry4 : test.entrySet()) {
                                        if (controlador == 0) {
                                            List<String> codeHttp1 = entry4.getValue();
                                            String code1 = codeHttp1.get(0);
                                            String[] tokens2 = code1.split(" ");
                                            String codigo = tokens2[1];
                                            if (codigo.equals("200")) {
                                                break;
                                            } else {
                                                while (codigo.equals("500") || codigo.equals("504") || codigo.equals("503")) {
                                                    resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                                                    resource_cxn.addRequestProperty("Authorization", "Bearer " +
Config.ACCESS_TOKEN);
                                                    test = resource_cxn.getHeaderFields();
                                                    int controlador1 = 0;
                                                    for (Map.Entry<String, List<String>> entry5 : test.entrySet()) {
                                                        if (controlador1 == 0) {
                                                            List<String> codeHttp2 = entry5.getValue();
                                                            String code2 = codeHttp2.get(0);
                                                            String[] tokens3 = code2.split(" ");
                                                            codigo = tokens3[1];
                                                            System.out.println("Codigo do erro: " + codigo);
                                                            if (codigo.equals("200")) {
                                                                break;
                                                            }
                                                        }
```

```
                                controlador1 = 1;
                            }
                        }
                    }
                    controlador = 1;
                }
            }


        } else if (tokens[m].equals("500") || tokens[m].equals("503")) {
            //Erro 500
            String codigo = tokens[m];
            while (codigo.equals(tokens[m]) || codigo.equals("504") || codigo.equals("503")) {
                resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                resource_cxn.addRequestProperty("Authorization", "Bearer " +
Config.ACCESS_TOKEN);
                test = resource_cxn.getHeaderFields();
                int controlador = 0;
                for (Map.Entry<String, List<String>> entry4 : test.entrySet()) {
                    if (controlador == 0) {
                        List<String> codeHttp1 = entry4.getValue();
                        String code1 = codeHttp1.get(0);
                        String[] tokens1 = code1.split(" ");
                        codigo = tokens1[1];
                        if (codigo.equals("200")) {
                            break;
                        }
                    }
                    controlador = 1;
                }
            }
        }
    }
    break;
}
}
} else {
    Map<String, List<String>> test3 = resource_cxn.getHeaderFields();
    for (Map.Entry<String, List<String>> entry3 : test3.entrySet()) {
        if (contador == 0) {
            List<String> codeHttp = entry3.getValue();
            String code = codeHttp.get(0);
            String[] tokens = code.split(" ");
            for (int m = 0; m < tokens.length; m++) {
                if (tokens[m].equals("401")) {
                    System.out.println("Token expirou, insira um novo
token.....");
                    Config.ACCESS_TOKEN = scan.next();
                    LocalDateTime timePoint = LocalDateTime.now();
                    System.out.println("Token inserido as " + timePoint.format(fmt));
                    resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                    resource_cxn.addRequestProperty("Authorization", "Bearer " +
Config.ACCESS_TOKEN);
                    test = resource_cxn.getHeaderFields();
                    int controlador = 0;
                    for (Map.Entry<String, List<String>> entry4 : test.entrySet()) {
                        if (controlador == 0) {
                            List<String> codeHttp1 = entry4.getValue();
                            String code1 = codeHttp1.get(0);
                            String[] tokens2 = code1.split(" ");
```

```
                        String codigo = tokens2[1];
                        if (codigo.equals("200")) {
                            break;
                        } else {
                            while (codigo.equals("500") || codigo.equals("504") || codigo.equals("503")) {
                                resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                                resource_cxn.addRequestProperty("Authorization", "Bearer " +
Config.ACCESS_TOKEN);
                                test = resource_cxn.getHeaderFields();
                                int controlador1 = 0;
                                for (Map.Entry<String, List<String>> entry5 : test.entrySet()) {
                                    if (controlador1 == 0) {
                                        List<String> codeHttp2 = entry5.getValue();
                                        String code2 = codeHttp2.get(0);
                                        String[] tokens3 = code2.split(" ");
                                        codigo = tokens3[1];
                                        System.out.println("Codigo http: " + codigo);
                                        if (codigo.equals("200")) {
                                            System.out.println("Gravando dados, aguarde...");
                                            break;
                                        }
                                    }

                                    controlador1 = 1;
                                }
                            }
                        }
                        controlador = 1;
                    }
                }

            } else if (tokens[m].equals("500") || tokens[m].equals("503")) {
                //Erro 500
                String codigo = tokens[m];
                while (codigo.equals(tokens[m]) || codigo.equals("504") || codigo.equals("503")) {
                    resource_cxn = (URLConnection) (new URL(resourceUrl).openConnection());
                    resource_cxn.addRequestProperty("Authorization", "Bearer " +
Config.ACCESS_TOKEN);
                    test = resource_cxn.getHeaderFields();
                    int controlador = 0;
                    for (Map.Entry<String, List<String>> entry4 : test.entrySet()) {
                        if (controlador == 0) {
                            List<String> codeHttp1 = entry4.getValue();
                            String code1 = codeHttp1.get(0);
                            String[] tokens1 = code1.split(" ");
                            codigo = tokens1[1];
                            System.out.println("Codigo http: " + codigo);
                            if (codigo.equals("200")) {
                                System.out.println("Gravando dados, aguarde...");
                                System.out.println("Concluido");
                                break;
                            }
                        }
                        controlador = 1;
                    }
                }
            }

        }
        contador = 1;
```

```
                }
            }
        }

        contador = 0;
        for (Map.Entry<String, List<String>> entry : test.entrySet()) {

            if (z == 0) {
                System.out.println(entry.getKey() + " : " + entry.getValue());
            }
            if (entry.getKey() != null) {
                if (entry.getKey().equals("Link")) {
                    List<String> newURLPagination = entry.getValue();
                    for (int k = 0; k < newURLPagination.size(); k++) {
                        if (k == (newURLPagination.size() - 1)) {
                            String[] tokens = newURLPagination.get(k).split(";");
                            tokens[0] = tokens[0].replace("<", "");
                            tokens[0] = tokens[0].replace(">", "");
                            RESOURCE_URL = tokens[0];
                        }
                    }
                } else if (entry.getKey().equals("Mendeley-Count")) {
                    List<String> mendeleyC = entry.getValue();
                    mendeleyCount = Integer.parseInt(mendeleyC.get(0));
                }
            }
        }
        if (z == 0) {
            System.out.println("");
            System.out.println("Gravando dados, aguarde...");
        }
        InputStream resource = resource_cxn.getInputStream();
        BufferedReader r = new BufferedReader(new InputStreamReader(resource, "UTF-8"));
        String line = null;
        while ((line = r.readLine()) != null) {
            arq = arq + line;
        }
        i = i + Config.LIMIT;
        z = z + Config.LIMIT;
        contador = 0;
        if (z >= mendeleyCount) {
            i = Config.COUNT_DOCS;
        }

        if (i == Config.COUNT_DOCS) {
            File fileDocuments = new File("../MendeleyDocs/src/mendeley/files/" + Config.FILE_NAME +
numFileDocuments + ".txt");
            File fileResource = new File("../MendeleyDocs/src/mendeley/fileresource/" + "LastURL_" +
numResourceFile + ".txt");
            BufferedWriter bfDocuments = new BufferedWriter(new FileWriter(fileDocuments));
            FileWriter fr = new FileWriter(fileResource);
            BufferedWriter bfResource = new BufferedWriter(fr);
            fr.write("");
            fr.flush();
            bfDocuments.append(arq);
            numFileDocuments++;
            bfResource.append(RESOURCE_URL + "\r\n");
            bfResource.append(Integer.toString(numFileDocuments) + "\r\n");
            bfResource.append(Integer.toString(numResourceFile) + "\r\n");
            bfResource.append(Integer.toString(mendeleyCount) + "\r\n");
```

```
                bfResource.append(Integer.toString(z));
                bfDocuments.close();
                bfResource.close();
                arq = "";
                LocalDateTime timePoint = LocalDateTime.now();
                System.out.println("Gravou: " + z + " as " + timePoint.format(fmt));
            }
        }

    }
  }

}
```

Apêndice B – Algoritmo para formatação dos dados coletados no Mendeley

```
public class FormatarDados {
public static void main(String[] args) throws IOException {

      String arq;
      String texto;
      int j = 0, numberCount = Config.START_NUMBER;
      System.out.println("Voce esta em : Formatação de dados");
      System.out.println("Formatando todos os dados, aguarde...");
      while (j != Config.QTD_DOCUMENTS) {
         if (numberCount == Config.START_NUMBER) {
            File file = new File(Config.CAMINHO_ARQUIVO + Config.FILE_NAME +
Config.START_NUMBER + ".txt");
            BufferedReader bf = new BufferedReader(new FileReader(file));

            File file1 = new File(Config.CAMINHO_ARQUIVO2 + Config.NEW_FILE_NAME +
Config.START_NUMBER + ".txt");
            BufferedWriter bfw = new BufferedWriter(new FileWriter(file1));
            arq = bf.readLine();
            String tokens[] = arq.split("\"title\"");
            if (file1.isFile()) {
               for (int i = 0; i < tokens.length; i++) {

                  if (i != 0) {
                     texto = "{\"title\"" + tokens[i];
                     bfw.append(texto + "\r\n");
                  }
               }
               bf.close();
               bfw.close();
            }
         }
         else {
            File file = new File(Config.CAMINHO_ARQUIVO + Config.FILE_NAME + numberCount + ".txt");
            BufferedReader bf = new BufferedReader(new FileReader(file));

            File file1 = new File(Config.CAMINHO_ARQUIVO2 + Config.NEW_FILE_NAME + numberCount
+ ".txt");
            BufferedWriter bfw = new BufferedWriter(new FileWriter(file1));

            arq = bf.readLine();
            String tokens[] = arq.split("\"title\"");
            if (file1.isFile()) {

               for (int i = 0; i < tokens.length; i++) {

                  if (i != 0) {
                     texto = "{\"title\"" + tokens[i];
                     bfw.append(texto + "\r\n");
                  }
               }
               bf.close();
               bfw.close();
            }
         }
         j++;
         numberCount++;
         if (j == Config.QTD_DOCUMENTS / 4) {
```

```
                System.out.println("25% concluido");
            }
            else if (j == Config.QTD_DOCUMENTS / 2) {
                System.out.println("50% concluido");
            }
            else if (j == (Config.QTD_DOCUMENTS / 2 + Config.QTD_DOCUMENTS / 4)) {
                System.out.println("75% concluido");
            }
        }
        System.out.println("Concluido");
    }
}
```

Apêndice C – Algoritmo para conversão dos documentos em JSON para Scripts de banco de dados (requer banco de dados mysql e acesso manual para cadastrar os dados automaticamente)

```
public class ConversorJsonScript {

   public static void main(String[] args) throws IOException {
      String id = "", title = "", type = "", resumo = "", source = "", year = "",
            keywords = "", link = "", arxiv = "", doi = "", isbn = "", issn = "", pmid = "", scopus = "",
            pui = "", pii = "", sgr = "";
      String first_name_authors = "", last_name_authors = "";
      String month = "", day = "", revision = "", pages = "", volume = "", issue = "",
            websites = "", publisher = "", city = "", edition = "", institution = "",
            series = "", chapter = "", editors = "", file_attached = "";
      String open_access = Config.OPEN_ACCESS;
      int reader_count = 0, group_count = 0;
      String reader_count_by_academic_status = "", reader_count_by_subdiscipline = "",
            reader_count_by_country = "", reader_count_by_user_role = "",
            reader_count_by_subject_area = "";
      int docCount = 0, docCount1 = 1, numberCount = Config.START_NUMBER_CONVERT, count = 0,
countAuthors = 1;
      int c2 = 0;
      DateTimeFormatter fmt = DateTimeFormatter.ofPattern("HH:mm:ss");

      String chave = ""; //Variavel responsavel pelo controle das condições. Ele pega a chave do json
      String chave2 = ""; //para keywords
      String countAbs = "";
      ConversorController conv = new ConversorController();
      //Processo de reconhecimento de arquivo
      File file = new File(Config.CAMINHO_ARQUIVO2 + Config.FILE_NAME_CONVERT +
Config.START_NUMBER_CONVERT + ".txt");
      FileReader fr = new FileReader(file);
      BufferedReader bf = new BufferedReader(fr);
      String arq = bf.readLine();
      //Arquivos de script
      File filescript = new File(Config.CAMINHO_SCRIPT + "document.sql");
      BufferedWriter bfSql = new BufferedWriter(new FileWriter(filescript));

      File filescript1 = new File(Config.CAMINHO_SCRIPT + "readers.sql");
      BufferedWriter bfSql1 = new BufferedWriter(new FileWriter(filescript1));

      File filescript2 = new File(Config.CAMINHO_SCRIPT + "aditional_fields.sql");
      BufferedWriter bfSql2 = new BufferedWriter(new FileWriter(filescript2));

      File filescript3 = new File(Config.CAMINHO_SCRIPT + "resumo_document.sql");
      BufferedWriter bfSql3 = new BufferedWriter(new FileWriter(filescript3));

      File filescript4 = new File(Config.CAMINHO_SCRIPT + "keywords_document.sql");
      BufferedWriter bfsql4 = new BufferedWriter(new FileWriter(filescript4));

      File filescript5 = new File(Config.CAMINHO_SCRIPT + "authors.sql");
      BufferedWriter bfSql5 = new BufferedWriter(new FileWriter(filescript5));

      File filescript6 = new File(Config.CAMINHO_SCRIPT + "identifiers.sql");
      BufferedWriter bfSql6 = new BufferedWriter(new FileWriter(filescript6));
      String[] control = new String[2];
      int c = 0;
```

```
        System.out.println("Criando Scripts para banco de dados, aguarde...");
        //O while controlara as readLine pra saber se tem null. Se tiver null o processo para
        while (arq != null) {
           String[] tokens = arq.split(",");
           for (int i = 0; i < tokens.length; i++) {
              String[] newTokens = tokens[i].split(":");

              if ("[{\"title\"".equals(newTokens[0]) || "{\"title\"".equals(newTokens[0])) {
                 chave = newTokens[0];
                 if (newTokens[0].equals("[{\"title\"") || newTokens[0].equals("{\"title\"")) {
                    for (int k = 1; k < newTokens.length; k++) {
                       if (newTokens[k] != null) {
                          title = title + newTokens[k];
                       }
                       if (k == (newTokens.length - 2)) {
                          title = title + ":";
                       }
                    }
                 }
              }
              else if ("[{\"title\"" != newTokens[0] && "[{\"title\"".equals(chave) || "{\"title\"" != newTokens[0] &&
"{\"title\"".equals(chave)) {
                 if ("\"type\"".equals(newTokens[0])) {
                    chave = newTokens[0];
                    type = newTokens[1];

                 } else if ("\"pages\"".equals(newTokens[0])) {
                    chave = newTokens[0];
                    pages = newTokens[1];
                 } else if ("\"authors\"".equals(newTokens[0])) {
                    chave = newTokens[0];
                    if (newTokens.length > 2) {
                       if ("[{\"first_name\"".equals(newTokens[1])) {
                          if (c == 0) {
                             c = 1;
                             chave2 = newTokens[0];
                             first_name_authors = first_name_authors + newTokens[2];
                          } else if (c == 1) {
                             chave2 = newTokens[0];
                             first_name_authors = first_name_authors + "," + newTokens[2];
                          }

                       } else if ("[{\"last_name\"".equals(newTokens[1])) {
                          if (c2 == 0) {
                             c2 = 1;
                             chave2 = newTokens[0];
                             last_name_authors = last_name_authors + newTokens[2];
                          } else if (c2 == 1) {
                             chave2 = newTokens[0];
                             last_name_authors = last_name_authors + "," + newTokens[2];
                          }

                       } else if ("{\"first_name\"".equals(newTokens[0])) {
                          if (c == 0) {
                             c = 1;
                             chave2 = newTokens[0];
                             first_name_authors = first_name_authors + newTokens[1] + " " + newTokens[2];
                          } else if (c == 1) {
                             chave2 = newTokens[0];
                             first_name_authors = first_name_authors + "," + newTokens[1] + " " + newTokens[2];
```

```
            }

        } else if ("{\"last_name\"".equals(newTokens[0])) {
            if (c2 == 0) {
                c2 = 1;
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c2 == 1) {
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("\"last_name\"".equals(newTokens[0])) {
            if (c2 == 0) {
                c2 = 1;
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c2 == 1) {
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("\"scopus_author_id\"".equals(newTokens[0])) {
            //System.out.println("vou fazer nada nao '-'");
        }
    } else if (newTokens.length == 2) {

        if ("{\"first_name\"".equals(newTokens[0])) {
            if (c == 0) {
                c = 1;
                chave2 = newTokens[0];
                first_name_authors = first_name_authors + newTokens[1];
            } else if (c == 1) {
                chave2 = newTokens[0];
                first_name_authors = first_name_authors + "," + newTokens[1];
            }

        } else if ("{\"last_name\"".equals(newTokens[0])) {
            if (c2 == 0) {
                c2 = 1;
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + newTokens[1];
            } else if (c2 == 1) {
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + "," + newTokens[1];
            }

        } else if ("\"last_name\"".equals(newTokens[0])) {
            if (c2 == 0) {
                c2 = 1;
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + newTokens[1];
            } else if (c2 == 1) {
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + "," + newTokens[1];
            }

        } else if ("\"scopus_author_id\"".equals(newTokens[0])) {
        }
    }
```

```
    } else {
        if (!title.equals("")) {
            title = title + " " + newTokens[0];
        }
        else {
            title = title + newTokens[0];

    }

    }
} else if ("\"type\"".equals(newTokens[0]) && chave != "\"type\"" && newTokens.length == 2) {
    chave = newTokens[0];
    type = newTokens[1];
}
else if ("\"year\"".equals(newTokens[0]) && newTokens.length == 2) {
    chave = newTokens[0];
    year = newTokens[1];
}
else if ("\"keywords\"".equals(newTokens[0])) {
    chave = newTokens[0];
    for (int j = 1; j < newTokens.length; j++) {
        if (newTokens[j] != null && j == (newTokens.length - 1)) {
            keywords = keywords + newTokens[j] + ",";
        } else {
            keywords = keywords + newTokens[j] + ":";
        }
    }
} else if ("\"keywords\"" != newTokens[0] && "\"keywords\"".equals(chave)) {
    chave2 = newTokens[0];
    if ("\"pages\"".equals(chave2) && newTokens.length == 2) {
        if (newTokens.length > 2) {
            chave = chave2;
            pages = newTokens[1];
        }
    } else if ("\"volume\"".equals(chave2) && newTokens.length == 2) {
        chave = chave2;
        volume = newTokens[1];

    } else if ("\"id\"".equals(chave2) && newTokens.length == 2) {
        chave = chave2;
        id = newTokens[1];
    } else if ("\"websites\"".equals(chave2) && newTokens.length == 2) {
        chave = chave2;
        websites = newTokens[1] + ":";
        for (int j = 2; j < newTokens.length; j++) {
            websites = websites + newTokens[j];
        }
    } else if ("\"issue\"".equals(chave2) && newTokens.length == 2) {
        chave = chave2;
        issue = newTokens[1];
    } else {
        for (int j = 0; j < newTokens.length; j++) {
            if (newTokens[j] != null && j == (newTokens.length - 1)) {
                keywords = keywords + newTokens[j] + ",";
            } else {
                keywords = keywords + newTokens[j] + ":";
            }
        }
    }
}
```

```
else if ("\"websites\"".equals(newTokens[0])) {
   chave = newTokens[0];
   websites = newTokens[1] + ":";
   for (int j = 2; j < newTokens.length; j++) {
      websites = websites + newTokens[j];
   }
} else if ("\"websites\"" != newTokens[0] && "\"websites\"".equals(chave)) {
   if ("\"id\"".equals(newTokens[0])) {
      chave = newTokens[0];
      id = newTokens[1];
   } else if ("\"edition\"".equals(newTokens[0])) {
      chave = newTokens[0];
      edition = newTokens[1];
   } else if ("\"file_attached\"".equals(newTokens[0])) {
      chave = newTokens[0];
      file_attached = newTokens[1];
   } else if ("\"institution\"".equals(newTokens[0])) {
      chave = newTokens[0];
      institution = newTokens[1];
   } else if ("\"day\"".equals(newTokens[0])) {
      chave = newTokens[0];
      day = newTokens[1];
   } else if ("\"month\"".equals(newTokens[0])) {
      chave = newTokens[0];
      month = newTokens[1];
   } else if ("\"publisher\"".equals(newTokens[0])) {
      chave = newTokens[0];
      publisher = newTokens[1];
   } else if ("\"accessed\"".equals(newTokens[0])) {
   } else {
   }
} else if ("\"month\"".equals(newTokens[0])) {
   chave = newTokens[0];
   month = newTokens[1];

}
else if ("\"pages\"".equals(newTokens[0])) {
   chave = newTokens[0];
   pages = newTokens[1];
}
else if ("\"volume\"".equals(newTokens[0])) {
   chave = newTokens[0];
   volume = newTokens[1];
}
else if ("\"day\"".equals(newTokens[0])) {
   chave = newTokens[0];
   day = newTokens[1];
}
else if ("\"revision\"".equals(newTokens[0])) {
   chave = newTokens[0];
   revision = newTokens[1];
}
else if ("\"pages\"".equals(newTokens[0]) && chave != "\"pages\"") {
   chave = newTokens[0];
   pages = newTokens[1];
}
else if ("\"issue\"".equals(newTokens[0])) {
   chave = newTokens[0];
   issue = newTokens[1];
}
```

```
else if ("\"city\"" != newTokens[0] && "\"city\"".equals(chave)) {
   if ("\"id\"".equals(newTokens[0])) {
      chave = newTokens[0];
      id = newTokens[1];

