Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Um Estudo sobre a Seleção Automática de Trabalhos em Revisões e Mapeamentos Sistemáticos da Literatura

Gleison de Andrade e Silva

Teresina-PI, 12 de Setembro de 2019

Gleison de Andrade e Silva

Um Estudo sobre a Seleção Automática de Trabalhos em Revisões e Mapeamentos Sistemáticos da Literatura

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Pedro de Alcântara dos Santos Neto

Coorientador: Raimundo Santos Moura

Teresina-PI

12 de Setembro de 2019

Gleison de Andrade e Silva
Um Estudo sobre a Seleção Automática de Trabalhos em Revisões e Mapeamentos Sistemáticos da Literatura/ Gleison de Andrade e Silva. – Teresina-PI, 12 de Setembro de 2019-
74 p. : il. (algumas color.) ; 30 cm.

Orientador: Pedro de Alcântara dos Santos Neto
Coorientador: Raimundo Santos Moura

Dissertação (Mestrado) – Universidade Federal do Piauí – UFPI
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação, 12 de Setembro de 2019.

1. Revisão Sistemática da Literatura. 2. Mapeamento Sistemático da Literatura. 3. Seleção de Estudos. 4. Triagem de Citação. 5. Aprendizagem de Máquina. 6. Mineração de Texto. 7. Processamento de Linguagem Natural. I. Pedro de Alcântara dos Santos Neto. II. Raimundo Santos Moura. III. Universidade Federal do Piauí. IV. Um Estudo sobre a Seleção Automática de Trabalhos em Revisões e Mapeamentos Sistemáticos da Literatura.

CDU 02:141:005.7

"Um Estudo sobre a Seleção Automática de Trabalhos em Revisões e Mapeamentos Sistemáticos da Literatura"

GLEISON DE ANDRADE E SILVA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Natureza da Universidade Federal do Piauí, como parte integrante dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Aprovada por:

Prof. Pedro de Alcântara dos Santos Neto
(Presidente da Banca Examinadora)

Prof. João Paulo Pordeus Gomes
(Examinador Externo à Instituição)

Prof. Raimundo Santos Moura
(Examinador Interno)

Prof. Vinicius Ponte Machado
(Examinador Interno)

Teresina, 12 de setembro de 2019

Aos meus pais Francisco Andrade Silva e Cláudia Regina Soares Silva,
por sempre estarem comigo em todos os momentos.

Agradecimentos

Agradeço a Deus. Pelas dificuldades que me tornaram mais forte para superá-las e assim conquistar meus objetivos.

Agradeço aos meus pais, Francisco Andrade Silva e Cláudia Regina Soares Silva, por todos os seus ensinamentos que me guiaram e me ajudaram em todos os momentos da minha caminhada. A eles devo tudo, são minha fonte de inspiração por sua dedicação com os filhos, sempre lutando para superar os desafios impostos pela vida. Às minhas irmães Suellen e Raquel, pelo companheirismo e apoio nessa árdua jornada.

Agradeço também à toda a minha família: avós, tios, tias, padrinhos, madrinhas, primos e primas. Pelo amor, carinho em todos os momentos, sempre foram a base para a minha formação como pessoa.

Agradeço ao meu orientador, Prof. Dr. Pedro de Alcântara dos Santos Neto, por todos os conselhos, pela paciência e ajuda nesse período. Sou muito grato por sua confiança e dedicação em todos esses anos de trabalho. Foram muitos ensinamentos, que não levo apenas para a vida acadêmica e sim para a vida. Sempre será uma referência como pessoa, professor e pesquisador.

Agradeço ao Prof. Dr. Raimundo Santos Moura, primeiramente por aceitar me coorientar no meio da caminhada, pelos conselhos, paciência e ajuda na reta final dessa jornada. Obrigado pelos ensinamentos, sua ajuda foi de grande valia para a conclusão deste trabalho. Sempre será uma referência na área de Processamento de Linguagem Natural no Piauí.

Agradeço muito à minha namorada Karmonny Cavalcante, pelo amor, carinho e compreensão durante toda a minha caminhada. Sempre esteve comigo nos bons momentos e, principalmente, nos momentos mais difíceis. Sou infinitamente grato por tudo que tem feito por mim e por sua dedicação.