   } else if ("\"series\"".equals(newTokens[0])) {
      chave = newTokens[0];
      series = series + newTokens[0];
   } else if ("\"editors\"".equals(newTokens[0])) {
      editors = newTokens[1];
      chave = newTokens[0];
   } else {
      city = city + "," + newTokens[0];
   }
} else if ("\"city\"".equals(newTokens[0])) {
   chave = newTokens[0];
   city = newTokens[1];
}
else if ("\"edition\"".equals(newTokens[0])) {
   chave = newTokens[0];
   edition = newTokens[1];
}
else if ("\"series\"".equals(newTokens[0])) {
   chave = newTokens[0];
   series = newTokens[1];
}
else if ("\"chapter\"".equals(newTokens[0])) {
   chave = newTokens[0];
   chapter = newTokens[1];
}
else if (!"\"editors\"".equals(newTokens[0]) && "\"editors\"".equals(chave)) {
   if (newTokens.length > 1) {
      if ("[{\"first_name\"".equals(newTokens[1]) || "[{\"last_name\"".equals(newTokens[1])) {
         editors = "";
      } else if (!"[{\"first_name\"".equals(newTokens[1]) || !"[{\"last_name\"".equals(newTokens[1])) {
         if ("\"id\"".equals(newTokens[0])) {
            chave = newTokens[0];
            id = newTokens[1];
         }
      }
   } else if (newTokens.length == 1) {
      if ("[{\"first_name\"".equals(newTokens[0]) || "[{\"last_name\"".equals(newTokens[0])
            || "{\"first_name\"".equals(newTokens[0]) || "{\"last_name\"".equals(newTokens[0])
            || "\"first_name\"".equals(newTokens[0]) || "\"last_name\"".equals(newTokens[0])) {
         editors = "";
      } else if (!"[{\"first_name\"".equals(newTokens[0]) || !"[{\"last_name\"".equals(newTokens[0])
            || !"{\"first_name\"".equals(newTokens[0]) || !"{\"last_name\"".equals(newTokens[0])
            || !"\"first_name\"".equals(newTokens[0]) || !"\"last_name\"".equals(newTokens[0])
            || !"[{\"first_name\"".equals(newTokens[0]) || !"[{\"last_name\"".equals(newTokens[0])) {
         if ("\"id\"".equals(newTokens[0])) {
            chave = newTokens[0];
            id = newTokens[1];
         }
      }
   }
} else if ("\"editors\"".equals(newTokens[0])) {
   chave = newTokens[0];
   editors = newTokens[1];
}
else if ("\"institution\"".equals(newTokens[0])) {
```

```
    chave = newTokens[0];
    institution = newTokens[1];

} else if ("\"institution\"" != newTokens[0] && "\"institution\"".equals(chave)) {
    if ("\"id\"".equals(newTokens[0])) {
        chave = newTokens[0];
        id = newTokens[1];

    } else {
        institution = institution + "," + newTokens[0];
    }
}
else if ("\"publisher\"".equals(newTokens[0])) {
    chave = newTokens[0];
    publisher = newTokens[1];

} else if ("\"file_attached\"".equals(newTokens[0])) {
    chave = newTokens[0];
    file_attached = newTokens[1];

} else if ("\"source\"".equals(newTokens[0])) {
    chave = newTokens[0];
    if (newTokens.length < 3) {
        source = newTokens[1];
    } else {
        for (int m = 1; m < newTokens.length; m++) {
            if (m == 1) {
                source = source + newTokens[m] + ":";
            } else {
                source = source + newTokens[m];
            }
        }
    }

} else if ("\"accessed\"".equals(newTokens[0])) {
} else if ("\"id\"".equals(newTokens[0]) || "\"id\"".equals(chave)) {
    if ("\"id\"".equals(newTokens[0])) {
        chave = newTokens[0];
        id = newTokens[1];
    } else if ("\"id\"".equals(chave)) {
    }

}
else if ("\"abstract\"".equals(newTokens[0])) {
    chave = newTokens[0];
    for (int k = 1; k < newTokens.length; k++) {
        if (newTokens[k] != null) {
            countAbs = countAbs + newTokens[k];
        }
    }
} else if ("\"abstract\"" != newTokens[0] && "\"abstract\"".equals(chave)) {
    chave2 = newTokens[0];
    if ("\"link\"".equals(chave2)) {
        chave = chave2;
        link = newTokens[1] + ":";
        for (int j = 2; j < newTokens.length; j++) {
            link = link + newTokens[j];
        }
        resumo = countAbs;
    } else {
```

```
                for (int k = 0; k < newTokens.length; k++) {
                    if (newTokens[k] != null) {
                        countAbs = countAbs + newTokens[k];
                    }
                }
            }
        }
        else if ("\"identifiers\"".equals(newTokens[0]) || "\"arxiv\"".equals(newTokens[0]) ||
"\"doi\"".equals(newTokens[0])
                || "\"isbn\"".equals(newTokens[0]) || "\"issn\"".equals(newTokens[0]) ||
"\"pmid\"".equals(newTokens[0])
                || "\"scopus\"".equals(newTokens[0]) || "\"pui\"".equals(newTokens[0]) ||
"\"sgr\"".equals(newTokens[0])) {
            chave = newTokens[0];
            if ("\"identifiers\"".equals(chave)) {
                if ("{\"arxiv\"".equals(newTokens[1])) {
                    arxiv = newTokens[2];
                } else if ("{\"doi\"".equals(newTokens[1])) {
                    doi = newTokens[2];
                } else if ("{\"isbn\"".equals(newTokens[1])) {
                    isbn = newTokens[2];
                } else if ("{\"issn\"".equals(newTokens[1])) {
                    issn = newTokens[2];
                } else if ("{\"pmid\"".equals(newTokens[1])) {
                    pmid = newTokens[2];
                } else if ("{\"scopus\"".equals(newTokens[1])) {
                    scopus = newTokens[2];
                } else if ("{\"pui\"".equals(newTokens[1])) {
                    pui = newTokens[2];
                } else if ("{\"sgr\"".equals(newTokens[1])) {
                    sgr = newTokens[2];
                }
            } else if ("\"arxiv\"".equals(chave)) {
                arxiv = newTokens[1];
            } else if ("\"doi\"".equals(chave)) {
                doi = newTokens[1];
            } else if ("\"isbn\"".equals(chave)) {
                isbn = newTokens[1];
            } else if ("\"issn\"".equals(chave)) {
                issn = newTokens[1];
            } else if ("\"pmid\"".equals(chave)) {
                pmid = newTokens[1];
            } else if ("\"scopus\"".equals(chave)) {
                scopus = newTokens[1];
            } else if ("\"pui\"".equals(chave)) {
                pui = newTokens[1];
            } else if ("\"pii\"".equals(chave)) {
                pii = newTokens[1];
            } else if ("\"sgr\"".equals(chave)) {
                sgr = newTokens[1];
            }
        } else if ("\"Business".equals(newTokens[0]) || "{".equals(newTokens[0]) ||
"\"Biochemistry".equals(newTokens[0]) || "\"Economics".equals(newTokens[0])
                || "\"Pharmacology".equals(newTokens[0])) {
            if ("\"reader_count_by_subject_area\"".equals(chave)) {
                chave2 = newTokens[0];
                reader_count_by_subject_area = reader_count_by_subject_area + "," + newTokens[0];
            } else if ("\"reader_count_by_subdiscipline\"".equals(chave)) {
                chave2 = newTokens[0];
                reader_count_by_subdiscipline = reader_count_by_subdiscipline + "," + newTokens[0];
```

```
    }
} else if ("\"authors\"".equals(newTokens[0]) || "{\"first_name\"".equals(newTokens[0])
        || "{\"last_name\"".equals(newTokens[0]) || "\"last_name\"".equals(newTokens[0])
        || "\"scopus_author_id\"".equals(newTokens[0])) {
    chave = newTokens[0];
    if (newTokens.length > 2) {
        if ("[{\"first_name\"".equals(newTokens[1])) {
            if (c == 0) {
                c = 1;
                chave2 = newTokens[0];
                first_name_authors = first_name_authors + newTokens[2];
            } else if (c == 1) {
                chave2 = newTokens[0];
                first_name_authors = first_name_authors + "," + newTokens[2];
            }

        } else if ("[{\"last_name\"".equals(newTokens[1])) {
            if (c2 == 0) {
                c2 = 1;
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + newTokens[2];
            } else if (c2 == 1) {
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + "," + newTokens[2];
            }

        } else if ("{\"first_name\"".equals(newTokens[0])) {
            if (c == 0) {
                c = 1;
                chave2 = newTokens[0];
                first_name_authors = first_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c == 1) {
                chave2 = newTokens[0];
                first_name_authors = first_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("{\"last_name\"".equals(newTokens[0])) {
            if (c2 == 0) {
                c2 = 1;
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c2 == 1) {
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("\"last_name\"".equals(newTokens[0])) {
            if (c2 == 0) {
                c2 = 1;
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c2 == 1) {
                chave2 = newTokens[0];
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("\"scopus_author_id\"".equals(newTokens[0])) {
        }
    } else if (newTokens.length == 2) {
```

```
        if ("{\"first_name\"".equals(newTokens[0])) {
          if (c == 0) {
            c = 1;
            chave2 = newTokens[0];
            first_name_authors = first_name_authors + newTokens[1];
          } else if (c == 1) {
            chave2 = newTokens[0];
            first_name_authors = first_name_authors + "," + newTokens[1];
          }

        } else if ("{\"last_name\"".equals(newTokens[0])) {
          if (c2 == 0) {
            c2 = 1;
            chave2 = newTokens[0];
            last_name_authors = last_name_authors + newTokens[1];
          } else if (c2 == 1) {
            chave2 = newTokens[0];
            last_name_authors = last_name_authors + "," + newTokens[1];
          }

        } else if ("\"last_name\"".equals(newTokens[0])) {
          if (c2 == 0) {
            c2 = 1;
            chave2 = newTokens[0];
            last_name_authors = last_name_authors + newTokens[1];
          } else if (c2 == 1) {
            chave2 = newTokens[0];
            last_name_authors = last_name_authors + "," + newTokens[1];
          }

        } else if ("\"scopus_author_id\"".equals(newTokens[0])) {
        }
      }
    } else if (!"\"authors\"".equals(newTokens[0]) && "\"authors\"".equals(chave)
        || !"\"authors\"".equals(newTokens[0]) && "{\"first_name\"".equals(chave)
        || !"\"authors\"".equals(newTokens[0]) && "{\"last_name\"".equals(chave)
        || !"\"authors\"".equals(newTokens[0]) && "\"last_name\"".equals(chave)
        || !"\"authors\"".equals(newTokens[0]) && "\"scopus_author_id\"".equals(chave)
        || !"{\"first_name\"".equals(newTokens[0]) && "\"authors\"".equals(chave)
        || !"{\"first_name\"".equals(newTokens[0]) && "{\"first_name\"".equals(chave)
        || !"{\"first_name\"".equals(newTokens[0]) && "{\"last_name\"".equals(chave)
        || !"{\"first_name\"".equals(newTokens[0]) && "\"last_name\"".equals(chave)
        || !"{\"first_name\"".equals(newTokens[0]) && "\"scopus_author_id\"".equals(chave)
        || !"{\"last_name\"".equals(newTokens[0]) && "\"authors\"".equals(chave)
        || !"{\"last_name\"".equals(newTokens[0]) && "{\"first_name\"".equals(chave)
        || !"{\"last_name\"".equals(newTokens[0]) && "{\"last_name\"".equals(chave)
        || !"{\"last_name\"".equals(newTokens[0]) && "\"last_name\"".equals(chave)
        || !"{\"last_name\"".equals(newTokens[0]) && "\"scopus_author_id\"".equals(chave)
        || !"\"last_name\"".equals(newTokens[0]) && "\"authors\"".equals(chave)
        || !"\"last_name\"".equals(newTokens[0]) && "{\"first_name\"".equals(chave)
        || !"\"last_name\"".equals(newTokens[0]) && "{\"last_name\"".equals(chave)
        || !"\"last_name\"".equals(newTokens[0]) && "\"last_name\"".equals(chave)
        || !"\"last_name\"".equals(newTokens[0]) && "\"scopus_author_id\"".equals(chave)
        || !"\"scopus_author_id\"".equals(newTokens[0]) && "\"authors\"".equals(chave)
        || !"\"scopus_author_id\"".equals(newTokens[0]) && "{\"first_name\"".equals(chave)
        || !"\"scopus_author_id\"".equals(newTokens[0]) && "{\"last_name\"".equals(chave)
        || !"\"scopus_author_id\"".equals(newTokens[0]) && "\"last_name\"".equals(chave)
        || !"\"scopus_author_id\"".equals(newTokens[0]) && "\"scopus_author_id\"".equals(chave)) {
      if ("\"year\"".equals(newTokens[0])) {
        chave = newTokens[0];
```

```
                year = newTokens[1];
            } else if ("\"identifiers\"".equals(newTokens[0]) || "\"arxiv\"".equals(newTokens[0]) ||
"\"doi\"".equals(newTokens[0])
                || "\"isbn\"".equals(newTokens[0]) || "\"issn\"".equals(newTokens[0]) ||
"\"pmid\"".equals(newTokens[0])
                || "\"scopus\"".equals(newTokens[0]) || "\"pui\"".equals(newTokens[0]) ||
"\"sgr\"".equals(newTokens[0])) {
                chave = newTokens[0];
                if ("\"identifiers\"".equals(chave)) {
                   if ("{\"arxiv\"".equals(newTokens[1])) {
                      arxiv = newTokens[2];
                   } else if ("{\"doi\"".equals(newTokens[1])) {
                      doi = newTokens[2];
                   } else if ("{\"isbn\"".equals(newTokens[1])) {
                      isbn = newTokens[2];
                   } else if ("{\"issn\"".equals(newTokens[1])) {
                      issn = newTokens[2];
                   } else if ("{\"pmid\"".equals(newTokens[1])) {
                      pmid = newTokens[2];
                   } else if ("{\"scopus\"".equals(newTokens[1])) {
                      scopus = newTokens[2];
                   } else if ("{\"pui\"".equals(newTokens[1])) {
                      pui = newTokens[2];
                   } else if ("{\"sgr\"".equals(newTokens[1])) {
                      sgr = newTokens[2];
                   }
                } else if ("\"arxiv\"".equals(chave)) {
                   arxiv = newTokens[1];
                } else if ("\"doi\"".equals(chave)) {
                   doi = newTokens[1];
                } else if ("\"isbn\"".equals(chave)) {
                   isbn = newTokens[1];
                } else if ("\"issn\"".equals(chave)) {
                   issn = newTokens[1];
                } else if ("\"pmid\"".equals(chave)) {
                   pmid = newTokens[1];
                } else if ("\"scopus\"".equals(chave)) {
                   scopus = newTokens[1];
                } else if ("\"pui\"".equals(chave)) {
                   pui = newTokens[1];
                } else if ("\"pii\"".equals(chave)) {
                   pii = newTokens[1];
                } else if ("\"sgr\"".equals(chave)) {
                   sgr = newTokens[1];
                }
             } else {
                for (int p = 0; p < newTokens.length; p++) {
                   if ("[{\"first_name\"".equals(newTokens[p]) || "[{\"last_name\"".equals(newTokens[p])
                         || "{\"first_name\"".equals(newTokens[p]) || "{\"last_name\"".equals(newTokens[p])
                         || "\"last_name\"".equals(newTokens[p]) || "\"scopus_author_id\"".equals(newTokens[p]))
{
                      chave2 = newTokens[p];
                   }
                }
                if (newTokens.length > 2) {

                   if ("[{\"first_name\"".equals(chave2)) {
                      if (c == 0) {
                         c = 1;
                         chave = chave2;
```

```
                first_name_authors = first_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c == 1) {
                chave = chave2;
                first_name_authors = first_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("[{\"last_name\"".equals(chave2)) {
            if (c2 == 0) {
                c2 = 1;
                chave = chave2;
                last_name_authors = last_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c2 == 1) {
                chave = chave2;
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("{\"first_name\"".equals(chave2)) {
            if (c == 0) {
                c = 1;
                chave = chave2;
                first_name_authors = first_name_authors + newTokens[1] + " " + newTokens[2] + ",";
            } else if (c == 1) {
                chave = chave2;
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("{\"last_name\"".equals(chave2)) {
            if (c2 == 0) {
                c2 = 1;
                chave = chave2;
                last_name_authors = last_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c2 == 1) {
                chave = chave2;
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("\"last_name\"".equals(chave2)) {
            if (c2 == 0) {
                c2 = 1;
                chave = chave2;
                last_name_authors = last_name_authors + newTokens[1] + " " + newTokens[2];
            } else if (c2 == 1) {
                chave = chave2;
                last_name_authors = last_name_authors + "," + newTokens[1] + " " + newTokens[2];
            }

        } else if ("\"scopus_author_id\"".equals(chave2)) {
            chave = chave2;
        }
    } else if (newTokens.length == 2) {
        if ("[{\"first_name\"".equals(chave2)) {
            if (c == 0) {
                c = 1;
                chave = chave2;
                first_name_authors = first_name_authors + newTokens[1];
            } else if (c == 1) {
                chave = chave2;
                first_name_authors = first_name_authors + "," + newTokens[1];
            }
```

```
    } else if ("[{\"last_name\"".equals(chave2)) {
      if (c2 == 0) {
        c2 = 1;
        chave = chave2;
        last_name_authors = last_name_authors + newTokens[1];
      } else if (c2 == 1) {
        chave = chave2;
        last_name_authors = last_name_authors + "," + newTokens[1];
      }

    } else if ("{\"first_name\"".equals(chave2)) {
      if (c == 0) {
        c = 1;
        chave = chave2;
        first_name_authors = first_name_authors + newTokens[1];
      } else if (c == 1) {
        chave = chave2;
        first_name_authors = first_name_authors + " " + newTokens[1];
      }

    } else if ("{\"last_name\"".equals(chave2)) {
      if (c2 == 0) {
        c2 = 1;
        chave = chave2;
        last_name_authors = last_name_authors + newTokens[1];
      } else if (c2 == 1) {
        chave = chave2;
        last_name_authors = last_name_authors + "," + newTokens[1];
      }

    } else if ("\"last_name\"".equals(chave2)) {
      if (c2 == 0) {
        c2 = 1;
        chave = chave2;
        last_name_authors = last_name_authors + newTokens[1];
      } else if (c2 == 1) {
        chave = chave2;
        last_name_authors = last_name_authors + "," + newTokens[1];
      }

    } else if ("\"scopus_author_id\"".equals(chave2)) {
      chave = chave2;
    }
  } else if (newTokens.length == 1) {
    if ("[{\"first_name\"".equals(chave2)) {
      if (c == 0) {
        c = 1;
        chave = chave2;
        first_name_authors = first_name_authors + newTokens[0];
      } else if (c == 1) {
        chave = chave2;
        first_name_authors = first_name_authors + " " + newTokens[0];
      }

    } else if ("[{\"last_name\"".equals(chave2)) {
      if (c2 == 0) {
        c2 = 1;
        chave = chave2;
        last_name_authors = last_name_authors + newTokens[0];
      } else if (c2 == 1) {
```

```
                        chave = chave2;
                        last_name_authors = last_name_authors + " " + newTokens[0];
                    }

                } else if ("{\"first_name\"".equals(chave2)) {
                    if (c == 0) {
                        c = 1;
                        chave = chave2;
                        first_name_authors = first_name_authors + newTokens[0];
                    } else if (c == 1) {
                        chave = chave2;
                        first_name_authors = first_name_authors + " " + newTokens[0];
                    }

                } else if ("{\"last_name\"".equals(chave2)) {
                    if (c2 == 0) {
                        c2 = 1;
                        chave = chave2;
                        last_name_authors = last_name_authors + newTokens[0];
                    } else if (c2 == 1) {
                        chave = chave2;
                        last_name_authors = last_name_authors + " " + newTokens[0];
                    }

                } else if ("\"last_name\"".equals(chave2)) {
                    if (c2 == 0) {
                        c2 = 1;
                        chave = chave2;
                        last_name_authors = last_name_authors + newTokens[0];
                    } else if (c2 == 1) {
                        chave = chave2;
                        last_name_authors = last_name_authors + "" + newTokens[0];
                    }

                } else if ("\"scopus_author_id\"".equals(chave2)) {
                    chave = chave2;
                }
            }

        }
    }
    else if ("\"link\"".equals(newTokens[0])) {
        chave = newTokens[0];
        link = newTokens[1] + ":" + newTokens[2];
    }
    else if ("\"reader_count\"".equals(newTokens[0])) {
        chave = newTokens[0];
        reader_count = Integer.parseInt(newTokens[1]);
    } else if ("\"reader_count_by_academic_status\"".equals(newTokens[0])) {
        chave = newTokens[0];
        reader_count_by_academic_status = reader_count_by_academic_status + newTokens[1] + ":" +
newTokens[2];
    } else if ("\"reader_count_by_academic_status\"".equals(chave)) {
        chave2 = newTokens[0];
        if ("\"reader_count_by_user_role\"".equals(chave2)) {
            chave = chave2;
            reader_count_by_user_role = reader_count_by_user_role + newTokens[1] + ":" + newTokens[2];
        } else if ("\"reader_count_by_subject_area\"".equals(chave2)) {
            chave = chave2;
```