Aos meus amigos que fiz durante a graduação e mestrado. Foram bons momentos de aprendizado e diversão, o que tornou esse longo período muito mais prazeroso para chegar ao objetivo final. Gostaria de agradecer ao Martony Demes, por fornecer os dados necessários para o inicio do meu trabalho, sua ajuda foi fundamental.

À empresa Infoway, pelo meu amadurecimento profissional e principalmente pela oportunidade de ter trabalhado com bons profissionais, em especial a minha equipe H7dra: Catharina, João Paulo, Dayvid, Marcos Vinicius, Paulo Barbosa e Karol Lima.

Aos professores pela paciência, dedicação, partilha de conhecimento e pelos ensinamentos para a vida. Todos deixaram sua marca, estiveram presentes e contribuíram para

minha formação como pessoa.

À Universidade Federal do Piauí e toda sua equipe de funcionários competentes que contribuíram muito para essa realização, em especial àqueles ligados ao Núcleo de Computação de Alto Desempenho.

Agradeço muito a todos que participaram dessa conquista. Foram muitos obstáculos superados com apoio das excelentes pessoas que me cercam, sozinho nunca conseguiria atingir os objetivos tão desejados, muito obrigado a TODOS.

"Inteligência é a habilidade de se adaptar às mudanças."
(Stephen Hawking)

Resumo

Quando se deseja iniciar uma nova pesquisa é essencial que se faça uma análise do atual estado da arte do tema de interesse. No entanto, garantir que essa análise seja completa e justa, sem que nenhum trabalho seja favorecido na análise, é uma tarefa difícil. Esta é a principal justificativa para a realização de uma Revisão Sistemática da Literatura (RSL) e de um Mapeamento Sistemático da Literatura (MSL). Existem diversos motivos para a realização de revisões/mapeamentos, dentre eles pode-se destacar: a necessidade de sumarizar toda evidência empírica sobre determinado tratamento ou tecnologia e identificar *gaps* na área de pesquisa e sugerir pontos de futura investigação. Apesar de seus benefícios, a condução desses trabalhos continua sendo um processo manual e de trabalho intensivo. Uma RSL/MSL requer bastante esforço e ferramentas que apoiam a sua execução são fundamentais para auxiliar em cada etapa, orientando a sua execução, organização, e consequentemente reduzindo o tempo necessário para sua conclusão. Existem esforços para semi automatizar etapas de uma RSL/MSL. Uma dessas abordagens é a aplicação de Aprendizagem de Máquina (AM) junto com a Mineração de Texto (MT) para auxiliar o estágio de Triagem de Citação (TC), conhecida também como Seleção de Estudos (SE). No entanto, nem todas as soluções propostas são transparentes e adequadas. Apesar das primeiras adaptações de RSLs na área de Computação, especialmente em Engenharia de Software (ES), terem ocorrido há mais de uma década, a perspectiva de boas ferramentas de apoio construídas com base em técnicas de MT é promissora, mas sua eficácia não foi totalmente explorada e testada. Este trabalho tem como principal objetivo a proposição de um método que visa reduzir o esforço e tempo de execução de Revisões e Mapeamentos Sistemáticos, com foco na fase de seleção inicial de estudos, na etapa de condução. Esse método utiliza características textuais de artigos, obtidas do título, resumo e palavras-chave e de posse dessas características, aplica-se técnicas de AM para classificar os artigos em aceitos ou rejeitados. Foram realizadas duas avaliações, a primeira para identificar que configurações foram capazes de reduzir a quantidade de artigos a serem lidos, além disso, a redução da perda de artigos relevantes. Na segunda foi realizada uma comparação com uma ferramenta existente na literatura com o mesmo objetivo. Os resultados apontam que com uma pequena perda de estudos relevantes é possível reduzir em até 69% o trabalho realizado na seleção de trabalhos. Além disso, o método proposto obteve resultados de redução de esforço inferiores em relação à ferramenta a qual foi comparada.

Palavras-chaves: Revisão Sistemática da Literatura. Mapeamento Sistemático da Literatura. Seleção de Estudos. Triagem de Citação. Aprendizagem de Máquina. Mineração de Texto. Processamento de Linguagem Natural.

Abstract

When it is desired to start a new research it is essential that an analysis of the current state of the art of the subject of interest is made. However, ensuring that this analysis is complete and fair without any work being favoured in the analysis is a difficult task. This is the main justification for a Systematic Literature Review (SLR) and a Systematic Mapping Literature (SML). There are several reasons for the realization of revisions/mappings, among them we can highlight: the need to summarize all empirical evidence about a particular treatment or technology; identify gaps in the research area and suggest future research points. Despite its benefits, conducting such work remains a manual and labour-intensive process. An SLR/SME requires a lot of effort and tools that support its execution are fundamental to assist in each step, guiding its execution, organization, and consequently reducing the time needed for its completion. There are efforts to semi-automate steps of an SLR/SME. One of these approaches is the application of Machine Learning (ML) along with Text Mining (TM) to semi-automate the Citation Screening (CS) stage, also known as Study Selection (SS). However, not all solutions proposed are transparent and appropriate. Although the first SLR adaptations in the area of Software Engineering (SE) occurred more than a decade ago, the perspective of good support tools built on TM techniques is promising, but its effectiveness has not been fully explored and tested. This work has as main objective the proposal of a method that aims to reduce the effort and execution time of Systematic Reviews and Mappings, focusing on the initial selection phase of studies of the conduction stage. This method uses textual characteristics of articles, obtained from the title, abstract and keywords, and possessing these characteristics apply ML techniques to classify the articles into accepted or rejected. Two evaluations were performed, the first aimed to identify which configurations were able to reduce the quantity of articles to be read, in addition, the reduction of the loss of relevant articles. In the second, a comparison was made with a tool existing in the literature with the same objective.The results show that with a small loss of relevant studies it is possible to reduce the amount of articles to be read by up to 69%. In addition, the proposed method obtained lower effort reduction results in relation to the tool to which it was compared.

Keywords: Systematic Literature Review. Systematic Mapping Literature. Study Selection. Citation Screening. Machine Learning. Text Mining. Natural Language Processing.

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Lista de abreviaturas e siglas

AM	*Aprendizagem de Máquina*
BOW	*Bag of words*
CAT	*Classificação Automática de Texto*
CC	*Ciência da Computação*
CNB	*Complement Naive Bayes*
CFS	*Correlation-based feature subset selection*
EE	*Estudo de Escopos*
ES	*Engenharia de Software*
ESP	*Especificidade*
FN	*Falsos Negativos*
FP	*Falsos Posiitivos*
IA	*Inteligência Artificial*
ID3	*Interative Dichotomiser 3*
LDA	*Latent Dirichlet Allocation*
LOST	*Laboratory of Software Technology*
MEV	*Modelo de Espaço Vetorial*
MNB	*Multinomial Naive Bayes*
MS	*Mapeamento Sistemático*
MSE	*Mapeamento Sistemático de Estudos*
MSL	*Mapeamento Sistemático da Literatura*
MT	*Mineração de Texto*
MVT	*Mineração Visual de Texto*
NB	*Naive Bayes*

NLTK	*Natural Language Toolkit*
PAL	*Patient Active Learning*
PLN	*Processamento de Linguagem Natural*
QP	*Questão de Pesquisa*
RF	*Random Forest*
ROC	*Receiver Operating Characteristic*
RS	*Revisão Sistemática*
RSL	*Revisão Sistemática da Literatura*
RT	*Revisão Terciária*
RTL	*Revisão Terciária da Literatura*
SCAS	*Score Citation Automation Selection*
SE	*Seleção de Estudos*
SMO	*Sequential Minimal Optimization*
SP	*Seleção de Palavras*
SVM	*Support Vector Machine*
TC	*Triagem de Citação*
TF-IDF	*Term Frequency–Inverse Document Frequency*
VN	*Verdadeiros Negativos*
VP	*Verdadeiros Positivos*
WE	*Word Embeddings*
WEKA	*Waikato Environment for Knowledge Analysis*
WSS	*Work Saved over Sampling*

Sumário

Introdução

Contexto e Motivação

Quando se deseja iniciar uma nova pesquisa é essencial que se faça uma análise do atual estado da arte do tema de interesse. No entanto, garantir que essa análise seja completa e justa, de tal forma que todos os trabalhos relevantes sejam identificados, é uma tarefa difícil. Esta é a principal justificativa para a realização de Revisões e Mapeamentos Sistemáticos da Literatura (RSL e MSL). Segundo Barbara e Charters (2007), uma revisão sistemática sintetiza o trabalho existente de uma maneira que seja justa e também vista como justa.