```
                reader_count_by_subject_area = reader_count_by_subject_area + newTokens[1] + ":" +
newTokens[2];
             } else if ("\"reader_count_by_subdiscipline\"".equals(chave2)) {
                chave = chave2;
                reader_count_by_subdiscipline = reader_count_by_subdiscipline + newTokens[1] + ":" +
newTokens[2] + ":" + newTokens[3];
             } else if ("\"reader_count_by_country\"".equals(chave2)) {
                chave = chave2;
                reader_count_by_country = reader_count_by_country + newTokens[1] + ":" + newTokens[2];
             } else if ("\"group_count\"".equals(chave2)) {
                chave = chave2;
                newTokens[1] = newTokens[1].replace("}", "");
                newTokens[1] = newTokens[1].replace("]", "");
                group_count = Integer.parseInt(newTokens[1]);
             } else {
                reader_count_by_academic_status = reader_count_by_academic_status + "," + newTokens[0] +
":" + newTokens[1];
             }
          } else if ("\"reader_count_by_user_role\"".equals(chave)) {
             chave2 = newTokens[0];
             if ("\"reader_count_by_subject_area\"".equals(chave2)) {
                chave = chave2;
                for (int p = 0; p < newTokens.length; p++) {
                   if (newTokens.length > 2) {
                      reader_count_by_subject_area = reader_count_by_subject_area + newTokens[1] + ":" +
newTokens[2];
                      p = newTokens.length - 1;
                   } else if (newTokens.length == 2) {
                      reader_count_by_subject_area = reader_count_by_subject_area + newTokens[1];
                      p = newTokens.length - 1;
                   } else if (newTokens.length == 1) {
                      reader_count_by_subject_area = reader_count_by_subject_area + newTokens[0];
                   }
                }
             } else if ("\"reader_count_by_subdiscipline\"".equals(chave2)) {
                chave = chave2;
                reader_count_by_subdiscipline = reader_count_by_subdiscipline + newTokens[1] + ":" +
newTokens[2] + ":" + newTokens[3];
             } else if ("\"reader_count_by_country\"".equals(chave2)) {
                chave = chave2;
                reader_count_by_country = reader_count_by_country + newTokens[1] + ":" + newTokens[2];
             } else if ("\"group_count\"".equals(chave2)) {
                chave = chave2;
                newTokens[1] = newTokens[1].replace("}", "");
                newTokens[1] = newTokens[1].replace("]", "");
                group_count = Integer.parseInt(newTokens[1]);
             } else {
                reader_count_by_user_role = reader_count_by_user_role + "," + newTokens[0] + ":" +
newTokens[1];
             }
          } else if ("\"reader_count_by_subject_area\"".equals(chave)) {
             chave2 = newTokens[0];
             if ("\"reader_count_by_subdiscipline\"".equals(chave2)) {
                chave = chave2;
                for (int p = 0; p < newTokens.length; p++) {
                   if (newTokens.length == 2) {
                      reader_count_by_subdiscipline = reader_count_by_subdiscipline + newTokens[1];
                      p = newTokens.length - 1;
                   } else if (newTokens.length > 2) {
```

```
                    reader_count_by_subdiscipline = reader_count_by_subdiscipline + newTokens[1] + ":" +
newTokens[2] + ":" + newTokens[3];
                    p = newTokens.length - 1;
                  }
                }

              } else if ("\"reader_count_by_country\"".equals(chave2)) {
                chave = chave2;
                reader_count_by_country = reader_count_by_country + newTokens[1] + ":" + newTokens[2];
              } else if ("\"group_count\"".equals(chave2)) {
                chave = chave2;
                newTokens[1] = newTokens[1].replace("}", "");
                newTokens[1] = newTokens[1].replace("]", "");
                group_count = Integer.parseInt(newTokens[1]);
              } else if ("reader_count_by_subject_area".equals(chave) && newTokens.length == 1) {
                reader_count_by_subject_area = reader_count_by_subject_area + "," + newTokens[0];
              } else {
                for (int p = 0; p < newTokens.length; p++) {
                  if (newTokens.length > 1) {
                    reader_count_by_subject_area = reader_count_by_subject_area + "," + newTokens[0] + ":"
+ newTokens[1];
                    p = (newTokens.length - 1);
                  } else {
                    reader_count_by_subject_area = reader_count_by_subject_area + "," + newTokens[p];
                  }
                }
              }
            } else if ("\"reader_count_by_subdiscipline\"".equals(chave)) {
              chave2 = newTokens[0];
              if ("\"reader_count_by_country\"".equals(chave2)) {
                chave = chave2;
                reader_count_by_country = reader_count_by_country + newTokens[1] + ":" + newTokens[2];
              } else if ("\"group_count\"".equals(chave2)) {
                chave = chave2;
                newTokens[1] = newTokens[1].replace("}", "");
                newTokens[1] = newTokens[1].replace("]", "");
                newTokens[1] = newTokens[1].replace("[", "");
                newTokens[1] = newTokens[1].replace("{", "");
                group_count = Integer.parseInt(newTokens[1]);
              } else {
                for (int p = 0; p < newTokens.length; p++) {
                  if (newTokens.length == 1) {
                    reader_count_by_subdiscipline = reader_count_by_subdiscipline + "," + newTokens[p];

                  } else if (newTokens.length == 2) {
                    reader_count_by_subdiscipline = reader_count_by_subdiscipline + newTokens[0] + ":" +
newTokens[1];
                    p = newTokens.length - 1;
                  } else if (newTokens.length == 3) {
                    reader_count_by_subdiscipline = reader_count_by_subdiscipline + "," + newTokens[0] + ":"
+ newTokens[1] + ":" + newTokens[2];
                    p = newTokens.length - 1;
                  }
                }

              }
            } else if ("\"reader_count_by_country\"".equals(chave)) {
              chave2 = newTokens[0];
              if ("\"group_count\"".equals(chave2)) {
                chave = chave2;
```

```
                newTokens[1] = newTokens[1].replace("}", "");
                newTokens[1] = newTokens[1].replace("]", "");
                newTokens[1] = newTokens[1].replace("[", "");
                newTokens[1] = newTokens[1].replace("{", "");
                group_count = Integer.parseInt(newTokens[1]);
              } else {
                reader_count_by_country = reader_count_by_country + "," + newTokens[0] + ":" +
newTokens[1];
              }
            }
          }
          last_name_authors = last_name_authors.replace("}", "");
          last_name_authors = last_name_authors.replace("{", "");
          last_name_authors = last_name_authors.replace("[", "");
          last_name_authors = last_name_authors.replace("]", "");
          last_name_authors = last_name_authors.replace("\"", "|");
          last_name_authors = last_name_authors.replace("\\", "|");

          first_name_authors = first_name_authors.replace("}", "");
          first_name_authors = first_name_authors.replace("{", "");
          first_name_authors = first_name_authors.replace("[", "");
          first_name_authors = first_name_authors.replace("]", "");
          first_name_authors = first_name_authors.replace("\"", "|");
          first_name_authors = first_name_authors.replace("\\", "|");

          String[] fn = first_name_authors.split(",");
          String[] ln = last_name_authors.split(",");
          reader_count_by_academic_status = reader_count_by_academic_status.replace("\"", "");
          reader_count_by_academic_status = reader_count_by_academic_status.replace("{", "");
          reader_count_by_academic_status = reader_count_by_academic_status.replace("}", "");

          reader_count_by_user_role = reader_count_by_user_role.replace("\"", "");
          reader_count_by_user_role = reader_count_by_user_role.replace("{", "");
          reader_count_by_user_role = reader_count_by_user_role.replace("}", "");

          reader_count_by_subject_area = reader_count_by_subject_area.replace("\"", "");
          reader_count_by_subject_area = reader_count_by_subject_area.replace("{", "");
          reader_count_by_subject_area = reader_count_by_subject_area.replace("}", "");

          reader_count_by_subdiscipline = reader_count_by_subdiscipline.replace("\"", "");
          reader_count_by_subdiscipline = reader_count_by_subdiscipline.replace("{", "");
          reader_count_by_subdiscipline = reader_count_by_subdiscipline.replace("}", "");

          reader_count_by_country = reader_count_by_country.replace("\"", "");
          reader_count_by_country = reader_count_by_country.replace("{", "");
          reader_count_by_country = reader_count_by_country.replace("}", "");

          //Limpeza de variaveis
          title = title.replace("\"", "");
          title = title.replace("/", "");
          title = title.replace("\\", "");

          type = type.replace("\"", "");
          type = type.replace("\\", "");

          id = id.replace("\"", "");
          id = id.replace("\\", "");

          source = source.replace("\"", "");
          source = source.replace("\\", "");
```

```
year = year.replace("\"", "");
year = year.replace("\\", "");

link = link.replace("\"", "");
link = link.replace("\\", "");

keywords = keywords.replace("[", "");
keywords = keywords.replace("]", "");
keywords = keywords.replace("\"", "");
keywords = keywords.replace("\\", "|");
//  System.out.println(keywords);
websites = websites.replace("[", "");
websites = websites.replace("]", "");
websites = websites.replace("\"", "");
websites = websites.replace("\\", "|");

resumo = resumo.replace("}", "");
resumo = resumo.replace("\\", "|");
resumo = resumo.replace("\"", "");

month = month.replace("\"", "");
month = month.replace("\\", "");

day = day.replace("\"", "");
day = day.replace("\\", "");

revision = revision.replace("\"", "");
revision = revision.replace("\\", "");

pages = pages.replace("\"", "");
pages = pages.replace("\\", "");

volume = volume.replace("\"", "");
volume = volume.replace("\\", "");

issue = issue.replace("\"", "");
issue = issue.replace("\\", "");

publisher = publisher.replace("\"", "");
publisher = publisher.replace("\\", "");

city = city.replace("\"", "");
city = city.replace("\\", "");

edition = edition.replace("\"", "");
edition = edition.replace("\\", "");

institution = institution.replace("\"", "");
institution = institution.replace("\\", "");

series = series.replace("\"", "");
series = series.replace("\\", "");

chapter = chapter.replace("\"", "");
chapter = chapter.replace("\\", "");

editors = editors.replace("\"", "");
editors = editors.replace("\\", "");
```

```
file_attached = file_attached.replace("\"", "");
file_attached = file_attached.replace("\\", "");

arxiv = arxiv.replace("\"", "");
arxiv = arxiv.replace("}", "");
arxiv = arxiv.replace("\\", "|");

doi = doi.replace("\"", "");
doi = doi.replace("}", "");
doi = doi.replace("\\", "|");

isbn = isbn.replace("\"", "");
isbn = isbn.replace("}", "");
isbn = isbn.replace("\\", "");

issn = issn.replace("\"", "");
issn = issn.replace("}", "");
issn = issn.replace("\\", "");

pmid = pmid.replace("\"", "");
pmid = pmid.replace("}", "");
pmid = pmid.replace("\\", "");

scopus = scopus.replace("\"", "");
scopus = scopus.replace("}", "");
scopus = scopus.replace("\\", "");

pui = pui.replace("\"", "");
pui = pui.replace("}", "");
pui = pui.replace("\\", "");

pii = pii.replace("\"", "");
pii = pii.replace("}", "");
pii = pii.replace("\\", "");

sgr = sgr.replace("\"", "");
sgr = sgr.replace("}", "");
sgr = sgr.replace("\\", "");

bfSql.append("insert into " + Config.DATABASE + ".document(id,id_document,title,type,source,year,link,month,reader_count) values(" + "\"" + docCount1 + "\","
        + "\"" + id + "\"," + "\"" + title + "\"," + "\"" + type + "\"," + "\"" + source + "\"," + "\"" + year + "\"," + "\"" + link + "\"," + "\"" + month + "\"," + "\"" + reader_count + "\"" + ");" + "\r\n");

bfSql1.append("insert into " + Config.DATABASE + ".readers(id,reader_count_by_academic_status,reader_count_by_subdiscipline,reader_count_by_user_role,"
        + "reader_count_by_subject_area,reader_count_by_country,group_count,documents_id) values(" + "\"" + docCount1 + "\","
        + "\"" + reader_count_by_academic_status + "\"," + "\"" + reader_count_by_subdiscipline + "\"," + "\"" + reader_count_by_user_role + "\","
        + "\"" + reader_count_by_subject_area + "\"," + "\"" + reader_count_by_country + "\"," + "\"" + group_count + "\"," + "\"" + docCount1 + "\"" + ");" + "\r\n");

bfSql2.append("insert into " + Config.DATABASE + ".aditional_fields(id,day,revision,pages,volume,issue,websites,publisher,city,edition,institution,"
        + "series,chapter,editors,file_attached,open_access,document_id) values(" + "\"" + docCount1 + "\"," + "\"" + day
        + "\"," + "\"" + revision + "\"," + "\"" + pages + "\"," + "\"" + volume + "\"," + "\"" + issue + "\"," + "\"" + websites + "\","
```

```
            + "\"" + publisher + "\"," + "\"" + city + "\"," + "\"" + edition + "\"," + "\"" + institution + "\"," + "\""
+ series + "\"," + "\"" + chapter + "\","
            + "\"" + editors + "\"," + "\"" + file_attached + "\"," + "\"" + open_access + "\"," + "\"" + docCount1
+ "\"" + ");" + "\r\n");

        bfSql3.append("insert into " + Config.DATABASE + ".resumo_document(id,resumo,document_id)
values(" + "\"" + docCount1 + "\"," + "\"" + resumo + "\"," + "\"" + docCount1 + "\"" + ");" + "\r\n");

        bfsql4.append("insert into " + Config.DATABASE + ".keywords_document(id,keywords,document_id)
values(" + "\"" + docCount1 + "\"," + "\"" + keywords + "\"," + "\"" + docCount1 + "\"" + ");" + "\r\n");

        for (int z = 0; z < fn.length; z++) {
          bfSql5.append("insert into " + Config.DATABASE +
".authors(id,first_name,last_name,idocumento,documents_id) values(" + "\"" + countAuthors + "\"," + "\"" +
fn[z] + "\"," + "\"" + ln[z] + "\"," + "\"" + docCount1 + "\"," + "\"" + docCount1 + "\"" + ");" + "\r\n");
          countAuthors++;
        }
        bfSql6.append("insert into " + Config.DATABASE +
".identifiers(id,arxiv,doi,isbn,issn,pmid,scopus,pui,pii,sgr,documents_id) values(" + "\"" + docCount1 + "\"," +
"\""
            + arxiv + "\"," + "\"" + doi + "\"," + "\"" + isbn + "\"," + "\"" + issn + "\"," + "\"" + pmid + "\"," +
"\"" + scopus + "\"," + "\"" + pui
            + "\"," + "\"" + pii + "\"," + "\"" + sgr + "\"," + "\"" + docCount1 + "\"" + ");" + "\r\n");
        //Limpeza das variaveis para percorrer um novo documento
        title = "";
        year = "";
        keywords = "";
        pages = "";
        volume = "";
        source = "";
        first_name_authors = "";
        last_name_authors = "";
        day = "";
        revision = "";
        month = "";
        editors = "";
        institution = "";
        publisher = "";
        issue = "";
        websites = "";
        city = "";
        edition = "";
        series = "";
        chapter = "";
        file_attached = "";
        id = "";
        resumo = "";
        countAbs = "";
        link = "";
        reader_count_by_academic_status = "";
        reader_count_by_user_role = "";
        reader_count_by_subject_area = "";
        reader_count_by_subdiscipline = "";
        reader_count_by_country = "";
        arxiv = "";
        doi = "";
        isbn = "";
        issn = "";
        pmid = "";
        scopus = "";
```

```
            pui = "";
            sgr = "";
            c = 0;
            c2 = 0;
            reader_count = 0;
            group_count = 0;
            arq = bf.readLine();
            docCount1++;
            if (arq == null) {
               LocalDateTime timePoint = LocalDateTime.now();
               System.out.println("Arquivo " + docCount + " lido as " + timePoint.format(fmt));
               docCount++;
               numberCount++;
               count = 0;
               if (docCount < Config.QTD_DOCUMENTS_CONVERT) {
                  file = new File(Config.CAMINHO_ARQUIVO2 + Config.FILE_NAME_CONVERT +
numberCount + ".txt");
                  fr = new FileReader(file);
                  bf = new BufferedReader(fr);
                  //if (docCount % 50 == 0) {
                  //}
                  arq = bf.readLine();
               } else {
                  arq = null;
               }
            }
         }
         System.out.println("Todos os documentos lidos");
         bfSql.close();
         bfSql1.close();
         bfSql2.close();
         bfSql3.close();
         bfsql4.close();
         bfSql5.close();
         bfSql6.close();
      }
   }
```

Apêndice D – Algoritmo para seleção dos documentos cadastrados no banco de dados

```
public class SelecinarDocumentos {
public static void main(String[] args) throws IOException {
        System.out.println("Carregando dados iniciais...");
        Scanner entrada = new Scanner (System.in);
        Authors autor = new Authors();
        int contarIndiceAut = 0;
        DocumentoDAO docDao = new DocumentoDAO();
        AuthorsDAO autDao = new AuthorsDAO();
        KeywordsDocumentDAO keyDao = new KeywordsDocumentDAO();
        ResumoDocumentDAO resDao = new ResumoDocumentDAO();

        Document doct = new Document();
        int contarIndiceAut2 = 0;
        System.out.println("Partida");
        contarIndiceAut = entrada.nextInt();

        List<Document> doc = docDao.obterFaixa();
        System.out.println("...");
        System.out.println(".......");
        System.out.println(".......");
        List<Authors> aut = autDao.obterFaixa();
        System.out.println(".......");
        List<KeywordsDocument> key = keyDao.obterFaixa();
        System.out.println(".....");
        List<ResumoDocument> res = resDao.obterFaixa();
        System.out.println(".....");

        System.out.println(doc.size());
        System.out.println(aut.size());
        System.out.println(key.size());
        System.out.println(res.size());

        for (int i = 0; i < aut.size(); i++) {
           if ((aut.get(i).getIdocumento() - 1) == contarIndiceAut) {
              contarIndiceAut++;
              contarIndiceAut2++;
           }

        }
        System.out.println("Autor soma: " + contarIndiceAut);
        System.out.println("Autor por índice: " + contarIndiceAut2);

        int qtdDocAtrib = 0, docCount = 1, countAutEmpty = 0, countAutFull = 0;
        int countAuthors = 0;
        String p;

        System.out.println("Valor de p (onde irá salvar o documento)");
        p = entrada.next();
        System.out.println("id de Authors (pegue o id da ultima execução. Se é a primeira vez, digite 0)");
        countAuthors = entrada.nextInt();
        //qtdDocAtrib é muito importante. Deixar o contador inicialmente no 0, apos ler 100000, deixar 100000.
        //A variavel automaticamente contara 100001, então nao precisa iniciar ela com esse valor
        System.out.println("Escolha uma opção: ");
        System.out.println("1 - Coletar o total de documentos que contem todas as variáveis do problema
preenchidas");
        System.out.println("2 - Coletar e iniciar o processo de seleção");
```

```
int op = entrada.nextInt();
if (op == 1) {


   System.out.println("Selecionando o total de documentos que contem as variaveis, aguarde...");

   for (int i = 0; i < doc.size(); i++) {
      //Segundo for é para tratamento dos authors
      countAutEmpty = 0;
      for (int j = 0; j < aut.size(); j++) {
         if (aut.get(j).getIdocumento() == doc.get(i).getId()) {
            if (aut.get(j).getLastName().isEmpty()) {
               countAutEmpty = 1;
            }

         }
      }
      if (!doc.get(i).getTitle().isEmpty() && !doc.get(i).getYear().isEmpty()
            && !key.get(i).getKeywords().isEmpty()
            && !doc.get(i).getMonth().isEmpty()
            && !res.get(i).getResumo().isEmpty() && !doc.get(i).getType().isEmpty()
            && !doc.get(i).getSource().isEmpty() && doc.get(i).getReaderCount() != null
            && countAutEmpty == 0) {
         qtdDocAtrib++;
      }
   }
   System.out.println("Total de documentos contem todos os campos preenchidos: " + qtdDocAtrib);

} else if (op == 2) {
   //Inicialização de arquivos
   File fileTitle = new File(Config.CAMINHO_SELECAO + p + "/title.txt");
   BufferedWriter bfTitle = new BufferedWriter(new FileWriter(fileTitle));
   File filetype = new File(Config.CAMINHO_SELECAO + p + "/type.txt");
   BufferedWriter bfType = new BufferedWriter(new FileWriter(filetype));
   File fileyear = new File(Config.CAMINHO_SELECAO + p + "/year.txt");
   BufferedWriter bfyear = new BufferedWriter(new FileWriter(fileyear));
   File filesource = new File(Config.CAMINHO_SELECAO + p + "/source.txt");
   BufferedWriter bfsource = new BufferedWriter(new FileWriter(filesource));
   File fileauthors = new File(Config.CAMINHO_SELECAO + p + "/authors.txt");
   BufferedWriter bfauthors = new BufferedWriter(new FileWriter(fileauthors));
   File filekeywords = new File(Config.CAMINHO_SELECAO + p + "/keywords.txt");
   BufferedWriter bfkeywords = new BufferedWriter(new FileWriter(filekeywords));
   File idDocument = new File(Config.CAMINHO_SELECAO + p + "/id.txt");
   BufferedWriter bfIdDocument = new BufferedWriter(new FileWriter(idDocument));
   File filemonth = new File(Config.CAMINHO_SELECAO +  p +"/month.txt");
   BufferedWriter bfmonth = new BufferedWriter(new FileWriter(filemonth));
   File fileabstract = new File(Config.CAMINHO_SELECAO + p + "/abstract.txt");
   BufferedWriter bfabstract = new BufferedWriter(new FileWriter(fileabstract));
   File filereadercount = new File(Config.CAMINHO_SELECAO + p + "/reader_count.txt");
   BufferedWriter bfreadercount = new BufferedWriter(new FileWriter(filereadercount));
   System.out.println("Iniciando o processo de seleção, aguarde...");
   for (int i = 0; i < doc.size(); i++) {
      countAutEmpty = 0;
      for (int j = 0; j < aut.size(); j++) {
         if (aut.get(j).getIdocumento() == doc.get(i).getId()) {
            if (aut.get(j).getLastName().isEmpty()) {
               countAutEmpty = 1;
            }