Uma RSL é um método científico capaz identificar, interpretar e sumarizar os trabalhos relevantes para determinada linha de pesquisa, área ou fenômeno de interesse, de forma não tendenciosa e replicável (KITCHENHAM, 2004). Essa metodologia utiliza estudos primários relativos a uma Questão de Pesquisa (QP) com o objetivo específico de integrar/sintetizar as evidências relacionadas a essa questão.

O método teve origem na Medicina e foi desenvolvido para que os médicos pudessem se manter atualizados por meio de um acompanhamento da evolução das publicações científicas (HIGGINS; GREEN et al., 2008). Entretanto, devido a sua eficácia, ele passou a ser aplicado em diversas linhas de pesquisas, inclusive ligadas a Ciência da Computação, como Engenharia de Software, Inteligência Artificial, Arquitetura de Computadores, Processamento de Imagens, dentre outras (BARBARA; CHARTERS, 2007; PETERSEN et al., 2008).

Existem diversos motivos para a realização de revisões, dentre eles pode-se destacar: a necessidade de sumarizar toda evidência empírica sobre determinado tratamento ou tecnologia; identificar *gaps* na área de pesquisa e sugerir pontos de futura investigação (BARBARA; CHARTERS, 2007). Com essa metodologia é possível analisar os trabalhos disponíveis sobre um tema de interesse e, assim, melhorar a fundamentação de novas soluções.

Existem outros tipos de levantamento bibliográfico um deles são os Mapeamentos Sistemáticos da Literatura (MSLs) e as Revisões Terciária da Literatura (RTLs). Todos esses levantamentos bibliográficos possuem uma etapa de seleção de estudos, que é o foco deste trabalho. Por conta disso, quando o termo *revisão* for utilizado, ele estará se referindo a todos os tipos de levantamentos bibliográficos aqui mencionados.

Devido a confiabilidade e benefícios da realização de revisões, seu número vem crescendo cada vez mais, como pode ser percebido por meio da Figura 1, obtida a partir de

uma busca no *Scopus*[1] no título, resumo e palavras chave dos artigos utilizando os termos *"Systematic Literature Review"* ***OR*** *"Systematic Review"* ***OR*** *"Systematic Mapping Studies"* ***OR*** *"Systematic Mapping"* ***OR*** *"Tertiary Literature Review"*. Os dados apresentados no gráfico são para publicações de 2000 à 2019[2], não tendo sido aplicado um filtro por área de pesquisa. Limitando a pesquisa a estudos da área de Ciência da Computação (CC), foram obtidos os dados mostrados na Figura 2. Além disso, as publicações na área da CC estão concentradas no Brasil, como mostra a Figura 3, que foi obtida também na *Scopus* no mesmo período mencionado anteriormente.

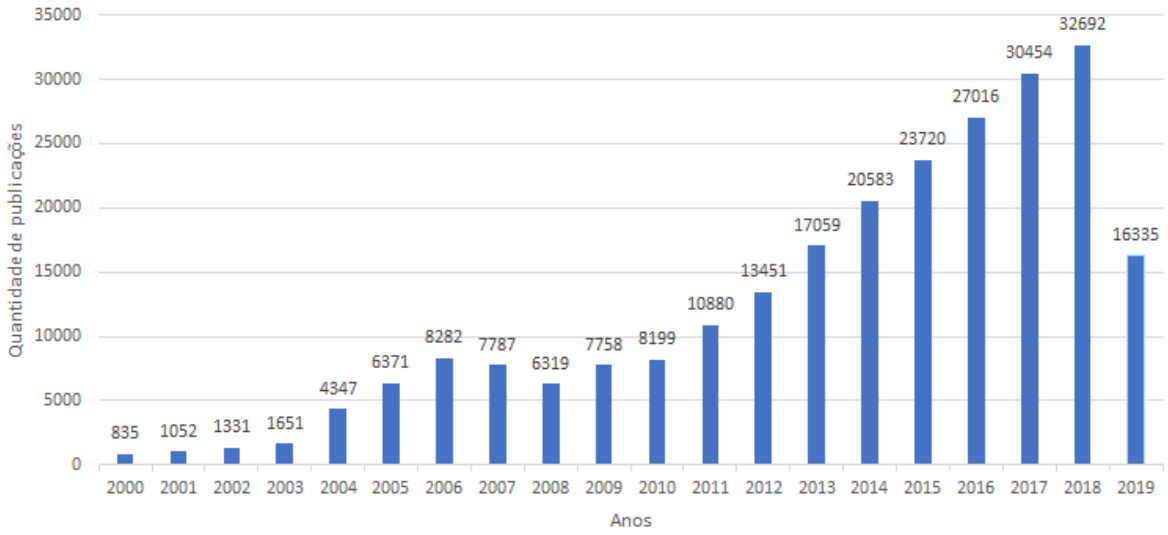

Figura 1 – Número de publicações sobre revisões de 2000 à Junho de 2019, obtidas no *Scopus*.

As vantagens das revisões são: possuem uma metodologia bem definida; permitem uma redução do viés da pesquisa, embora não reduza o viés dos estudos primários; fornecem informações sobre os efeitos de alguns fenômenos em uma ampla gama de configurações e métodos empíricos (BARBARA; CHARTERS, 2007).

A principal desvantagem das revisões é que elas exigem um esforço considerável quando comparadas as revisões tradicionais da literatura. As revisões tradicionais possuem uma temática mais aberta, com questões não bem definidas, sem um protocolo rígido para sua confecção e com fontes de dados não pré-definidas, sendo com frequência menos abrangente (CORDEIRO et al., 2007).

Apesar de seus benefícios, a condução de revisões sistemáticas continua sendo um processo manual e de trabalho intensivo (BRERETON et al., 2007; BABAR; ZHANG, 2009; RIAZ et al., 2010). Uma revisão requer bastante esforço. Por conta disso, ferramentas que apoiam sua execução são fundamentais para auxiliar em cada etapa, orientando a

[1] https://www.scopus.com
[2] Busca realizada em Junho de 2019

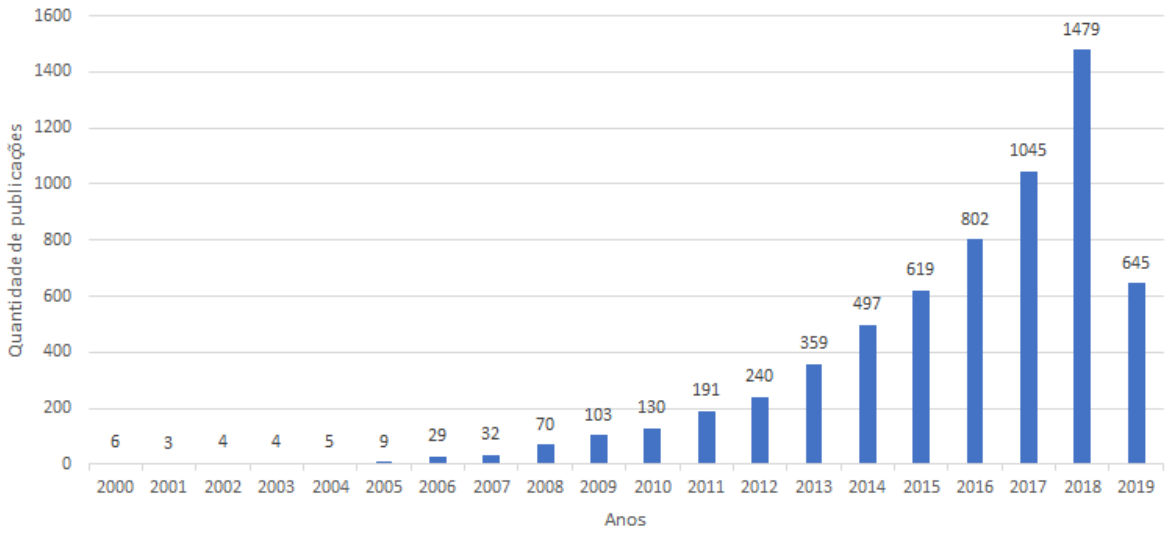

Figura 2 – Número de publicações sobre revisões de 2000 à Junho de 2019 na área da CC, obtidas no *Scopus*.