         }
```

```
            }
            if (!doc.get(i).getTitle().isEmpty() && !doc.get(i).getYear().isEmpty()
                && !key.get(i).getKeywords().isEmpty()
                && !doc.get(i).getMonth().isEmpty()
                && !res.get(i).getResumo().isEmpty() && !doc.get(i).getType().isEmpty()
                && !doc.get(i).getSource().isEmpty() && doc.get(i).getReaderCount() != null
                && countAutEmpty == 0) {
              qtdDocAtrib++;
              countAuthors++;
              bfTitle.append(doc.get(i).getTitle() + "\r\n");
              bfType.append(doc.get(i).getType() + "\r\n");
              bfyear.append(doc.get(i).getYear() + "\r\n");
              bfmonth.append(doc.get(i).getMonth() + "\r\n");
              bfsource.append(doc.get(i).getSource() + "\r\n");
              bfkeywords.append(key.get(i).getKeywords() + "\r\n");
              bfabstract.append(res.get(i).getResumo() + "\r\n");
              bfreadercount.append(doc.get(i).getReaderCount() + "\r\n");
              bfIdDocument.append(doc.get(i).getId() + "\r\n");

              for (int j = 0; j < aut.size(); j++) {
                if (aut.get(j).getDocumentsId().getId() == doc.get(i).getId()) {
                  bfauthors.append(aut.get(j).getFirstName() + "\r\n");
                  bfauthors.append(aut.get(j).getLastName() + "\r\n");
                  bfauthors.append(countAuthors + "\r\n");
                }
              }

            }
          }
          bfTitle.close();
          bfType.close();
          bfyear.close();
          bfsource.close();
          bfkeywords.close();
          bfmonth.close();
          bfabstract.close();
          bfreadercount.close();
          bfauthors.close();
          bfIdDocument.close();
          System.out.println(qtdDocAtrib + " documentos foram selecionados. Acesse a pasta
DadosSelecionados");
          System.out.println("Atual: " + countAuthors);
        }

      }

}
```

Apêndice E – Algoritmo para remover documentos repetidos

```
public class RemoverDocumento {

    public static void main(String[] args) throws FileNotFoundException, IOException {

        int cout = 0, id = 0, countAuthorsNum = 1, countAuthorsArq = 0, control = 0;
        String countAuthors = "";
        ArrayList<String> dadosTitle = new ArrayList();
        ArrayList<String> dadosType = new ArrayList();
        ArrayList<Integer> dadosYear = new ArrayList();
        ArrayList<String> dadosSource = new ArrayList();
        ArrayList<String> dadosAuthors = new ArrayList();
        ArrayList<String> dadosKeywords = new ArrayList();
        ArrayList<Integer> dadosMonth = new ArrayList();
        ArrayList<String> dadosAbstract = new ArrayList();
        ArrayList<String> dadosReaderCount = new ArrayList();
        ArrayList<String> dadosIdDocument = new ArrayList();

        //Recuperando os arquivos selecionados na etapa anterior
        File fileTitle = new File(Config.CAMINHO_SELECAO + "title.txt");
        BufferedReader bfTitle = new BufferedReader(new FileReader(fileTitle));
        File filetype = new File(Config.CAMINHO_SELECAO + "type.txt");
        BufferedReader bfType = new BufferedReader(new FileReader(filetype));
        File fileyear = new File(Config.CAMINHO_SELECAO + "year.txt");
        BufferedReader bfyear = new BufferedReader(new FileReader(fileyear));
        File filesource = new File(Config.CAMINHO_SELECAO + "source.txt");
        BufferedReader bfsource = new BufferedReader(new FileReader(filesource));
        File fileauthors = new File(Config.CAMINHO_SELECAO + "authors.txt");
        BufferedReader bfauthors = new BufferedReader(new FileReader(fileauthors));
        File filekeywords = new File(Config.CAMINHO_SELECAO + "keywords.txt");
        BufferedReader bfkeywords = new BufferedReader(new FileReader(filekeywords));
        File filemonth = new File(Config.CAMINHO_SELECAO + "month.txt");
        BufferedReader bfmonth = new BufferedReader(new FileReader(filemonth));
        File fileabstract = new File(Config.CAMINHO_SELECAO + "abstract.txt");
        BufferedReader bfabstract = new BufferedReader(new FileReader(fileabstract));
        File filereadercount = new File(Config.CAMINHO_SELECAO + "reader_count.txt");
        BufferedReader bfreadercount = new BufferedReader(new FileReader(filereadercount));
        File idDocument = new File(Config.CAMINHO_SELECAO + "id.txt");
        BufferedReader bfIdDocument = new BufferedReader(new FileReader(idDocument));
        // Gerando os arquivos para limpeza
        File fileTitle1 = new File(Config.CAMINHO_REPETIDO + "title.txt");
        BufferedWriter bfTitle1 = new BufferedWriter(new FileWriter(fileTitle1));
        File filetype1 = new File(Config.CAMINHO_REPETIDO + "type.txt");
        BufferedWriter bfType1 = new BufferedWriter(new FileWriter(filetype1));
        File fileyear1 = new File(Config.CAMINHO_REPETIDO + "year.txt");
        BufferedWriter bfyear1 = new BufferedWriter(new FileWriter(fileyear1));
        File filesource1 = new File(Config.CAMINHO_REPETIDO + "source.txt");
        BufferedWriter bfsource1 = new BufferedWriter(new FileWriter(filesource1));
        File fileauthors1 = new File(Config.CAMINHO_REPETIDO + "authors.txt");
        BufferedWriter bfauthors1 = new BufferedWriter(new FileWriter(fileauthors1));
        File filekeywords1 = new File(Config.CAMINHO_REPETIDO + "keywords.txt");
        BufferedWriter bfkeywords1 = new BufferedWriter(new FileWriter(filekeywords1));
        File filemonth1 = new File(Config.CAMINHO_REPETIDO + "month.txt");
        BufferedWriter bfmonth1 = new BufferedWriter(new FileWriter(filemonth1));
        File fileabstract1 = new File(Config.CAMINHO_REPETIDO + "abstract.txt");
        BufferedWriter bfabstract1 = new BufferedWriter(new FileWriter(fileabstract1));
        File filereadercount1 = new File(Config.CAMINHO_REPETIDO + "reader_count.txt");
        BufferedWriter bfreadercount1 = new BufferedWriter(new FileWriter(filereadercount1));
        File idDocument1 = new File(Config.CAMINHO_REPETIDO + "id.txt");
```

```
BufferedWriter bfIdDocument1 = new BufferedWriter(new FileWriter(idDocument1));

File tratAuthors = new File(Config.CAMINHO_REPETIDO + "authorspreview.txt");
BufferedWriter tratBuff = new BufferedWriter(new FileWriter(tratAuthors));

System.out.println("Removendo artigos repetidos, aguarde...");
System.out.println("Isso pode levar um tempo dependendo da quantidade de dados");
String lerTitle = bfTitle.readLine();
while (lerTitle != null) {
   cout++;
   lerTitle = bfTitle.readLine();
}
System.out.println("Total de documentos em análise: " + cout);
bfTitle.close();
int n = 1;
String countAuthors2 = "";
System.out.println("Preparando authors. Isso pode demorar algum tempo");
String lerAuthors = bfauthors.readLine();
while (lerAuthors != null) {
   for (int i = 0; i < 3; i++) {
      if (i == 0) {
         countAuthors = countAuthors + lerAuthors + "#";
      } else if (i == 1) {
         countAuthors = countAuthors + lerAuthors + ",";
      } else if (i == 2) {
         if (n == Integer.parseInt(lerAuthors) - 1) {
            n++;
            tratBuff.append(countAuthors2 + "\r\n");
            countAuthors2 = "";
         }
         countAuthors = countAuthors + lerAuthors + "}";
         countAuthors2 = countAuthors2 + countAuthors;
         countAuthors = "";
      }
      lerAuthors = bfauthors.readLine();

   }
}
tratBuff.close();

BufferedReader tratReader = new BufferedReader(new FileReader(tratAuthors));

String arq1 = tratReader.readLine();
countAuthors = "";
while (arq1 != null) {
   String[] tokens = arq1.split("}");
   String[] tokens2;
   for (int i = 0; i < tokens.length; i++) {
      tokens2 = tokens[i].split(",");
      int comp = Integer.parseInt(tokens2[1]);
      if (comp == countAuthorsNum) {
         countAuthors = countAuthors + tokens2[0] + ",";
      } else if (comp == (countAuthorsNum + 1)) {
         dadosAuthors.add(countAuthors);
         countAuthors = "";
         countAuthors = countAuthors + tokens2[0] + ",";
         countAuthorsNum++;
      }
   }
   arq1 = tratReader.readLine();
```

```
}
dadosAuthors.add(countAuthors);
countAuthors = "";
String[] tokens = countAuthors2.split("}");
String[] tokens2;
for (int i = 0; i < tokens.length; i++) {
   tokens2 = tokens[i].split(",");
   countAuthors = countAuthors + tokens2[0] + ",";
}
dadosAuthors.add(countAuthors);
tratReader.close();

bfTitle = new BufferedReader(new FileReader(fileTitle));
lerTitle = bfTitle.readLine();
String lerType = bfType.readLine();
String lerYear = bfyear.readLine();
String lerSource = bfsource.readLine();
String lerKeywords = bfkeywords.readLine();
String lerMonth = bfmonth.readLine();
String lerAbstract = bfabstract.readLine();
String lerReaderCount = bfreadercount.readLine();
String lerIdDocument = bfIdDocument.readLine();
while (lerTitle != null) {

   dadosTitle.add(lerTitle);
   dadosType.add(lerType);
   dadosYear.add(Integer.parseInt(lerYear));
   dadosSource.add(lerSource);
   dadosKeywords.add(lerKeywords);
   dadosMonth.add(Integer.parseInt(lerMonth));
   dadosAbstract.add(lerAbstract);
   dadosReaderCount.add(lerReaderCount);
   dadosIdDocument.add(lerIdDocument);


   lerTitle = bfTitle.readLine();
   lerType = bfType.readLine();
   lerYear = bfyear.readLine();
   lerSource = bfsource.readLine();
   lerKeywords = bfkeywords.readLine();
   lerMonth = bfmonth.readLine();
   lerAbstract = bfabstract.readLine();
   lerReaderCount = bfreadercount.readLine();
   lerIdDocument = bfIdDocument.readLine();

}

System.out.println(dadosTitle.size());
System.out.println(dadosType.size());
System.out.println(dadosYear.size());
System.out.println(dadosSource.size());
System.out.println(dadosKeywords.size());
System.out.println(dadosMonth.size());
System.out.println(dadosAbstract.size());
System.out.println(dadosReaderCount.size());
System.out.println(dadosAuthors.size());
System.out.println(dadosIdDocument.size());
//Tratamento de remoção de artigos repetidos
System.out.println("Removendo documentos repetidos...");
for (int i = 0; i < dadosTitle.size(); i++) {
```

```
    lerTitle = dadosTitle.get(i);
    for (int j = 0; j < dadosTitle.size(); j++) {
        if (j == i) {
        } else if (lerTitle.equalsIgnoreCase(dadosTitle.get(j))) {
            dadosTitle.remove(j);
            dadosType.remove(j);
            dadosYear.remove(j);
            dadosSource.remove(j);
            dadosKeywords.remove(j);
            dadosMonth.remove(j);
            dadosAbstract.remove(j);
            dadosReaderCount.remove(j);
            dadosAuthors.remove(j);
            dadosIdDocument.remove(j);
            i--;
            break;
        }
    }
}
System.out.println(dadosTitle.size());
System.out.println(dadosType.size());
System.out.println(dadosYear.size());
System.out.println(dadosSource.size());
System.out.println(dadosKeywords.size());
System.out.println(dadosMonth.size());
System.out.println(dadosAbstract.size());
System.out.println(dadosReaderCount.size());
System.out.println(dadosAuthors.size());
System.out.println(dadosIdDocument.size());
System.out.println("Removendo meses irregulares");
for (int i = 0; i < dadosMonth.size(); i++) {
    if (dadosMonth.get(i) < 1 || dadosMonth.get(i) > 12) {
        System.out.println("Mês irregular: " + dadosMonth.get(i));
        dadosTitle.remove(i);
        dadosType.remove(i);
        dadosYear.remove(i);
        dadosSource.remove(i);
        dadosKeywords.remove(i);
        dadosMonth.remove(i);
        dadosAbstract.remove(i);
        dadosReaderCount.remove(i);
        dadosAuthors.remove(i);
        dadosIdDocument.remove(i);
        i--;
    }
}
System.out.println(dadosTitle.size());
System.out.println(dadosType.size());
System.out.println(dadosYear.size());
System.out.println(dadosSource.size());
System.out.println(dadosKeywords.size());
System.out.println(dadosMonth.size());
System.out.println(dadosAbstract.size());
System.out.println(dadosReaderCount.size());
System.out.println(dadosAuthors.size());
System.out.println(dadosIdDocument.size());
System.out.println("Removendo anos irregulares");
for (int i = 0; i < dadosYear.size(); i++) {
    if (dadosYear.get(i) > 2017 ) {
        System.out.println("Ano irregular: " + dadosYear.get(i));
```

```
            dadosTitle.remove(i);
            dadosType.remove(i);
            dadosYear.remove(i);
            dadosSource.remove(i);
            dadosKeywords.remove(i);
            dadosMonth.remove(i);
            dadosAbstract.remove(i);
            dadosReaderCount.remove(i);
            dadosAuthors.remove(i);
            dadosIdDocument.remove(i);
            i--;
        }
    }

    System.out.println(dadosTitle.size());
    System.out.println(dadosType.size());
    System.out.println(dadosYear.size());
    System.out.println(dadosSource.size());
    System.out.println(dadosKeywords.size());
    System.out.println(dadosMonth.size());
    System.out.println(dadosAbstract.size());
    System.out.println(dadosReaderCount.size());
    System.out.println(dadosAuthors.size());
    System.out.println(dadosIdDocument.size());

    System.out.println("Inserindo as alterações em um novo arquivo");
    for (int i = 0; i < dadosTitle.size(); i++) {

        if (!dadosTitle.get(i).equals("a")) {
            bfTitle1.append(dadosTitle.get(i) + "\r\n");
            bfType1.append(dadosType.get(i) + "\r\n");
            bfyear1.append(dadosYear.get(i) + "\r\n");
            bfsource1.append(dadosSource.get(i) + "\r\n");
            bfkeywords1.append(dadosKeywords.get(i) + "\r\n");
            bfmonth1.append(dadosMonth.get(i) + "\r\n");
            bfabstract1.append(dadosAbstract.get(i) + "\r\n");
            bfreadercount1.append(dadosReaderCount.get(i) + "\r\n");
            bfIdDocument1.append(dadosIdDocument.get(i) + "\r\n");

        }

    }
    countAuthorsNum = 0;
    for (int i = 0; i < dadosAuthors.size(); i++) {
        String[] tratarAuthors = dadosAuthors.get(i).split(",");
        countAuthorsArq++;
        for (int j = 0; j < tratarAuthors.length; j++) {
            String[] tratarAuthors1 = tratarAuthors[j].split("#");
            for (int k = 0; k < tratarAuthors1.length; k++) {
                if (countAuthorsNum == 0) {
                    bfauthors1.append(tratarAuthors1[k] + "\r\n");
                    countAuthorsNum++;
                } else if (countAuthorsNum == 1) {
                    bfauthors1.append(tratarAuthors1[k] + "\r\n");
                    bfauthors1.append(countAuthorsArq + "\r\n");
                    countAuthorsNum = 0;
                }
            }
        }
    }
```

```
        bfTitle1.close();
        bfType1.close();
        bfyear1.close();
        bfsource1.close();
        bfkeywords1.close();
        bfmonth1.close();
        bfabstract1.close();
        bfreadercount1.close();
        bfauthors1.close();
        bfIdDocument1.close();
        System.out.println("Processo de remoção concluido");
    }
}
```

Apêndice F – Algoritmo para limpeza dos documentos

```
public class LimparDocumento {

    public static void main(String[] args) throws FileNotFoundException, IOException {

        Scanner scan = new Scanner(System.in);

        System.out.println("Escolha uma opção:");
        System.out.println("1 - Limpar documntos da pasta destino");
        System.out.println("2 - Iniciar processo de limpeza");
        int op = scan.nextInt();

        if (op == 1) {
            File eTitle = new File(Config.CAMINHO_LIMPAR + "title.txt");
            File eType = new File(Config.CAMINHO_LIMPAR + "type.txt");
            File eSource = new File(Config.CAMINHO_LIMPAR + "source.txt");
            File eKeywords = new File(Config.CAMINHO_LIMPAR + "keywords.txt");
            File eAuthors = new File(Config.CAMINHO_LIMPAR + "authors.txt");
            File eAbstract = new File(Config.CAMINHO_LIMPAR + "abstract.txt");
            File eYear = new File(Config.CAMINHO_LIMPAR + "year.txt");
            File eMonth = new File(Config.CAMINHO_LIMPAR + "month.txt");
            File eReaderCount = new File(Config.CAMINHO_LIMPAR + "reader_count.txt");
            if (eTitle.exists()) {
                eTitle.delete();
                eType.delete();
                eSource.delete();
                eKeywords.delete();
                eAuthors.delete();
                eAbstract.delete();
                eYear.delete();
                eMonth.delete();
                eReaderCount.delete();
                System.out.println("Arquivos deletados com sucesso");
            } else {
                System.out.println("Não existem arquivos na pasta destino");
            }
        } else if (op == 2) {
            System.out.println("Limpando os arquivos");
            System.out.println("Dependendo da quantidade de documentos, isso pode levar um tempo...");
            Limpeza limpar = new Limpeza();
            File fTitle = new File(Config.CAMINHO_REPETIDO + "title.txt");
            BufferedReader titleBuff = new BufferedReader(new FileReader(fTitle));
            File fType = new File(Config.CAMINHO_REPETIDO + "type.txt");
            BufferedReader typeBuff = new BufferedReader(new FileReader(fType));
            File fSource = new File(Config.CAMINHO_REPETIDO + "source.txt");
            BufferedReader sourceBuff = new BufferedReader(new FileReader(fSource));
            File fYear = new File(Config.CAMINHO_REPETIDO + "year.txt");
            BufferedReader yearBuff = new BufferedReader(new FileReader(fYear));
            File fKeywords = new File(Config.CAMINHO_REPETIDO + "keywords.txt");
            BufferedReader keywordsBuff = new BufferedReader(new FileReader(fKeywords));
            File fAuthors = new File(Config.CAMINHO_REPETIDO + "authors.txt");
            BufferedReader authorsBuff = new BufferedReader(new FileReader(fAuthors));
            File fMonth = new File(Config.CAMINHO_REPETIDO + "month.txt");
            BufferedReader monthBuff = new BufferedReader(new FileReader(fMonth));
            File fReader_count = new File(Config.CAMINHO_REPETIDO + "reader_count.txt");
            BufferedReader reader_countBuff = new BufferedReader(new FileReader(fReader_count));
            File fAbstract = new File(Config.CAMINHO_REPETIDO + "abstract.txt");
            BufferedReader abstractBuff = new BufferedReader(new FileReader(fAbstract));
```

```
File eTitle = new File(Config.CAMINHO_LIMPAR + "title.txt");
BufferedWriter titleBf = new BufferedWriter(new FileWriter(eTitle));
File eType = new File(Config.CAMINHO_LIMPAR + "type.txt");
BufferedWriter typeBf = new BufferedWriter(new FileWriter(eType));
File eSource = new File(Config.CAMINHO_LIMPAR + "source.txt");
BufferedWriter sourceBf = new BufferedWriter(new FileWriter(eSource));
File eKeywords = new File(Config.CAMINHO_LIMPAR + "keywords.txt");
BufferedWriter keywordsBf = new BufferedWriter(new FileWriter(eKeywords));
File eAuthors = new File(Config.CAMINHO_LIMPAR + "authors.txt");
BufferedWriter authorsBf = new BufferedWriter(new FileWriter(eAuthors));
File eAbstract = new File(Config.CAMINHO_LIMPAR + "abstract.txt");
BufferedWriter abstractBf = new BufferedWriter(new FileWriter(eAbstract));
File eYear = new File(Config.CAMINHO_LIMPAR + "year.txt");
BufferedWriter yearBf = new BufferedWriter(new FileWriter(eYear));
File eMonth = new File(Config.CAMINHO_LIMPAR + "month.txt");
BufferedWriter monthBf = new BufferedWriter(new FileWriter(eMonth));
File eReaderCount = new File(Config.CAMINHO_LIMPAR + "reader_count.txt");
BufferedWriter readerCountBf = new BufferedWriter(new FileWriter(eReaderCount));

String arq = titleBuff.readLine();
int num = 0;
while (arq != null) {
   limpar.IniciarProcessoLimpeza(arq, "title");
   titleBf.append(limpar.arq + "\r\n");
   arq = titleBuff.readLine();
   num++;
}
titleBuff.close();
titleBf.close();
System.out.println(num);

arq = typeBuff.readLine();
num = 0;
while (arq != null) {

   limpar.IniciarProcessoLimpeza(arq, "type");
   typeBf.append(limpar.arq + "\r\n");
   arq = typeBuff.readLine();
   num++;
}
typeBuff.close();
typeBf.close();
System.out.println(num);
arq = sourceBuff.readLine();
num = 0;
while (arq != null) {

   limpar.IniciarProcessoLimpeza(arq, "source");
   sourceBf.append(limpar.arq + "\r\n");
   arq = sourceBuff.readLine();
   num++;
}
sourceBuff.close();
sourceBf.close();
System.out.println(num);
arq = keywordsBuff.readLine();
num = 0;
while (arq != null) {

   limpar.IniciarProcessoLimpeza(arq, "keywords");
```

```
    keywordsBf.append(limpar.arq + "\r\n");
    arq = keywordsBuff.readLine();
    num++;
}
keywordsBuff.close();
keywordsBf.close();
System.out.println(num);
arq = authorsBuff.readLine();
num = 0;
while (arq != null) {

    limpar.IniciarProcessoLimpeza(arq, "authors");
    authorsBf.append(limpar.arq + "\r\n");
    arq = authorsBuff.readLine();
    num++;
}
authorsBuff.close();
authorsBf.close();
System.out.println(num);
arq = abstractBuff.readLine();
num = 0;
while (arq != null) {

    limpar.IniciarProcessoLimpeza(arq, "abstract");
    abstractBf.append(limpar.arq + "\r\n");
    arq = abstractBuff.readLine();
    num++;
}
abstractBuff.close();
abstractBf.close();
System.out.println(num);
arq = yearBuff.readLine();
num = 0;
while (arq != null) {
    yearBf.append(arq + "\r\n");
    arq = yearBuff.readLine();
    num++;
}
yearBuff.close();
yearBf.close();
System.out.println(num);

arq = monthBuff.readLine();
num = 0;
while (arq != null) {
    monthBf.append(arq + "\r\n");
    arq = monthBuff.readLine();
    num++;
}
monthBuff.close();
monthBf.close();
System.out.println(num);

arq = reader_countBuff.readLine();
num = 0;
while (arq != null) {
    readerCountBf.append(arq + "\r\n");
    arq = reader_countBuff.readLine();
    num++;
}
```

```
            reader_countBuff.close();
            readerCountBf.close();
            System.out.println(num);

            System.out.println("Processo de limpeza concluido");
        }

    }
}
```

Apêndice G – Algoritmo de pré-adaptação dos documentos (visto no Capítulo 4)

```
public class AdaptarTitleResumoDefinitivo {

   public static void main(String[] args) throws FileNotFoundException, IOException {

      int ponteiroPalavraAlvo = 0, contadorPalavraAlvo = 0, controlador = 0, controlId = 0, menor = 0, count = 0,
documentos = 0, qtdDocumentos3 = 0, contadorOcorrencia = 0;
      String palavraAlvo, palavraCompleta = "", novaPalavra = "", pala = "";
      String[] palavras, palavrasComparar;

      File newTitle = new File(Config.CAMINHO_TITLE_ABSTRACT + "NewTitle.txt");
      BufferedWriter bfNewTitle = new BufferedWriter(new FileWriter(newTitle));

      File newAbstract = new File(Config.CAMINHO_TITLE_ABSTRACT + "NewAbstract.txt");
      BufferedWriter bfNewAbstract = new BufferedWriter(new FileWriter(newAbstract));

      File newKeywords = new File(Config.CAMINHO_TITLE_ABSTRACT + "newKeywords.txt");
      BufferedWriter bfNewKeywords = new BufferedWriter(new FileWriter(newKeywords));

      File newAuthors = new File(Config.CAMINHO_TITLE_ABSTRACT + "newAuthors.txt");
      BufferedWriter bfNewAuthors = new BufferedWriter(new FileWriter(newAuthors));

      File keywordsE1 = new File(Config.CAMINHO_ADAPTADO + "keywords_num.txt");
      BufferedWriter bfKeywordsE1 = new BufferedWriter(new FileWriter(keywordsE1));

      File fTitle = new File(Config.CAMINHO_LIMPAR + "title.txt");
      BufferedReader titleBuff = new BufferedReader(new FileReader(fTitle));

      File fAbstract = new File(Config.CAMINHO_LIMPAR + "abstract.txt");
      BufferedReader abstractBuff = new BufferedReader(new FileReader(fAbstract));

      File fKeywords = new File(Config.CAMINHO_LIMPAR + "keywords.txt");
      BufferedReader keywordsBuff = new BufferedReader(new FileReader(fKeywords));

      File fAuthors = new File(Config.CAMINHO_LIMPAR + "authors.txt");
      BufferedReader authorsBuff = new BufferedReader(new FileReader(fAuthors));

      File oTitle = new File(Config.CAMINHO_TITLE_ABSTRACT + "Title.txt");
      BufferedWriter oTitleBuff = new BufferedWriter(new FileWriter(oTitle));
      File oAbstract = new File(Config.CAMINHO_TITLE_ABSTRACT + "Abstract.txt");
      BufferedWriter oAbstractBuff = new BufferedWriter(new FileWriter(oAbstract));

      String termo = "", newNovaPalavra = "";
      ArrayList<String> dadosTitle = new ArrayList();
      ArrayList<String> dadosTitleContados = new ArrayList();
      ArrayList<String> dadosTitleNew = new ArrayList();
      ArrayList<String> dadosTitleArq = new ArrayList();

      ArrayList<String> dadosAbstract = new ArrayList();
      ArrayList<String> dadosAbstractContados = new ArrayList();
      ArrayList<String> dadosAbstractNew = new ArrayList();
      ArrayList<String> dadosAbstractArq = new ArrayList();

      ArrayList<String> dadosKeywords = new ArrayList();
      ArrayList<String> dadosKeywordsNew = new ArrayList();
      ArrayList<String> dadosKeywordsContados = new ArrayList();
      ArrayList<String> dadosKeywordsArq = new ArrayList();

      ArrayList<String> dadosAuthors = new ArrayList();
```

```
ArrayList<String> dadosAuthorsContados = new ArrayList();
ArrayList<String> dadosAuthorsContadosPorId = new ArrayList();

System.out.println("|__ Adaptando title:");
String arq = titleBuff.readLine();
while (arq != null) {
   dadosTitle.add(arq);
   arq = titleBuff.readLine();
}

for (int i = 0; i < dadosTitle.size(); i++) {
   dadosTitleNew.add(String.valueOf("0"));
   dadosTitleArq.add(String.valueOf(i));
}

int qtdDocumentos = 0;
System.out.println(dadosTitle.size() + " documentos a serem processados, aguarde...");
while (qtdDocumentos < dadosTitle.size()) {
   if (qtdDocumentos % 1000 == 0) {
      System.out.println(qtdDocumentos + " documentos processados");
   }

   palavras = dadosTitle.get(qtdDocumentos).split(" ");
   count = 0;
   controlador = 0;
   while (ponteiroPalavraAlvo < palavras.length) {

      palavraAlvo = palavras[ponteiroPalavraAlvo];
      for (int i = 0; i < dadosTitle.size(); i++) {
         palavrasComparar = dadosTitle.get(i).split(" ");
         for (int j = 0; j < palavrasComparar.length; j++) {
            if (!"".equals(palavrasComparar[j])) {
               if (palavraAlvo.equals(palavrasComparar[j])) {
                  dadosTitleNew.set(i, "1");
                  contadorPalavraAlvo++;
               } else {
               }
            }
         }
      }

      palavraAlvo = palavraAlvo + "_" + contadorPalavraAlvo;
      termo = termo + palavraAlvo + " ";

      dadosTitleContados.add(palavraAlvo);
      contadorPalavraAlvo = 0;
      ponteiroPalavraAlvo++;
      for (int i = 0; i < dadosTitleNew.size(); i++) {
         if (dadosTitleNew.get(i).equals("1")) {
            pala = pala + i + ",";
            dadosTitleNew.set(i, "0");
         }
      }
      String[] indice = pala.split(",");
      String[] comparar = palavraAlvo.split("_");

      for (int i = 0; i < indice.length; i++) {
         if (!"0".equals(comparar[1])) {
            String[] palavraArq = dadosTitle.get(Integer.parseInt(indice[i])).split(" ");
            for (int j = 0; j < palavraArq.length; j++) {
```

```
                if (!"".equals(palavraArq)) {
                    if (palavraArq[j].equals(comparar[0])) {
                        palavraArq[j] = "";
                    }
                    if (!"".equals(palavraArq[j])) {
                        novaPalavra = novaPalavra + palavraArq[j] + " ";
                    }

                }
            }

            dadosTitle.set(Integer.parseInt(indice[i]), novaPalavra);

            novaPalavra = "";
        }
    }
    for (int i = 0; i < indice.length; i++) {
        if (!"".equals(indice[i])) {
        if (dadosTitleArq.get(Integer.parseInt(indice[i])).equals(indice[i])) {
            dadosTitleArq.set(Integer.parseInt(indice[i]), palavraAlvo);
        }
        else {
            String valor = dadosTitleArq.get(Integer.parseInt(indice[i]));
            valor = valor + " " + palavraAlvo;
            dadosTitleArq.set(Integer.parseInt(indice[i]), valor);
        }
    }

    }
    pala = "";
  }
  termo = "";
  ponteiroPalavraAlvo = 0;
  qtdDocumentos++;
  controlador = 0;
}
for (int i = 0; i < dadosTitleArq.size(); i++) {
  bfNewTitle.append(dadosTitleArq.get(i) + "\r\n");
}

bfNewTitle.close();

qtdDocumentos = 0;
termo = "";
int qtdDocumentos2 = dadosTitleContados.size();
System.out.println("Iniciando processo de ordenação de dados. Total de palavras: " + qtdDocumentos2);
for (int i = 0; i < dadosTitleContados.size(); i++) {
  if (qtdDocumentos % 5000 == 0 && qtdDocumentos != 0) {
    System.out.println(qtdDocumentos + " palavras ordenadas");
  }

  String[] palavraAComparar = dadosTitleContados.get(i).split("_");
  int num = Integer.parseInt(palavraAComparar[1]);
  controlador = 0;
  for (int j = i; j < dadosTitleContados.size(); j++) {
    String[] palavraAComparar1 = dadosTitleContados.get(j).split("_");
    menor = Integer.parseInt(palavraAComparar1[1]);
    if (num < menor) {
      controlador = 1;
      num = menor;
```

```
            termo = palavraAComparar1[0] + "_" + palavraAComparar1[1];
            controlId = j;
        }
    }
    if (controlador == 1) {
        dadosTitleContados.set(i, termo);
        dadosTitleContados.set(controlId, palavraAComparar[0] + "_" + palavraAComparar[1]);
    }
    qtdDocumentos++;

}

for (int i = 0; i < dadosTitleContados.size(); i++) {
    oTitleBuff.append(dadosTitleContados.get(i) + "\r\n");
}
oTitleBuff.close();
dadosTitle = null;
dadosTitleContados = null;
dadosTitleNew = null;
System.out.println("Processo concluido");

System.out.println("");
System.out.println("|__ Adaptando Authors:");
String idAuthors = "";
arq = authorsBuff.readLine();

contadorOcorrencia = 0;
palavraCompleta = "";
while (arq != null) {
    if (contadorOcorrencia == 0) {
        palavraCompleta = palavraCompleta + arq;
    } else if (contadorOcorrencia == 1) {
        palavraCompleta = palavraCompleta + " " + arq;
    } else if (contadorOcorrencia == 2) {
        palavraCompleta = palavraCompleta + ":" + arq;
        dadosAuthors.add(palavraCompleta);
        dadosAuthorsContados.add("0");
    }
    contadorOcorrencia++;
    if (contadorOcorrencia > 2) {
        contadorOcorrencia = 0;
        palavraCompleta = "";
    }
    arq = authorsBuff.readLine();
}
System.out.println(dadosAuthors.size());
qtdDocumentos = 0;
palavraAlvo = "";
System.out.println("Realizando a contagem de autores. Total: " + dadosAuthors.size());
while (qtdDocumentos < dadosAuthors.size()) {
    if (qtdDocumentos % 5000 == 0) {
        System.out.println(qtdDocumentos + " autores contados");
    }
        if (dadosAuthorsContados.get(qtdDocumentos).equals("0")) {
            palavras = dadosAuthors.get(qtdDocumentos).split(":");
        for (int j = 0; j < dadosAuthors.size(); j++) {
            if (dadosAuthorsContados.get(j).equals("0")) {
                String[] palavras1 = dadosAuthors.get(j).split(":");
                if (palavras[0].equals(palavras1[0]) && palavras[1].equals(palavras1[1])) {
```

```
                idAuthors = idAuthors + j + ",";

                contadorPalavraAlvo++;
            }
        }

    }
    String[] forPalavras = idAuthors.split(",");
    for (int z = 0; z < forPalavras.length; z++) {
        palavraAlvo = dadosAuthors.get(Integer.parseInt(forPalavras[z]));
        palavraAlvo = palavraAlvo + "_" + contadorPalavraAlvo;
        dadosAuthorsContados.set(Integer.parseInt(forPalavras[z]), palavraAlvo);

    }
    qtdDocumentos++;
    contadorPalavraAlvo = 0;
    palavraAlvo = "";
    idAuthors = "";
    } else {
      qtdDocumentos++;
    }
}

System.out.println("Reagrupando autores por id. Isso levará algum tempo. Total: " + dadosTitleArq.size());
qtdDocumentos = 1;
contadorPalavraAlvo = 0;
while (qtdDocumentos <= dadosTitleArq.size()) {
  for (int i = 0; i < dadosAuthorsContados.size(); i++) {
    palavras = dadosAuthorsContados.get(i).split(":");
    String[] palavras1 = palavras[1].split("_");
    if (Integer.parseInt(palavras1[0]) == qtdDocumentos) {

      contadorPalavraAlvo = contadorPalavraAlvo + Integer.parseInt(palavras1[1]);
    }
  }
  qtdDocumentos++;
  if (qtdDocumentos % 5000 == 0) {
    System.out.println(qtdDocumentos + " autores reagrupados");
  }
  dadosAuthorsContadosPorId.add(String.valueOf(contadorPalavraAlvo));
  contadorPalavraAlvo = 0;
}

System.out.println(dadosAuthorsContadosPorId.size());

for (int i = 0; i < dadosAuthorsContadosPorId.size(); i++) {
  bfNewAuthors.append(dadosAuthorsContadosPorId.get(i) + "\r\n");
}

bfNewAuthors.close();
dadosTitleArq = null;
dadosAuthorsContadosPorId = null;
dadosAuthors = null;
dadosAuthorsContados = null;
System.out.println("Concluido");

System.out.println("|__ Adaptando abstract:");
arq = abstractBuff.readLine();
while (arq != null) {
  dadosAbstract.add(arq);
```

```
   arq = abstractBuff.readLine();
}

for (int i = 0; i < dadosAbstract.size(); i++) {
   dadosAbstractNew.add(String.valueOf("0"));
   dadosAbstractArq.add(String.valueOf(i));
}

qtdDocumentos = 0;
System.out.println(dadosAbstract.size() + " documentos a serem processados, aguarde...");
while (qtdDocumentos < dadosAbstract.size()) {
   if (qtdDocumentos % 100 == 0) {
      System.out.println(qtdDocumentos + " documentos processados");
   }

   palavras = dadosAbstract.get(qtdDocumentos).split(" ");
   count = 0;
   controlador = 0;
   while (ponteiroPalavraAlvo < palavras.length) {

      palavraAlvo = palavras[ponteiroPalavraAlvo];
      for (int i = 0; i < dadosAbstract.size(); i++) {
         palavrasComparar = dadosAbstract.get(i).split(" ");
         for (int j = 0; j < palavrasComparar.length; j++) {
            if (!"".equals(palavrasComparar[j])) {
               if (palavraAlvo.equals(palavrasComparar[j])) {
                  dadosAbstractNew.set(i, "1");
                  contadorPalavraAlvo++;
               } else {
               }
            }
         }
      }
      palavraAlvo = palavraAlvo + "_" + contadorPalavraAlvo;
      termo = termo + palavraAlvo + " ";
      dadosAbstractContados.add(palavraAlvo);
      contadorPalavraAlvo = 0;
      ponteiroPalavraAlvo++;
      for (int i = 0; i < dadosAbstractNew.size(); i++) {
         if (dadosAbstractNew.get(i).equals("1")) {
            pala = pala + i + ",";
            dadosAbstractNew.set(i, "0");
         }
      }
      String[] indice = pala.split(",");
      String[] comparar = palavraAlvo.split("_");

      for (int i = 0; i < indice.length; i++) {
         if (!"0".equals(comparar[1])) {
            String[] palavraArq = dadosAbstract.get(Integer.parseInt(indice[i])).split(" ");
            for (int j = 0; j < palavraArq.length; j++) {
               if (!"".equals(palavraArq)) {
                  if (palavraArq[j].equals(comparar[0])) {
                     palavraArq[j] = "";
                  }
                  if (!"".equals(palavraArq[j])) {
                     novaPalavra = novaPalavra + palavraArq[j] + " ";
                  }

               }
```

```
                }
                dadosAbstract.set(Integer.parseInt(indice[i]), novaPalavra);

                novaPalavra = "";
            }
        }
        for (int i = 0; i < indice.length; i++) {
            if (!"".equals(indice[i])) {
            if (dadosAbstractArq.get(Integer.parseInt(indice[i])).equals(indice[i])) {
                dadosAbstractArq.set(Integer.parseInt(indice[i]), palavraAlvo);
            }
            else {
                String valor = dadosAbstractArq.get(Integer.parseInt(indice[i]));
                valor = valor + " " + palavraAlvo;
                dadosAbstractArq.set(Integer.parseInt(indice[i]), valor);
            }
        }

        }

        pala = "";
    }
    termo = "";
    ponteiroPalavraAlvo = 0;
    qtdDocumentos++;
    controlador = 0;
}
for (int i = 0; i < dadosAbstractArq.size(); i++) {
    bfNewAbstract.append(dadosAbstractArq.get(i) + "\r\n");
}
bfNewAbstract.close();

qtdDocumentos = 0;
qtdDocumentos2 = dadosAbstractContados.size();
termo = "";
System.out.println("Iniciando processo de ordenação de dados. Total de palavras: " + qtdDocumentos2);
for (int i = 0; i < dadosAbstractContados.size(); i++) {
    controlador = 0;
    if (qtdDocumentos % 5000 == 0 && qtdDocumentos != 0) {
        System.out.println(qtdDocumentos + " palavras ordenadas");
    }
    String[] palavraAComparar = dadosAbstractContados.get(i).split("_");
    int num = Integer.parseInt(palavraAComparar[1]);

    for (int j = i; j < dadosAbstractContados.size(); j++) {
        String[] palavraAComparar1 = dadosAbstractContados.get(j).split("_");
        menor = Integer.parseInt(palavraAComparar1[1]);
        if (num < menor) {
            controlador = 1;
            num = menor;
            termo = palavraAComparar1[0] + "_" + palavraAComparar1[1];
            controlId = j;
        }
    }
    if (controlador == 1) {
        dadosAbstractContados.set(i, termo);
        dadosAbstractContados.set(controlId, palavraAComparar[0] + "_" + palavraAComparar[1]);
    }
    qtdDocumentos++;
}
```

```
for (int i = 0; i < dadosAbstractContados.size(); i++) {
   oAbstractBuff.append(dadosAbstractContados.get(i) + "\r\n");
}
oAbstractBuff.close();
dadosAbstract = null;
dadosAbstractContados = null;
dadosAbstractArq = null;
dadosAbstractNew = null;
System.out.println("Processo concluido");

System.out.println("|__ Adaptando keywords:");

arq = keywordsBuff.readLine();
while (arq != null) {
   dadosKeywords.add(arq);
   arq = keywordsBuff.readLine();
}

for (int i = 0; i < dadosKeywords.size(); i++) {
   dadosKeywordsNew.add(String.valueOf("0"));
   dadosKeywordsArq.add(String.valueOf(i));
}

qtdDocumentos = 0;
System.out.println(dadosKeywords.size() + " documentos a serem processados, aguarde...");
while (qtdDocumentos < dadosKeywords.size()) {
   if (qtdDocumentos % 1000 == 0) {
      System.out.println(qtdDocumentos + " documentos processados");
   }

   palavras = dadosKeywords.get(qtdDocumentos).split(",");
   count = 0;
   controlador = 0;
   while (ponteiroPalavraAlvo < palavras.length) {

      palavraAlvo = palavras[ponteiroPalavraAlvo];
      for (int i = 0; i < dadosKeywords.size(); i++) {
         palavrasComparar = dadosKeywords.get(i).split(",");
         for (int j = 0; j < palavrasComparar.length; j++) {
            if (!"".equals(palavrasComparar[j])) {
               if (palavraAlvo.equals(palavrasComparar[j])) {
                  dadosKeywordsNew.set(i, "1");
                  contadorPalavraAlvo++;
               } else {
               }
            }
         }
      }

      palavraAlvo = palavraAlvo + "_" + contadorPalavraAlvo;
      termo = termo + palavraAlvo + ",";

      dadosKeywordsContados.add(palavraAlvo);
      contadorPalavraAlvo = 0;
      ponteiroPalavraAlvo++;
      for (int i = 0; i < dadosKeywordsNew.size(); i++) {
         if (dadosKeywordsNew.get(i).equals("1")) {
            pala = pala + i + ",";
            dadosKeywordsNew.set(i, "0");
```

```
            }
        }
        String[] indice = pala.split(",");
        String[] comparar = palavraAlvo.split("_");

        for (int i = 0; i < indice.length; i++) {
            if (!"0".equals(comparar[1])) {
                String[] palavraArq = dadosKeywords.get(Integer.parseInt(indice[i])).split(",");
                for (int j = 0; j < palavraArq.length; j++) {
                    if (!"".equals(palavraArq)) {
                        if (palavraArq[j].equals(comparar[0])) {
                            palavraArq[j] = "";
                        }
                        if (!"".equals(palavraArq[j])) {
                            novaPalavra = novaPalavra + palavraArq[j] + ",";
                        }