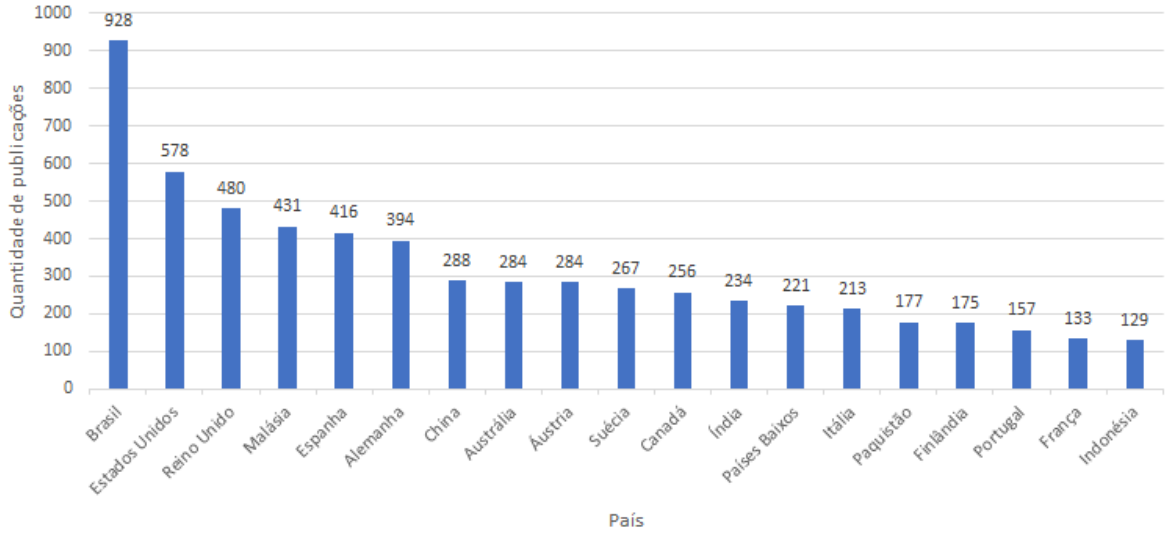

Figura 3 – Número de publicações sobre revisões por país na área da CC de 2000 à Junho de 2019, obtidas no *Scopus*.

sua execução, organização e consequentemente reduzindo o tempo necessário para sua conclusão (BARBARA; CHARTERS, 2007).

Definição do Problema

O grande e crescente número de trabalhos publicados torna a identificação de estudos relevantes de uma área uma tarefa complexa e demorada. Existem esforços para automatizar parte das etapas de uma revisão. Uma dessas abordagens é a aplicação de Aprendizagem de Máquina (AM) junto com a Mineração de Texto (MT), para tentar

automatizar o estágio de Triagem de Citação (TC) conhecida também como Seleção de Estudos (SE). Segundo Olorisade et al. (2016), nem todas as soluções propostas são transparentes e adequadas. Uma solução é considerada transparente quando todas as informações necessárias para sua replicação são divulgadas e sua adequação é provada quando a solução pode ser aplicada em um contexto real.

Apesar das primeiras adaptações de revisões na área de Engenharia de Software (ES) terem ocorridas há mais de uma década, a perspectiva de boas ferramentas de apoio, construídas com base em técnicas de MT, é ainda promissora mas sua eficácia não foi totalmente explorada e testada (FENG; CHIAM; LO, 2017).

Dentre os problemas identificados em Olorisade et al. (2016), relacionados aos algoritmos utilizados para a automação das etapas de revisões, destacam-se: a falta de informações que dificultam a avaliação do processo e a validade estatística da maioria dos estudos; dificuldade para identificar como os trabalhos tratam o *overfitting*, além do impacto causado pelo tamanho dos dados nesse contexto não ter sido analisado.

A vasta gama de diferentes questões exploradas dificultam tirar conclusões sobre as abordagens mais eficazes (O'MARA-EVES et al., 2015). É necessário que os estudos neste campo passem a relatar com mais informações todo o processo executado e assim auxiliar a reprodução independente dos estudos. Embora muitos estudos já tenham sido publicados nessa área, ainda não há progressos concretos (OLORISADE et al., 2016).