                    }
                }


                dadosKeywords.set(Integer.parseInt(indice[i]), novaPalavra);

                novaPalavra = "";
            }
        }
        for (int i = 0; i < indice.length; i++) {
            if (!"".equals(indice[i])) {
            if (dadosKeywordsArq.get(Integer.parseInt(indice[i])).equals(indice[i])) {
                dadosKeywordsArq.set(Integer.parseInt(indice[i]), palavraAlvo);
            }
            else {
                String valor = dadosKeywordsArq.get(Integer.parseInt(indice[i]));
                valor = valor + "," + palavraAlvo;
                dadosKeywordsArq.set(Integer.parseInt(indice[i]), valor);
            }
        }
        }
        pala = "";
    }
    termo = "";
    ponteiroPalavraAlvo = 0;
    qtdDocumentos++;
    controlador = 0;
}
for (int i = 0; i < dadosKeywordsArq.size(); i++) {
    bfNewKeywords.append(dadosKeywordsArq.get(i) + "\r\n");
}
bfNewKeywords.close();
dadosKeywords = null;
newKeywords = new File(Config.CAMINHO_TITLE_ABSTRACT + "newKeywords.txt");
BufferedReader bfNewKeywords1 = new BufferedReader(new FileReader(newKeywords));
arq = bfNewKeywords1.readLine();

while (arq != null) {
    dadosKeywords.add(arq);
    arq = bfNewKeywords1.readLine();
}
System.out.println(dadosKeywords.size());
qtdDocumentos = 0;
```

```
        ponteiroPalavraAlvo = 0;
        contadorPalavraAlvo = 0;
        palavraCompleta = "";

        contadorOcorrencia = 0;
        for (int i = 0; i < dadosKeywords.size(); i++) {
           palavras = dadosKeywords.get(i).split(",");
           for (int j = 0; j < palavras.length; j++) {
              String[] palavras1 = palavras[j].split("_");

              if (palavras1.length > 1) {
                 contadorOcorrencia = contadorOcorrencia + Integer.parseInt(palavras1[1]);
              }
           }
           bfKeywordsE1.append(contadorOcorrencia + "\r\n");
           contadorOcorrencia = 0;

        }
        bfKeywordsE1.close();
        System.out.println("Concluído");

        System.out.println("Processo Concluido");
     }
}
```

Apêndice H – Algoritmo de adaptação final (visto no Capítulo 4)

```
public class AdaptarDocumento5 {

   public static void main(String[] args) throws IOException {

      int contadorOcorrencia = 0, ponteiroPalavraAlvo = 0;
      String palavraAlvo, palavraCompleta = "", palavraType, palavraCompletaType = "",
palavraCompletaSource = "";
      int contadorPalavraAlvo = 0, maior = 0, menor = 0, controlador = 0;
      String[] palavras, palavrasComparar;
      Discretizar d = new Discretizar();

      File newTitle = new File(Config.CAMINHO_TITLE_ABSTRACT + "NewTitle.txt");
      BufferedReader bfNewTitle = new BufferedReader(new FileReader(newTitle));
      File newAbstract = new File(Config.CAMINHO_TITLE_ABSTRACT + "NewAbstract.txt");
      BufferedReader bfNewAbstract = new BufferedReader(new FileReader(newAbstract));
      File newKeywords = new File(Config.CAMINHO_TITLE_ABSTRACT + "NewKeywords.txt");
      BufferedReader bfNewKeywords = new BufferedReader(new FileReader(newKeywords));
      File newAuthors = new File(Config.CAMINHO_TITLE_ABSTRACT + "newAuthors.txt");
      BufferedReader bfNewAuthors = new BufferedReader(new FileReader(newAuthors));

      File fType = new File(Config.CAMINHO_LIMPAR + "type.txt");
      BufferedReader typeBuff = new BufferedReader(new FileReader(fType));
      File fSource = new File(Config.CAMINHO_LIMPAR + "source.txt");
      BufferedReader sourceBuff = new BufferedReader(new FileReader(fSource));
      File fYear = new File(Config.CAMINHO_LIMPAR + "year.txt");
      BufferedReader yearBuff = new BufferedReader(new FileReader(fYear));
      File fMonth = new File(Config.CAMINHO_LIMPAR + "month.txt");
      BufferedReader monthBuff = new BufferedReader(new FileReader(fMonth));
      File fReader_count = new File(Config.CAMINHO_LIMPAR + "reader_count.txt");
      BufferedReader reader_countBuff = new BufferedReader(new FileReader(fReader_count));

      File fTitleFive = new File(Config.CAMINHO_ADAPTADO + "title5.txt");
      BufferedWriter bfTitleFive = new BufferedWriter(new FileWriter(fTitleFive));
      File fTitleTree = new File(Config.CAMINHO_ADAPTADO + "title3.txt");
      BufferedWriter bfTitleTree = new BufferedWriter(new FileWriter(fTitleTree));
      File fTitleTwo = new File(Config.CAMINHO_ADAPTADO + "title2.txt");
      BufferedWriter bfTitleTwo = new BufferedWriter(new FileWriter(fTitleTwo));

      File fKeywordsFive = new File(Config.CAMINHO_ADAPTADO + "keywords5.txt");
      BufferedWriter bfKeywordsFive = new BufferedWriter(new FileWriter(fKeywordsFive));
      File fKeywordsTree = new File(Config.CAMINHO_ADAPTADO + "keywords3.txt");
      BufferedWriter bfKeywordsTree = new BufferedWriter(new FileWriter(fKeywordsTree));
      File fKeywordsTwo = new File(Config.CAMINHO_ADAPTADO + "keywords2.txt");
      BufferedWriter bfKeywordsTwo = new BufferedWriter(new FileWriter(fKeywordsTwo));

      File fAuthorsFive = new File(Config.CAMINHO_ADAPTADO + "authors5.txt");
      BufferedWriter bfAuthorsFive = new BufferedWriter(new FileWriter(fAuthorsFive));
      File fAuthorsTree = new File(Config.CAMINHO_ADAPTADO + "authors3.txt");
      BufferedWriter bfAuthorsTree = new BufferedWriter(new FileWriter(fAuthorsTree));
      File fAuthorsTwo = new File(Config.CAMINHO_ADAPTADO + "authors2.txt");
      BufferedWriter bfAuthorsTwo = new BufferedWriter(new FileWriter(fAuthorsTwo));

      File fAbstractFive = new File(Config.CAMINHO_ADAPTADO + "abstract5.txt");
      BufferedWriter bfAbstractFive = new BufferedWriter(new FileWriter(fAbstractFive));
      File fAbstractTree = new File(Config.CAMINHO_ADAPTADO + "abstract3.txt");
      BufferedWriter bfAbstractTree = new BufferedWriter(new FileWriter(fAbstractTree));
      File fAbstractTwo = new File(Config.CAMINHO_ADAPTADO + "abstract2.txt");
      BufferedWriter bfAbstractTwo = new BufferedWriter(new FileWriter(fAbstractTwo));
```

```
        File fReaderCountFive = new File(Config.CAMINHO_ADAPTADO + "reader5.txt");
        BufferedWriter bfReaderCountFive = new BufferedWriter(new FileWriter(fReaderCountFive));
        File fReaderCountTree = new File(Config.CAMINHO_ADAPTADO + "reader3.txt");
        BufferedWriter bfReaderCountTree = new BufferedWriter(new FileWriter(fReaderCountTree));
        File fReaderCountTwo = new File(Config.CAMINHO_ADAPTADO + "reader2.txt");
        BufferedWriter bfReaderCountTwo = new BufferedWriter(new FileWriter(fReaderCountTwo));

        File titleE1 = new File(Config.CAMINHO_ADAPTADO + "title_num.txt");
        BufferedWriter bfTitleE1 = new BufferedWriter(new FileWriter(titleE1));

        File abstractE1 = new File(Config.CAMINHO_ADAPTADO + "abstract_num.txt");
        BufferedWriter bfAbstractE1 = new BufferedWriter(new FileWriter(abstractE1));

        File data5 = new File(Config.CAMINHO_ADAPTADO + Config.FILE_NAME_DATA_SET5 + ".data");
        BufferedWriter dataBf5 = new BufferedWriter(new FileWriter(data5));
        File names5 = new File(Config.CAMINHO_ADAPTADO + Config.FILE_NAME_DATA_SET5 +
".names");
        BufferedWriter namesBf5 = new BufferedWriter(new FileWriter(names5));
        File data3 = new File(Config.CAMINHO_ADAPTADO + Config.FILE_NAME_DATA_SET3 + ".data");
        BufferedWriter dataBf3 = new BufferedWriter(new FileWriter(data3));
        File names3 = new File(Config.CAMINHO_ADAPTADO + Config.FILE_NAME_DATA_SET3 +
".names");
        BufferedWriter namesBf3 = new BufferedWriter(new FileWriter(names3));
        File data2 = new File(Config.CAMINHO_ADAPTADO + Config.FILE_NAME_DATA_SET2 + ".data");
        BufferedWriter dataBf2 = new BufferedWriter(new FileWriter(data2));
        File names2 = new File(Config.CAMINHO_ADAPTADO + Config.FILE_NAME_DATA_SET2 +
".names");
        BufferedWriter namesBf2 = new BufferedWriter(new FileWriter(names2));

        File titleTitle = new File(Config.CAMINHO_TITLE_ABSTRACT + "Title.txt");
        BufferedReader bfTitleTitle = new BufferedReader(new FileReader(titleTitle));

        File absAbstract = new File(Config.CAMINHO_TITLE_ABSTRACT + "Abstract.txt");
        BufferedReader bfAbsAbstract = new BufferedReader(new FileReader(absAbstract));

        ArrayList<String> dadosTitle = new ArrayList();
        ArrayList<String> dadosTitleContados = new ArrayList();
        ArrayList<String> dadosTitleNum = new ArrayList();
        ArrayList<String> dadosTitleDiscretizados = new ArrayList();
        ArrayList<String> dadosTitleSeparados = new ArrayList();

        ArrayList<String> dadosType = new ArrayList();
        ArrayList<String> dadosTypeContados = new ArrayList();
        ArrayList<String> dadosTypeNum = new ArrayList();
        ArrayList<String> dadosTypeContadosNames = new ArrayList();

        ArrayList<String> dadosSource = new ArrayList();
        ArrayList<String> dadosSourceContados = new ArrayList();
        ArrayList<String> dadosSourceNum = new ArrayList();
        ArrayList<String> dadosSourceContadosNames = new ArrayList();

        ArrayList<String> dadosYear = new ArrayList();
        ArrayList<String> dadosYearDiscretizados = new ArrayList();

        ArrayList<String> dadosKeywords = new ArrayList();
        ArrayList<String> dadosKeywordsNum = new ArrayList();
        ArrayList<String> dadosKeywordsDiscretizados = new ArrayList();

        ArrayList<String> dadosMonth = new ArrayList();
```

```
ArrayList<String> dadosMonthName = new ArrayList();
int[] month = new int[12];

ArrayList<String> dadosAbstract = new ArrayList();
ArrayList<String> dadosAbstractContados = new ArrayList();
ArrayList<String> dadosAbstractNum = new ArrayList();
ArrayList<String> dadosAbstractDiscretizados = new ArrayList();

ArrayList<String> dadosReaderCount = new ArrayList();
ArrayList<String> dadosReaderCountDiscretizados = new ArrayList();

ArrayList<String> dadosAuthors = new ArrayList();
ArrayList<String> dadosAuthorsContados = new ArrayList();
ArrayList<String> dadosAuthorsContadosPorId = new ArrayList();
ArrayList<String> dadosAuthorsDiscretizados = new ArrayList();

String[] nome;

System.out.println("Adaptação para até 5 classes de saída");
System.out.println("|__ Adaptando title:");
String arq = bfNewTitle.readLine();

while (arq != null) {
   dadosTitle.add(arq);
   arq = bfNewTitle.readLine();
}
int qtdDocumentos = 0;
System.out.println("Iniciando o processo de title. Total: " + dadosTitle.size());

for (int i = 0; i < dadosTitle.size(); i++) {
   if (i % 1000 == 0) {
      System.out.println(i + " documentos processados");
   }
   palavras = dadosTitle.get(i).split(" ");
   for (int j = 0; j < palavras.length; j++) {
      String[] palavras1 = palavras[j].split("_");
      if (Integer.parseInt(palavras1[1]) <= Config.QTD_FREQUENCIA_TITLE) {
         titleTitle = new File(Config.CAMINHO_TITLE_ABSTRACT + "Title.txt");
         bfTitleTitle = new BufferedReader(new FileReader(titleTitle));
         arq = bfTitleTitle.readLine();
         String[] palavraContada = arq.split("_");
         while (arq != null) {
            if (arq.equals(palavras[j])) {
               contadorOcorrencia = contadorOcorrencia + Integer.parseInt(palavraContada[1]);
               arq = null;
            } else {
               arq = bfTitleTitle.readLine();
            }

            if (arq != null) {
               palavraContada = arq.split("_");
            }
         }
         bfTitleTitle.close();
      }
   }
   dadosTitleNum.add(String.valueOf(contadorOcorrencia));
   contadorOcorrencia = 0;
}
```

```
        for (int i = 0; i < dadosTitleNum.size(); i++) {
            int num = Integer.parseInt(dadosTitleNum.get(i));
            int num2 = Integer.parseInt(dadosTitleNum.get(i));
            for (int j = 0; j < dadosTitleNum.size(); j++) {
                if (num > Integer.parseInt(dadosTitleNum.get(j))) {
                    maior = num;
                } else if (num < Integer.parseInt(dadosTitleNum.get(j))) {
                    num = Integer.parseInt(dadosTitleNum.get(j));
                    maior = num;
                }
                if (num2 > Integer.parseInt(dadosTitleNum.get(j))) {
                    num2 = Integer.parseInt(dadosTitleNum.get(j));
                    menor = num2;
                } else if (num2 < Integer.parseInt(dadosTitleNum.get(j))) {
                    menor = num2;
                }
            }
        }

        d.Discretizacao5(maior, menor);
        nome = Config.VALUES5.split(",");
        for (int i = 0; i < dadosTitleNum.size(); i++) {
            if (Double.parseDouble(dadosTitleNum.get(i)) >= Math.floor(menor)
                    && Double.parseDouble(dadosTitleNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
                dadosTitleDiscretizados.add(nome[0]);
            } else if (Double.parseDouble(dadosTitleNum.get(i)) >= (Math.floor(menor) + d.discretizar)
                    && Double.parseDouble(dadosTitleNum.get(i)) < ((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) {
                dadosTitleDiscretizados.add(nome[1]);
            } else if (Double.parseDouble(dadosTitleNum.get(i)) >= ((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)
                    && Double.parseDouble(dadosTitleNum.get(i)) < (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar) {
                dadosTitleDiscretizados.add(nome[2]);
            } else if (Double.parseDouble(dadosTitleNum.get(i)) >= (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar
                    && Double.parseDouble(dadosTitleNum.get(i)) < (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar * 2.0f) {
                dadosTitleDiscretizados.add(nome[3]);
            } else if (Double.parseDouble(dadosTitleNum.get(i)) >= (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar * 2.0f
                    && Double.parseDouble(dadosTitleNum.get(i)) <= (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar * 3.0f) {
                dadosTitleDiscretizados.add(nome[4]);
            }
        }
        for (int i = 0; i < dadosTitleDiscretizados.size(); i++) {
            bfTitleFive.append(dadosTitleDiscretizados.get(i) + "\r\n");
        }
        bfTitleFive.close();
        System.out.println("dadosTitleDiscretizados: " + dadosTitleDiscretizados.size());
        dadosTitleDiscretizados = null;
        dadosTitleDiscretizados = new ArrayList();
        System.out.println("5 classes de saida ok");
        // 3 classes de saida
        d.Discretizacao(maior, menor);
        nome = Config.VALUES3.split(",");
        for (int i = 0; i < dadosTitleNum.size(); i++) {
            if (Double.parseDouble(dadosTitleNum.get(i)) >= Math.floor(menor)
                    && Double.parseDouble(dadosTitleNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
```

```
            dadosTitleDiscretizados.add(nome[0]);
        } else if (Double.parseDouble(dadosTitleNum.get(i)) >= (Math.floor(menor) + d.discretizar)
                && Double.parseDouble(dadosTitleNum.get(i)) < ((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) {
            dadosTitleDiscretizados.add(nome[1]);
        } else if (Double.parseDouble(dadosTitleNum.get(i)) >= ((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)
                && Double.parseDouble(dadosTitleNum.get(i)) <= (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar) {
            dadosTitleDiscretizados.add(nome[2]);

        }
    }

    for (int i = 0; i < dadosTitleDiscretizados.size(); i++) {
        bfTitleTree.append(dadosTitleDiscretizados.get(i) + "\r\n");
    }
    bfTitleTree.close();
    System.out.println("dadosTitleDiscretizados: " + dadosTitleDiscretizados.size());
    dadosTitleDiscretizados = null;
    dadosTitleDiscretizados = new ArrayList();
    System.out.println("3 classes de saida ok");

    d.Discretizacao2(maior, menor);
    nome = Config.VALUES2.split(",");
    for (int i = 0; i < dadosTitleNum.size(); i++) {
        if (Double.parseDouble(dadosTitleNum.get(i)) >= Math.floor(menor)
                && Double.parseDouble(dadosTitleNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
            dadosTitleDiscretizados.add(nome[0]);
        } else if (Double.parseDouble(dadosTitleNum.get(i)) >= (Math.floor(menor) + d.discretizar)
                && Double.parseDouble(dadosTitleNum.get(i)) <= ((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) {
            dadosTitleDiscretizados.add(nome[1]);
        }

    }

    for (int i = 0; i < dadosTitleDiscretizados.size(); i++) {
        bfTitleTwo.append(dadosTitleDiscretizados.get(i) + "\r\n");
    }
    bfTitleTwo.close();
    System.out.println("dadosTitleDiscretizados: " + dadosTitleDiscretizados.size());
    dadosTitleDiscretizados = null;
    System.out.println("2 classes de saida ok");
    System.out.println("Maior em title: " + maior);
    System.out.println("Menor em title: " + menor);

    namesBf5.append("title " + Config.VALUES5 + "\r\n");
    namesBf3.append("title " + Config.VALUES3 + "\r\n");
    namesBf2.append("title " + Config.VALUES2 + "\r\n");

    dadosTitle = null;
    dadosTitleNum = null;
    System.out.println("Concluido");
    System.out.println("");
    System.out.println("|__ Adaptando Type:");
    //Fim do tratamento de title
    //Inicio do tratamento de type
    //Armazenando o tipo em uma lista
    arq = typeBuff.readLine();
```

```
while (arq != null) {
   arq = arq.replace(" ", "_");
   dadosType.add(arq);
   arq = typeBuff.readLine();
}
//Realizando a contagem de type
contadorPalavraAlvo = 0;
for (int i = 0; i < dadosType.size(); i++) {
   if (qtdDocumentos % 5000 == 0 && qtdDocumentos != 0) {
      System.out.println(qtdDocumentos + " documentos processados");
   }
   controlador = 0;
   palavraType = dadosType.get(i);
   for (int z = 0; z < dadosTypeContados.size(); z++) {
      if (!dadosTypeContados.isEmpty()) {
         palavras = dadosTypeContados.get(z).split("-");
         if (palavraType.equals(palavras[0])) {
            controlador = 1;
            break;
         }
      }
   }
   if (controlador == 0) {
      for (int j = 0; j < dadosType.size(); j++) {
         if (palavraType.equals(dadosType.get(j))) {
            contadorPalavraAlvo++;
         }
      }
      if (contadorPalavraAlvo < 10) {
         dadosTypeContados.add(palavraType + "-0" + contadorPalavraAlvo);
         contadorPalavraAlvo = 0;
      } else {
         dadosTypeContados.add(palavraType + "-" + contadorPalavraAlvo);
         contadorPalavraAlvo = 0;
      }
   }
   qtdDocumentos++;
}
int achou = 0;
//Verificando quantos tipos de documento têm na base
for (int i = 0; i < dadosTypeContados.size(); i++) {
   String variavel = dadosTypeContados.get(i);
   for (int j = 0; j < dadosTypeContados.size(); j++) {
      if (i != j) {
         if (variavel.equals(dadosTypeContados.get(j)) && !variavel.equals("a")) {
            dadosTypeContados.set(j, "a");
            achou = 1;
         }
      }
   }
   if (achou == 1) {
      i = 0;
   }
   achou = 0;
}
achou = 0;
for (int i = 0; i < dadosTypeContados.size(); i++) {
   String variavel = dadosTypeContados.get(i);
   for (int j = 0; j < dadosTypeContados.size(); j++) {
      if (variavel.equals(dadosTypeContados.get(j)) && variavel.equals("a")) {
```

```
                dadosTypeContados.remove(j);
                achou = 1;
            }

        }
        if (achou == 1) {
            i = 0;
        }
        achou = 0;
    }
    String termo = "";
    int controlId = 0;
    for (int i = 0; i < dadosTypeContados.size(); i++) {
        controlador = 0;
        String[] palavraAComparar = dadosTypeContados.get(i).split("-");
        int num = Integer.parseInt(palavraAComparar[1]);
        for (int j = i; j < dadosTypeContados.size(); j++) {
            String[] palavraAComparar1 = dadosTypeContados.get(j).split("-");
            menor = Integer.parseInt(palavraAComparar1[1]);
            if (num > menor) {
                controlador = 1;
                num = menor;
                termo = palavraAComparar1[0] + "-" + palavraAComparar1[1];
                controlId = j;
            }
        }
        if (controlador == 1) {

            dadosTypeContados.set(i, termo);
            dadosTypeContados.set(controlId, palavraAComparar[0] + "-" + palavraAComparar[1]);
        }
    }

    System.out.println("Pontuações dos tipos de documento:");
    System.out.println("----------------------------------");
    for (int i = 0; i < dadosTypeContados.size(); i++) {
        System.out.println(i + " - " + dadosTypeContados.get(i));
    }
    System.out.println("----------------------------------");
    System.out.println("Em caso de tipos com mesma pontuação, acesse o arquivo .names na seção types e
altere a ordem conforme seu critério");

    int count = 0;
    for (int i = 0; i < dadosTypeContados.size(); i++) {
        String[] palavraType1 = dadosTypeContados.get(i).split("-");
        if (count == 0) {
            palavraCompletaType = palavraCompletaType + palavraType1[0];
            count = 1;
        } else if (count == 1) {
            palavraCompletaType = palavraCompletaType + "," + palavraType1[0];
        }

    }
    System.out.println("Maior em type: " + maior);
    System.out.println("Menor em type: " + menor);

    namesBf5.append("type " + palavraCompletaType + "\r\n");
    namesBf3.append("type " + palavraCompletaType + "\r\n");
    namesBf2.append("type " + palavraCompletaType + "\r\n");
```

```
System.out.println("dadosTypeDiscretizados: " + dadosType.size());
dadosTypeContados = null;
System.out.println("Concluído");
//Início do processo de adaptação de source
//Armazenando o source em uma lista
System.out.println("");
System.out.println("|__ Adaptando Source");
arq = sourceBuff.readLine();
while (arq != null) {
   arq = arq.replace(" ", "_");
   arq = arq.replace("-", "_");
   dadosSource.add(arq);
   arq = sourceBuff.readLine();
}

contadorPalavraAlvo = 0;
qtdDocumentos = 0;
for (int i = 0; i < dadosSource.size(); i++) {
   if (qtdDocumentos % 5000 == 0 && qtdDocumentos != 0) {
      System.out.println(qtdDocumentos + " documentos processados");
   }
   controlador = 0;
   String palavraSource = dadosSource.get(i);
   for (int z = 0; z < dadosSourceContados.size(); z++) {
      if (!dadosSourceContados.isEmpty()) {
         palavras = dadosSourceContados.get(z).split("-");
         if (palavraSource.equals(palavras[0])) {
            controlador = 1;
            break;
         }
      }
   }
   if (controlador == 0) {
      for (int j = 0; j < dadosSource.size(); j++) {
         if (palavraSource.equals(dadosSource.get(j))) {
            contadorPalavraAlvo++;
         }
      }
      if (contadorPalavraAlvo < 10) {
         dadosSourceContados.add(palavraSource + "-0" + contadorPalavraAlvo);
         contadorPalavraAlvo = 0;
      } else {
         dadosSourceContados.add(palavraSource + "-" + contadorPalavraAlvo);
         contadorPalavraAlvo = 0;
      }
   }
   qtdDocumentos++;
}

termo = "";
controlId = 0;
for (int i = 0; i < dadosSourceContados.size(); i++) {
   controlador = 0;
   String[] palavraAComparar = dadosSourceContados.get(i).split("-");
   int num = Integer.parseInt(palavraAComparar[1]);

   for (int j = i; j < dadosSourceContados.size(); j++) {
      String[] palavraAComparar1 = dadosSourceContados.get(j).split("-");
      menor = Integer.parseInt(palavraAComparar1[1]);
      if (num > menor) {
```

```
                controlador = 1;
                num = menor;
                termo = palavraAComparar1[0] + "-" + palavraAComparar1[1];
                controlId = j;
            }
        }
        if (controlador == 1) {

            dadosSourceContados.set(i, termo);
            dadosSourceContados.set(controlId, palavraAComparar[0] + "-" + palavraAComparar[1]);
        }

    }

    System.out.println("Pontuações dos source de documento:");
    System.out.println("----------------------------------");
    for (int i = 0; i < dadosSourceContados.size(); i++) {
        System.out.println(i + " - " + dadosSourceContados.get(i));
    }
    System.out.println("----------------------------------");
    System.out.println("Em caso de source com mesma pontuação, acesse o arquivo .names na seção source e
altere a ordem conforme seu critério");

    count = 0;
    for (int i = 0; i < dadosSourceContados.size(); i++) {
        String[] palavraType1 = dadosSourceContados.get(i).split("-");
        if (count == 0) {
            palavraCompletaSource = palavraCompletaSource + palavraType1[0];
            count = 1;
        } else if (count == 1) {
            palavraCompletaSource = palavraCompletaSource + "," + palavraType1[0];
        }

    }
    System.out.println("Maior em source: " + maior);
    System.out.println("Menor em source: " + menor);

    namesBf5.append("source " + palavraCompletaSource + "\r\n");
    namesBf3.append("source " + palavraCompletaSource + "\r\n");
    namesBf2.append("source " + palavraCompletaSource + "\r\n");

    System.out.println("dadosSource: " + dadosSource.size());
    System.out.println("concluido");
    dadosSourceContados = null;

    System.out.println("");
    System.out.println("|__ Adaptando Year");
    arq = yearBuff.readLine();
    while (arq != null) {
        dadosYear.add(arq);
        arq = yearBuff.readLine();
    }
    for (int i = 0; i < dadosYear.size(); i++) {
        int num = Integer.parseInt(dadosYear.get(i));
        int num2 = Integer.parseInt(dadosYear.get(i));
        for (int j = 0; j < dadosYear.size(); j++) {
            if (i != j) {

            }
            if (num > Integer.parseInt(dadosYear.get(j))) {
```

```
            maior = num;
        } else if (num < Integer.parseInt(dadosYear.get(j))) {
            num = Integer.parseInt(dadosYear.get(j));
            maior = num;
        }
        if (num2 > Integer.parseInt(dadosYear.get(j))) {
            num2 = Integer.parseInt(dadosYear.get(j));
            menor = num2;
        } else if (num2 < Integer.parseInt(dadosYear.get(j))) {
            menor = num2;
        }
    }
}
nome = Config.VALUES_YEAR.split(",");
for (int i = 0; i < dadosYear.size(); i++) {
    if (Integer.parseInt(dadosYear.get(i)) >= menor && Integer.parseInt(dadosYear.get(i)) <= 1999) {
        dadosYearDiscretizados.add(nome[0]);
    } else if (Integer.parseInt(dadosYear.get(i)) > 1999 && Integer.parseInt(dadosYear.get(i)) <= 2007) {
        dadosYearDiscretizados.add(nome[1]);
    } else if (Integer.parseInt(dadosYear.get(i)) > 2007 && Integer.parseInt(dadosYear.get(i)) <= 2011) {
        dadosYearDiscretizados.add(nome[2]);
    } else if (Integer.parseInt(dadosYear.get(i)) > 2011 && Integer.parseInt(dadosYear.get(i)) <= 2015) {
        dadosYearDiscretizados.add(nome[3]);
    } else if (Integer.parseInt(dadosYear.get(i)) > 2015 && Integer.parseInt(dadosYear.get(i)) <= 2017) {
        dadosYearDiscretizados.add(nome[4]);
    }
}

System.out.println("Maior em year: " + maior);
System.out.println("Menor em year: " + menor);

namesBf5.append("year " + Config.VALUES_YEAR + "\r\n");
namesBf3.append("year " + Config.VALUES_YEAR + "\r\n");
namesBf2.append("year " + Config.VALUES_YEAR + "\r\n");

System.out.println("dadosYearDiscretizados: " + dadosYearDiscretizados.size());

System.out.println("Concluído");
//condição para keywords
dadosYear = null;
System.out.println("");
System.out.println("|__ Adaptando Keywords:");
File keywordsE1 = new File(Config.CAMINHO_ADAPTADO + "keywords_num.txt");
BufferedReader keywords1Bf = new BufferedReader(new FileReader(keywordsE1));
arq = keywords1Bf.readLine();
while (arq != null) {
    dadosKeywordsNum.add(arq);
    arq = keywords1Bf.readLine();
}
System.out.println(dadosKeywordsNum.size());

for (int i = 0; i < dadosKeywordsNum.size(); i++) {
    int num = Integer.parseInt(dadosKeywordsNum.get(i));
    int num2 = Integer.parseInt(dadosKeywordsNum.get(i));
    for (int j = 0; j < dadosKeywordsNum.size(); j++) {
        if (num > Integer.parseInt(dadosKeywordsNum.get(j))) {
            maior = num;
        } else if (num < Integer.parseInt(dadosKeywordsNum.get(j))) {
            num = Integer.parseInt(dadosKeywordsNum.get(j));
            maior = num;
```

```
            }
            if (num2 > Integer.parseInt(dadosKeywordsNum.get(j))) {
               num2 = Integer.parseInt(dadosKeywordsNum.get(j));
               menor = num2;
            } else if (num2 < Integer.parseInt(dadosKeywordsNum.get(j))) {
               menor = num2;
            }
         }
      }
      d.Discretizacao5(maior, menor);
      nome = Config.VALUES5.split(",");
      for (int i = 0; i < dadosKeywordsNum.size(); i++) {
         if (Double.parseDouble(dadosKeywordsNum.get(i)) >= Math.floor(menor)
               && Double.parseDouble(dadosKeywordsNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
            dadosKeywordsDiscretizados.add(nome[0]);
         } else if (Double.parseDouble(dadosKeywordsNum.get(i)) >= (Math.floor(menor) + d.discretizar)
               && Double.parseDouble(dadosKeywordsNum.get(i)) < ((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)) {
            dadosKeywordsDiscretizados.add(nome[1]);
         } else if (Double.parseDouble(dadosKeywordsNum.get(i)) >= ((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)
               && Double.parseDouble(dadosKeywordsNum.get(i)) < (((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)) + d.discretizar) {
            dadosKeywordsDiscretizados.add(nome[2]);
         } else if (Double.parseDouble(dadosKeywordsNum.get(i)) >= (((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)) + d.discretizar
               && Double.parseDouble(dadosKeywordsNum.get(i)) < (((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)) + d.discretizar * 2.0f) {
            dadosKeywordsDiscretizados.add(nome[3]);
         } else if (Double.parseDouble(dadosKeywordsNum.get(i)) >= (((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)) + d.discretizar * 2.0f
               && Double.parseDouble(dadosKeywordsNum.get(i)) <= (((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)) + d.discretizar * 3.0f) {
            dadosKeywordsDiscretizados.add(nome[4]);

         }
      }

      for (int i = 0; i < dadosKeywordsDiscretizados.size(); i++) {
         bfKeywordsFive.append(dadosKeywordsDiscretizados.get(i) + "\r\n");
      }
      bfKeywordsFive.close();
      System.out.println("dadosKeywordsDiscretizados: " + dadosKeywordsDiscretizados.size());
      dadosKeywordsDiscretizados = null;
      dadosKeywordsDiscretizados = new ArrayList();
      System.out.println("5 classes de saida ok");

      d.Discretizacao(maior, menor);
      nome = Config.VALUES3.split(",");
      for (int i = 0; i < dadosKeywordsNum.size(); i++) {
         if (Double.parseDouble(dadosKeywordsNum.get(i)) >= Math.floor(menor)
               && Double.parseDouble(dadosKeywordsNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
            dadosKeywordsDiscretizados.add(nome[0]);
         } else if (Double.parseDouble(dadosKeywordsNum.get(i)) >= (Math.floor(menor) + d.discretizar)
               && Double.parseDouble(dadosKeywordsNum.get(i)) < ((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)) {
            dadosKeywordsDiscretizados.add(nome[1]);
         } else if (Double.parseDouble(dadosKeywordsNum.get(i)) >= ((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)
```

```
            && Double.parseDouble(dadosKeywordsNum.get(i)) <= (((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)) + d.discretizar) {
            dadosKeywordsDiscretizados.add(nome[2]);
        }

    }

    for (int i = 0; i < dadosKeywordsDiscretizados.size(); i++) {
        bfKeywordsTree.append(dadosKeywordsDiscretizados.get(i) + "\r\n");
    }
    bfKeywordsTree.close();
    System.out.println("dadosKeywordsDiscretizados: " + dadosKeywordsDiscretizados.size());
    dadosKeywordsDiscretizados = null;
    dadosKeywordsDiscretizados = new ArrayList();
    System.out.println("3 classes de saida ok");
    d.Discretizacao2(maior, menor);
    nome = Config.VALUES2.split(",");
    for (int i = 0; i < dadosKeywordsNum.size(); i++) {
        if (Double.parseDouble(dadosKeywordsNum.get(i)) >= Math.floor(menor)
            && Double.parseDouble(dadosKeywordsNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
            dadosKeywordsDiscretizados.add(nome[0]);
        } else if (Double.parseDouble(dadosKeywordsNum.get(i)) >= (Math.floor(menor) + d.discretizar)
            && Double.parseDouble(dadosKeywordsNum.get(i)) <= ((Math.floor(menor) + d.discretizar) *
2.0f) - Math.floor(menor)) {
            dadosKeywordsDiscretizados.add(nome[1]);
        }

    }
    for (int i = 0; i < dadosKeywordsDiscretizados.size(); i++) {
        bfKeywordsTwo.append(dadosKeywordsDiscretizados.get(i) + "\r\n");
    }
    bfKeywordsTwo.close();
    System.out.println("dadosKeywordsDiscretizados: " + dadosKeywordsDiscretizados.size());
    dadosKeywordsDiscretizados = null;
    System.out.println("2 classes de saida ok");
    System.out.println("Maior em keywords: " + maior);
    System.out.println("Menor em keywords: " + menor);

    namesBf5.append("keywords " + Config.VALUES5 + "\r\n");
    namesBf3.append("keywords " + Config.VALUES3 + "\r\n");
    namesBf2.append("keywords " + Config.VALUES2 + "\r\n");

    d.discretizar = 0;
    System.out.println("concluido");
    dadosKeywordsNum = null;
    //Iniciando adaptação de authors
    System.out.println("");
    System.out.println("|__ Adaptando Authors:");
    arq = bfNewAuthors.readLine();
    while (arq != null) {
        dadosAuthorsContadosPorId.add(arq);
        arq = bfNewAuthors.readLine();
    }

    System.out.println("Realizando a discretização, aguarde...");

    System.out.println(dadosAuthorsContadosPorId.size());
    for (int i = 0; i < dadosAuthorsContadosPorId.size(); i++) {
        int num = Integer.parseInt(dadosAuthorsContadosPorId.get(i));
        int num2 = Integer.parseInt(dadosAuthorsContadosPorId.get(i));
```

```
            for (int j = 0; j < dadosAuthorsContadosPorId.size(); j++) {
               if (num > Integer.parseInt(dadosAuthorsContadosPorId.get(j))) {
                  maior = num;
               } else if (num < Integer.parseInt(dadosAuthorsContadosPorId.get(j))) {
                  num = Integer.parseInt(dadosAuthorsContadosPorId.get(j));
                  maior = num;
               }
               if (num2 > Integer.parseInt(dadosAuthorsContadosPorId.get(j))) {
                  num2 = Integer.parseInt(dadosAuthorsContadosPorId.get(j));
                  menor = num2;
               } else if (num2 < Integer.parseInt(dadosAuthorsContadosPorId.get(j))) {
                  menor = num2;
               }
            }
         }
         d.Discretizacao5(maior, menor);
         nome = Config.VALUES5.split(",");
         for (int i = 0; i < dadosAuthorsContadosPorId.size(); i++) {
            if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= Math.floor(menor)
                  && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) < (Math.floor(menor) + d.discretizar))
{
               dadosAuthorsDiscretizados.add(nome[0]);
            } else if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= (Math.floor(menor) +
d.discretizar)
                  && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) < ((Math.floor(menor) + d.discretizar)
* 2.0f) - Math.floor(menor)) {
               dadosAuthorsDiscretizados.add(nome[1]);
            } else if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= ((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)
                  && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) < (((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)) + d.discretizar) {
               dadosAuthorsDiscretizados.add(nome[2]);
            } else if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= (((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)) + d.discretizar
                  && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) < (((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)) + d.discretizar * 2.0f) {
               dadosAuthorsDiscretizados.add(nome[3]);
            } else if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= (((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)) + d.discretizar * 2.0f
                  && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) <= (((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)) + d.discretizar * 3.0f) {
               dadosAuthorsDiscretizados.add(nome[4]);

            }
         }

         for (int i = 0; i < dadosAuthorsDiscretizados.size(); i++) {
            bfAuthorsFive.append(dadosAuthorsDiscretizados.get(i) + "\r\n");
         }
         bfAuthorsFive.close();
         System.out.println("dadosAuthorsDiscretizados: " + dadosAuthorsDiscretizados.size());
         dadosAuthorsDiscretizados = null;
         System.out.println("5 classes de saida ok");
         dadosAuthorsDiscretizados = new ArrayList();

         d.Discretizacao(maior, menor);
         nome = Config.VALUES3.split(",");
         for (int i = 0; i < dadosAuthorsContadosPorId.size(); i++) {
            if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= Math.floor(menor)
```

```
            && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) < (Math.floor(menor) + d.discretizar))
{
            dadosAuthorsDiscretizados.add(nome[0]);
        } else if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= (Math.floor(menor) +
d.discretizar)
            && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) < ((Math.floor(menor) + d.discretizar)
* 2.0f) - Math.floor(menor)) {
            dadosAuthorsDiscretizados.add(nome[1]);
        } else if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= ((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)
            && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) <= (((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)) + d.discretizar) {
            dadosAuthorsDiscretizados.add(nome[2]);
        }

    }

    for (int i = 0; i < dadosAuthorsDiscretizados.size(); i++) {
      bfAuthorsTree.append(dadosAuthorsDiscretizados.get(i) + "\r\n");
    }
    bfAuthorsTree.close();
    System.out.println("dadosAuthorsDiscretizados: " + dadosAuthorsDiscretizados.size());
    dadosAuthorsDiscretizados = null;
    System.out.println("3 classes de saida ok");
    dadosAuthorsDiscretizados = new ArrayList();

    d.Discretizacao2(maior, menor);
    nome = Config.VALUES2.split(",");
    for (int i = 0; i < dadosAuthorsContadosPorId.size(); i++) {
      if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= Math.floor(menor)
            && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) < (Math.floor(menor) + d.discretizar))
{
            dadosAuthorsDiscretizados.add(nome[0]);
        } else if (Double.parseDouble(dadosAuthorsContadosPorId.get(i)) >= (Math.floor(menor) +
d.discretizar)
            && Double.parseDouble(dadosAuthorsContadosPorId.get(i)) <= ((Math.floor(menor) +
d.discretizar) * 2.0f) - Math.floor(menor)) {
            dadosAuthorsDiscretizados.add(nome[1]);
        }
    }

    for (int i = 0; i < dadosAuthorsDiscretizados.size(); i++) {
      bfAuthorsTwo.append(dadosAuthorsDiscretizados.get(i) + "\r\n");
    }
    bfAuthorsTwo.close();
    System.out.println("dadosAuthorsDiscretizados: " + dadosAuthorsDiscretizados.size());
    dadosAuthorsDiscretizados = null;
    System.out.println("2 classes de saida ok");
    System.out.println("Maior em authors: " + maior);
    System.out.println("Menor em authors: " + menor);

    namesBf5.append("authors " + Config.VALUES5 + "\r\n");
    namesBf3.append("authors " + Config.VALUES3 + "\r\n");
    namesBf2.append("authors " + Config.VALUES2 + "\r\n");

    System.out.println("Concluido");
    dadosAuthorsContadosPorId = null;
    //Armazenando o year em uma lista
    System.out.println("");
    System.out.println("|__ Adaptando Month:");
```

```
arq = monthBuff.readLine();
while (arq != null) {
   dadosMonth.add(arq);
   arq = monthBuff.readLine();
}
System.out.println(dadosMonth.size());
for (int i = 0; i < dadosMonth.size(); i++) {
   if (dadosMonth.get(i).equals("1")) {
      dadosMonthName.add("janeiro");
   } else if (dadosMonth.get(i).equals("2")) {
      dadosMonthName.add("fevereiro");
   } else if (dadosMonth.get(i).equals("3")) {
      dadosMonthName.add("marco");
   } else if (dadosMonth.get(i).equals("4")) {
      dadosMonthName.add("abril");
   } else if (dadosMonth.get(i).equals("5")) {
      dadosMonthName.add("maio");
   } else if (dadosMonth.get(i).equals("6")) {
      dadosMonthName.add("junho");
   } else if (dadosMonth.get(i).equals("7")) {
      dadosMonthName.add("julho");
   } else if (dadosMonth.get(i).equals("8")) {
      dadosMonthName.add("agosto");
   } else if (dadosMonth.get(i).equals("9")) {
      dadosMonthName.add("setembro");
   } else if (dadosMonth.get(i).equals("10")) {
      dadosMonthName.add("outubro");
   } else if (dadosMonth.get(i).equals("11")) {
      dadosMonthName.add("novembro");
   } else if (dadosMonth.get(i).equals("12")) {
      dadosMonthName.add("dezembro");
   }
}
for (int i = 0; i < month.length; i++) {
   month[i] = 0;
}
//contando cada mes na base de dados toda
for (int i = 0; i < dadosMonth.size(); i++) {
   if (dadosMonth.get(i).equals("1")) {
      month[0]++;
   } else if (dadosMonth.get(i).equals("2")) {
      month[1]++;
   } else if (dadosMonth.get(i).equals("3")) {
      month[2]++;
   } else if (dadosMonth.get(i).equals("4")) {
      month[3]++;
   } else if (dadosMonth.get(i).equals("5")) {
      month[4]++;
   } else if (dadosMonth.get(i).equals("6")) {
      month[5]++;
   } else if (dadosMonth.get(i).equals("7")) {
      month[6]++;
   } else if (dadosMonth.get(i).equals("8")) {
      month[7]++;
   } else if (dadosMonth.get(i).equals("9")) {
      month[8]++;
   } else if (dadosMonth.get(i).equals("10")) {
      month[9]++;
   } else if (dadosMonth.get(i).equals("11")) {
      month[10]++;
```

```
    } else if (dadosMonth.get(i).equals("12")) {
        month[11]++;
    }
}

contadorOcorrencia = 0;
count = 0;
qtdDocumentos = 0;

for (int i = 0; i < month.length; i++) {
    if (month[i] != -1) {
        int num = month[i];
        contadorOcorrencia = i;
        for (int j = 0; j < month.length; j++) {
            if (num > month[j] && month[j] != -1) {
                num = month[j];
                //menor = month[j];
                contadorOcorrencia = j;
            }
        }
        if (i < month.length) {
            if (count == 0) {
                count = 1;
                if (contadorOcorrencia == 0) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "janeiro";
                } else if (contadorOcorrencia == 1) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "fevereiro";
                } else if (contadorOcorrencia == 2) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "marco";
                } else if (contadorOcorrencia == 3) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "abril";
                } else if (contadorOcorrencia == 4) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "maio";
                } else if (contadorOcorrencia == 5) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "junho";
                } else if (contadorOcorrencia == 6) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "julho";
                } else if (contadorOcorrencia == 7) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "agosto";
                } else if (contadorOcorrencia == 8) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "setembro";
                } else if (contadorOcorrencia == 9) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "outubro";
                } else if (contadorOcorrencia == 10) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "novembro";
                } else if (contadorOcorrencia == 11) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + "dezembro";
                }
                month[contadorOcorrencia] = -1;
                qtdDocumentos++;
                i = -1;
            } else if (count == 1) {
                if (contadorOcorrencia == 0) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + ",janeiro";
                } else if (contadorOcorrencia == 1) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + ",fevereiro";
                } else if (contadorOcorrencia == 2) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + ",marco";
                } else if (contadorOcorrencia == 3) {
                    Config.VALUES_MONTH = Config.VALUES_MONTH + ",abril";
```

```
            } else if (contadorOcorrencia == 4) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",maio";
            } else if (contadorOcorrencia == 5) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",junho";
            } else if (contadorOcorrencia == 6) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",julho";
            } else if (contadorOcorrencia == 7) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",agosto";
            } else if (contadorOcorrencia == 8) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",setembro";
            } else if (contadorOcorrencia == 9) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",outubro";
            } else if (contadorOcorrencia == 10) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",novembro";
            } else if (contadorOcorrencia == 11) {
               Config.VALUES_MONTH = Config.VALUES_MONTH + ",dezembro";
            }
            month[contadorOcorrencia] = -1;
            qtdDocumentos++;
            i = -1;
         }
      }
   }
}

namesBf5.append("month " + Config.VALUES_MONTH + "\r\n");
namesBf3.append("month " + Config.VALUES_MONTH + "\r\n");
namesBf2.append("month " + Config.VALUES_MONTH + "\r\n");

//Tratamento abstract
System.out.println("Concluido");
dadosMonth = null;
//condição para abstract
System.out.println("");
System.out.println("|__ Adaptando abstract:");
System.out.println("Esse processo pode levar um tempo");
arq = bfNewAbstract.readLine();
while (arq != null) {
   dadosAbstract.add(arq);
   arq = bfNewAbstract.readLine();
}
System.out.println("Processando documentos. Total: " + dadosAbstract.size());

for (int i = 0; i < dadosAbstract.size(); i++) {
   if (i % 500 == 0 || i < 100) {
      System.out.println(i + " documentos processados");
   }
   palavras = dadosAbstract.get(i).split(" ");
   for (int j = 0; j < palavras.length; j++) {
      String[] palavras1 = palavras[j].split("_");
      if (Integer.parseInt(palavras1[1]) <= Config.QTD_FREQUENCIA_ABSTRACT) {
         absAbstract = new File(Config.CAMINHO_TITLE_ABSTRACT + "Abstract.txt");
         bfAbsAbstract = new BufferedReader(new FileReader(absAbstract));
         arq = bfAbsAbstract.readLine();
         String[] palavraContada = arq.split("_");
         while (arq != null) {
            if (arq.equals(palavras[j])) {
               contadorOcorrencia = contadorOcorrencia + Integer.parseInt(palavraContada[1]);
               arq = null;
            } else {
```

```
                    arq = bfAbsAbstract.readLine();
                }
                if (arq != null) {
                    palavraContada = arq.split("_");
                }
            }
            bfAbsAbstract.close();
        }
    }
    dadosAbstractNum.add(String.valueOf(contadorOcorrencia));
    contadorOcorrencia = 0;
}
System.out.println("Aplicando a discretização de abstract. Aguarde uns instantes...");

for (int i = 0; i < dadosAbstractNum.size(); i++) {
    int num = Integer.parseInt(dadosAbstractNum.get(i));
    int num2 = Integer.parseInt(dadosAbstractNum.get(i));
    for (int j = 0; j < dadosAbstractNum.size(); j++) {
        if (num > Integer.parseInt(dadosAbstractNum.get(j))) {
            maior = num;
        } else if (num < Integer.parseInt(dadosAbstractNum.get(j))) {
            num = Integer.parseInt(dadosAbstractNum.get(j));
            maior = num;
        }
        if (num2 > Integer.parseInt(dadosAbstractNum.get(j))) {
            num2 = Integer.parseInt(dadosAbstractNum.get(j));
            menor = num2;
        } else if (num2 < Integer.parseInt(dadosAbstractNum.get(j))) {
            menor = num2;
        }
    }
}
d.Discretizacao5(maior, menor);
nome = Config.VALUES5.split(",");
for (int i = 0; i < dadosAbstractNum.size(); i++) {
    if (Double.parseDouble(dadosAbstractNum.get(i)) >= Math.floor(menor)
            && Double.parseDouble(dadosAbstractNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
        dadosAbstractDiscretizados.add(nome[0]);
    } else if (Double.parseDouble(dadosAbstractNum.get(i)) >= (Math.floor(menor) + d.discretizar)
            && Double.parseDouble(dadosAbstractNum.get(i)) < ((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) {
        dadosAbstractDiscretizados.add(nome[1]);
    } else if (Double.parseDouble(dadosAbstractNum.get(i)) >= ((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)
            && Double.parseDouble(dadosAbstractNum.get(i)) < (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar) {
        dadosAbstractDiscretizados.add(nome[2]);
    } else if (Double.parseDouble(dadosAbstractNum.get(i)) >= (((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)) + d.discretizar
            && Double.parseDouble(dadosAbstractNum.get(i)) < (((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) + d.discretizar * 2.0f) {
        dadosAbstractDiscretizados.add(nome[3]);
    } else if (Double.parseDouble(dadosAbstractNum.get(i)) >= (((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)) + d.discretizar * 2.0f
            && Double.parseDouble(dadosAbstractNum.get(i)) <= (((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)) + d.discretizar * 3.0f) {
        dadosAbstractDiscretizados.add(nome[4]);

    }
}
```

```
        for (int i = 0; i < dadosAbstractDiscretizados.size(); i++) {
            bfAbstractFive.append(dadosAbstractDiscretizados.get(i) + "\r\n");
        }
        bfAbstractFive.close();
        System.out.println("dadosAbstractDiscretizados: " + dadosAbstractDiscretizados.size());
        dadosAbstractDiscretizados = null;
        dadosAbstractDiscretizados = new ArrayList();
        System.out.println("5 classes de saida ok");
        d.Discretizacao(maior, menor);
        nome = Config.VALUES3.split(",");
        for (int i = 0; i < dadosAbstractNum.size(); i++) {
            if (Double.parseDouble(dadosAbstractNum.get(i)) >= Math.floor(menor)
                    && Double.parseDouble(dadosAbstractNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
                dadosAbstractDiscretizados.add(nome[0]);
            } else if (Double.parseDouble(dadosAbstractNum.get(i)) >= (Math.floor(menor) + d.discretizar)
                    && Double.parseDouble(dadosAbstractNum.get(i)) < ((Math.floor(menor) + d.discretizar) * 2.0f) -
Math.floor(menor)) {
                dadosAbstractDiscretizados.add(nome[1]);
            } else if (Double.parseDouble(dadosAbstractNum.get(i)) >= ((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)
                    && Double.parseDouble(dadosAbstractNum.get(i)) <= (((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)) + d.discretizar) {
                dadosAbstractDiscretizados.add(nome[2]);
            }
        }

        for (int i = 0; i < dadosAbstractDiscretizados.size(); i++) {
            bfAbstractTree.append(dadosAbstractDiscretizados.get(i) + "\r\n");
        }
        bfAbstractTree.close();
        System.out.println("dadosAbstractDiscretizados: " + dadosAbstractDiscretizados.size());
        dadosAbstractDiscretizados = null;
        dadosAbstractDiscretizados = new ArrayList();
        System.out.println("3 classes de saida ok");
        d.Discretizacao2(maior, menor);
        nome = Config.VALUES2.split(",");
        for (int i = 0; i < dadosAbstractNum.size(); i++) {
            if (Double.parseDouble(dadosAbstractNum.get(i)) >= Math.floor(menor)
                    && Double.parseDouble(dadosAbstractNum.get(i)) < (Math.floor(menor) + d.discretizar)) {
                dadosAbstractDiscretizados.add(nome[0]);
            } else if (Double.parseDouble(dadosAbstractNum.get(i)) >= (Math.floor(menor) + d.discretizar)
                    && Double.parseDouble(dadosAbstractNum.get(i)) <= ((Math.floor(menor) + d.discretizar) * 2.0f)
- Math.floor(menor)) {
                dadosAbstractDiscretizados.add(nome[1]);
            }

        }

        for (int i = 0; i < dadosAbstractDiscretizados.size(); i++) {
            bfAbstractTwo.append(dadosAbstractDiscretizados.get(i) + "\r\n");
        }
        bfAbstractTwo.close();
        System.out.println("dadosAbstractDiscretizados: " + dadosAbstractDiscretizados.size());
        dadosAbstractDiscretizados = null;

        namesBf5.append("abstract " + Config.VALUES5 + "\r\n");
        namesBf3.append("abstract " + Config.VALUES3 + "\r\n");
        namesBf2.append("abstract " + Config.VALUES2 + "\r\n");
```

```
        System.out.println("Maior em abstract: " + maior);
        System.out.println("Menor em abstract: " + menor);
        System.out.println("");
        dadosAbstract = null;
        dadosAbstractNum = null;
        System.out.println("2 classes de saida ok");
        System.out.println("|__ Adaptando Reader_Count:");
        arq = reader_countBuff.readLine();
        while (arq != null) {
           dadosReaderCount.add(arq);
           arq = reader_countBuff.readLine();
        }
        //Iniciando processo de discretização
        for (int i = 0; i < dadosReaderCount.size(); i++) {
           int num = Integer.parseInt(dadosReaderCount.get(i));
           int num2 = Integer.parseInt(dadosReaderCount.get(i));
           for (int j = 0; j < dadosReaderCount.size(); j++) {
              if (num > Integer.parseInt(dadosReaderCount.get(j))) {
                 maior = num;
              } else if (num < Integer.parseInt(dadosReaderCount.get(j))) {
                 num = Integer.parseInt(dadosReaderCount.get(j));
                 maior = num;
              }
              if (num2 > Integer.parseInt(dadosReaderCount.get(j))) {
                 num2 = Integer.parseInt(dadosReaderCount.get(j));
                 menor = num2;
              } else if (num2 < Integer.parseInt(dadosReaderCount.get(j))) {
                 menor = num2;
              }
           }
        }
        nome = Config.VALUES_READER_COUNT5.split(",");
        int[] contarReaderCount = new int[5];
        for (int i = 0; i < contarReaderCount.length; i++) {
           contarReaderCount[i] = 0;
        }

        for (int i = 0; i < dadosReaderCount.size(); i++) {
           d.DiscretizacaoPorcentagem(Integer.parseInt(dadosReaderCount.get(i)), maior);
           if (d.discretizar >= 0 && d.discretizar <= Config.PRIMEIRA_FAIXA5) {
              dadosReaderCountDiscretizados.add(nome[0]);
              contarReaderCount[0] = contarReaderCount[0] + Integer.parseInt(dadosReaderCount.get(i));
           } else if (d.discretizar > Config.PRIMEIRA_FAIXA5 && d.discretizar <=
Config.SEGUNDA_FAIXA5) {
              dadosReaderCountDiscretizados.add(nome[1]);
              contarReaderCount[1] = contarReaderCount[1] + Integer.parseInt(dadosReaderCount.get(i));
           } else if (d.discretizar > Config.SEGUNDA_FAIXA5 && d.discretizar <=
Config.TERCEIRA_FAIXA5) {
              dadosReaderCountDiscretizados.add(nome[2]);
              contarReaderCount[2] = contarReaderCount[2] + Integer.parseInt(dadosReaderCount.get(i));
           } else if (d.discretizar > Config.TERCEIRA_FAIXA5 && d.discretizar <= Config.QUARTA_FAIXA5)
{
              dadosReaderCountDiscretizados.add(nome[3]);
              contarReaderCount[3] = contarReaderCount[3] + Integer.parseInt(dadosReaderCount.get(i));
           } else if (d.discretizar > Config.QUARTA_FAIXA5 && d.discretizar <= Config.QUINTA_FAIXA5) {
              dadosReaderCountDiscretizados.add(nome[4]);
              contarReaderCount[4] = contarReaderCount[4] + Integer.parseInt(dadosReaderCount.get(i));
           }
        }
```

```
        for (int i = 0; i < dadosReaderCountDiscretizados.size(); i++) {
            bfReaderCountFive.append(dadosReaderCountDiscretizados.get(i) + "\r\n");
        }
        bfReaderCountFive.close();
        System.out.println("dadosReaderCountDiscretizados: " + dadosReaderCountDiscretizados.size());
        dadosReaderCountDiscretizados = null;
        dadosReaderCountDiscretizados = new ArrayList();
        System.out.println("5 classes de saida ok");
        contarReaderCount = new int[3];
        for (int i = 0; i < contarReaderCount.length; i++) {
            contarReaderCount[i] = 0;
        }
        nome = Config.VALUES_READER_COUNT3.split(",");
        for (int i = 0; i < dadosReaderCount.size(); i++) {
            d.DiscretizacaoPorcentagem(Integer.parseInt(dadosReaderCount.get(i)), maior);
            if (d.discretizar >= 0 && d.discretizar <= Config.PRIMEIRA_FAIXA3) {
                dadosReaderCountDiscretizados.add(nome[0]);
                contarReaderCount[0] = contarReaderCount[0] + Integer.parseInt(dadosReaderCount.get(i));
            } else if (d.discretizar > Config.PRIMEIRA_FAIXA3 && d.discretizar <=
Config.SEGUNDA_FAIXA3) {
                dadosReaderCountDiscretizados.add(nome[1]);
                contarReaderCount[1] = contarReaderCount[1] + Integer.parseInt(dadosReaderCount.get(i));
            } else if (d.discretizar > Config.SEGUNDA_FAIXA3 && d.discretizar <=
Config.TERCEIRA_FAIXA3) {
                dadosReaderCountDiscretizados.add(nome[2]);
                contarReaderCount[2] = contarReaderCount[2] + Integer.parseInt(dadosReaderCount.get(i));
            }

        }

        for (int i = 0; i < dadosReaderCountDiscretizados.size(); i++) {
            bfReaderCountTree.append(dadosReaderCountDiscretizados.get(i) + "\r\n");
        }
        bfReaderCountTree.close();
        System.out.println("dadosReaderCountDiscretizados: " + dadosReaderCountDiscretizados.size());
        dadosReaderCountDiscretizados = null;
        dadosReaderCountDiscretizados = new ArrayList();
        System.out.println("3 classes de saida ok");
        contarReaderCount = new int[2];
        for (int i = 0; i < contarReaderCount.length; i++) {
            contarReaderCount[i] = 0;
        }
        nome = Config.VALUES_READER_COUNT2.split(",");
        for (int i = 0; i < dadosReaderCount.size(); i++) {
            d.DiscretizacaoPorcentagem(Integer.parseInt(dadosReaderCount.get(i)), maior);
            if (d.discretizar >= 0 && d.discretizar <= Config.PRIMEIRA_FAIXA2) {
                dadosReaderCountDiscretizados.add(nome[0]);
                contarReaderCount[0] = contarReaderCount[0] + Integer.parseInt(dadosReaderCount.get(i));
            } else if (d.discretizar > Config.PRIMEIRA_FAIXA2 && d.discretizar <=
Config.SEGUNDA_FAIXA2) {
                dadosReaderCountDiscretizados.add(nome[1]);
                contarReaderCount[1] = contarReaderCount[1] + Integer.parseInt(dadosReaderCount.get(i));
            }

        }

        for (int i = 0; i < dadosReaderCountDiscretizados.size(); i++) {
            bfReaderCountTwo.append(dadosReaderCountDiscretizados.get(i) + "\r\n");
        }
        bfReaderCountTwo.close();
```

```
System.out.println("dadosReaderCountDiscretizados: " + dadosReaderCountDiscretizados.size());
dadosReaderCountDiscretizados = null;
System.out.println("2 classes de saida ok");
System.out.println("Maior em reader_count: " + maior);
System.out.println("Menor em reader_count: " + menor);

namesBf5.append("reader_count " + Config.VALUES_READER_COUNT5 + "\r\n");
namesBf3.append("reader_count " + Config.VALUES_READER_COUNT3 + "\r\n");
namesBf2.append("reader_count " + Config.VALUES_READER_COUNT2 + "\r\n");
namesBf5.close();
namesBf3.close();
namesBf2.close();

fTitleFive = new File(Config.CAMINHO_ADAPTADO + "title5.txt");
BufferedReader bTitleFive = new BufferedReader(new FileReader(fTitleFive));
fTitleTree = new File(Config.CAMINHO_ADAPTADO + "title3.txt");
BufferedReader bTitleTree = new BufferedReader(new FileReader(fTitleTree));
fTitleTwo = new File(Config.CAMINHO_ADAPTADO + "title2.txt");
BufferedReader bTitleTwo = new BufferedReader(new FileReader(fTitleTwo));

fKeywordsFive = new File(Config.CAMINHO_ADAPTADO + "keywords5.txt");
BufferedReader bKeywordsFive = new BufferedReader(new FileReader(fKeywordsFive));
fKeywordsTree = new File(Config.CAMINHO_ADAPTADO + "keywords3.txt");
BufferedReader bKeywordsTree = new BufferedReader(new FileReader(fKeywordsTree));
fKeywordsTwo = new File(Config.CAMINHO_ADAPTADO + "keywords2.txt");
BufferedReader bKeywordsTwo = new BufferedReader(new FileReader(fKeywordsTwo));

fAuthorsFive = new File(Config.CAMINHO_ADAPTADO + "authors5.txt");
BufferedReader bAuthorsFive = new BufferedReader(new FileReader(fAuthorsFive));
fAuthorsTree = new File(Config.CAMINHO_ADAPTADO + "authors3.txt");
BufferedReader bAuthorsTree = new BufferedReader(new FileReader(fAuthorsTree));
fAuthorsTwo = new File(Config.CAMINHO_ADAPTADO + "authors2.txt");
BufferedReader bAuthorsTwo = new BufferedReader(new FileReader(fAuthorsTwo));

fAbstractFive = new File(Config.CAMINHO_ADAPTADO + "abstract5.txt");
BufferedReader bAbstractFive = new BufferedReader(new FileReader(fAbstractFive));
fAbstractTree = new File(Config.CAMINHO_ADAPTADO + "abstract3.txt");
BufferedReader bAbstractTree = new BufferedReader(new FileReader(fAbstractTree));
fAbstractTwo = new File(Config.CAMINHO_ADAPTADO + "abstract2.txt");
BufferedReader bAbstractTwo = new BufferedReader(new FileReader(fAbstractTwo));

fReaderCountFive = new File(Config.CAMINHO_ADAPTADO + "reader5.txt");
BufferedReader bReaderCountFive = new BufferedReader(new FileReader(fReaderCountFive));
fReaderCountTree = new File(Config.CAMINHO_ADAPTADO + "reader3.txt");
BufferedReader bReaderCountTree = new BufferedReader(new FileReader(fReaderCountTree));
fReaderCountTwo = new File(Config.CAMINHO_ADAPTADO + "reader2.txt");
BufferedReader bReaderCountTwo = new BufferedReader(new FileReader(fReaderCountTwo));
System.out.println(dadosType.size());
System.out.println(dadosSource.size());
System.out.println(dadosYearDiscretizados.size());
System.out.println(dadosMonthName.size());

for (int i = 0; i < dadosType.size(); i++) {
   dataBf5.append(bTitleFive.readLine() + "," + dadosType.get(i) + "," + dadosSource.get(i) + "," +
dadosYearDiscretizados.get(i) + "," + bKeywordsFive.readLine() + "," + bAuthorsFive.readLine() + "," +
dadosMonthName.get(i) + "," + bAbstractFive.readLine() + "," + bReaderCountFive.readLine() + "\r\n");
   dataBf3.append(bTitleTree.readLine() + "," + dadosType.get(i) + "," + dadosSource.get(i) + "," +
dadosYearDiscretizados.get(i) + "," + bKeywordsTree.readLine() + "," + bAuthorsTree.readLine() + "," +
dadosMonthName.get(i) + "," + bAbstractTree.readLine() + "," + bReaderCountTree.readLine() + "\r\n");
```

```
            dataBf2.append(bTitleTwo.readLine() + "," + dadosType.get(i) + "," + dadosSource.get(i) + "," +
dadosYearDiscretizados.get(i) + "," + bKeywordsTwo.readLine() + "," + bAuthorsTwo.readLine() + "," +
dadosMonthName.get(i) + "," + bAbstractTwo.readLine() + "," + bReaderCountTwo.readLine() + "\r\n");
        }

        dataBf5.close();
        dataBf3.close();
        dataBf2.close();
        System.out.println("Concluido");
    }
}
```