Visão Geral da Proposta

Este trabalho tem como principal objetivo propor um método que visa reduzir o esforço e tempo de execução de revisões, com foco na fase de seleção inicial de estudos, na etapa de condução (que será discutida na Seção 1.1.1). O método proposto utiliza características textuais de artigos (obtidas do título, resumo e palavras chave), e de posse dessas características aplicam-se técnicas de Aprendizagem de Máquina para classificar os artigos em duas classes (aceitos ou rejeitados) para o objetivo da pesquisa em questão. O método funciona desta forma, pois na seleção inicial de estudos os pesquisadores selecionam os artigos apenas com a leitura do título, resumo e palavras chave.

Visando alcançar os objetivos deste trabalho, o método foi dividido em quatro passos. O primeiro denominado *Artigos Identificados* é responsável por obter o conjunto de artigos (dados) que serão classificados em aceitos e rejeitados. Na *Fila de Leitura*, é feita uma ordenação dos artigos identificados no passo anterior, essa ordenação define a prioridade de leitura dos artigos no próximo passo. Inicialmente os trabalhos são ordenados aleatoriamente, após a identificação de alguns estudos relevantes o método passa a organizar os artigos por probabilidade de aceitação. Uma vez definidos os trabalhos a serem avaliados, os artigos são classificados pelo revisor em aceitos e rejeitados, esse passo é o de *Classificação*

Manual de Artigos. Por fim, na *Identificação de Artigos Relevantes* são realizados algumas tarefas, primeiro os artigos já avaliados são utilizados para treinar um classificador e ao final os artigos não avaliados são submetidos ao classificador. No final é obtida uma lista de artigos classificados automaticamente, os considerados aceitos são priorizados para leitura no segundo passo do método e então o processo é reiniciado. O critério de parada é quando não existem artigos candidatos a aceitação e o pesquisador decide encerrar. Caso o mesmo deseje continuar o processo de avaliação, é possível realizar a leitura de artigos de forma aleatória, assim como é feita no início da execução do método. O método proposto neste trabalho é apresentado e discutido detalhadamente no Capítulo 3.

Objetivos

O objetivo principal deste trabalho é propor um método para apoiar a etapa de condução em revisões, especificamente focada na redução de esforço na seleção de estudos primários. Associado a esse objetivo, tem-se a questão de pesquisa norteadora do trabalho:

> É possível automatizar a seleção de estudos em revisões, reduzindo a quantidade de artigos a serem lidos e garantindo que a perda de estudos relevantes seja mínima?

Para alcançar o objetivo geral deste trabalho foram definidos alguns objetivos específicos:

- Realizar um estudo das principais técnicas focadas na redução de esforço na etapa de seleção de estudos em revisões da literatura;
- Identificar quais são as melhores técnicas de Processamento de Linguagem Natural (PLN) para apoiar a resolução do problema de seleção de estudos em revisões sistemáticas;
- Identificar qual algoritmo de aprendizagem de máquina possui a melhor performance nos contextos avaliados;
- Propor um método, que utiliza as técnicas de PLN e algoritmos de aprendizagem, para auxiliar a seleção de estudos;
- Avaliar o método proposto em diferentes revisões;
- Realizar uma comparação do método proposto com métodos similares existentes na literatura.

Justificativa

Uma revisão fornece subsídios importantes para uma área de pesquisa. Para sua realização são necessárias ferramentas que auxiliem sua execução. Além da redução de esforço e tempo, ferramentas podem auxiliar no trabalho em equipe, na extração de dados e categorização, sumarização dos resultados, permitindo que o máximo de informações obtidas sejam visualizadas de forma simples.

Os itens citados no parágrafo anterior são resolvidos com facilidade por ferramentas simples, sem a necessidade de implementação de mecanismos mais robustos. Existem ferramentas com o objetivo de suprir essas necessidades. Em Marshall, Brereton e Kitchenham (2014) foram identificadas algumas dessas ferramentas. No entanto, no âmbito do nosso laboratório de pesquisa, denominado LOST[3], existe uma ferramenta que possui os mesmos objetivos das ferramentas identificadas no referido mapeamento. Essa ferramenta é denominada *TheEnd*[4]. As proposições realizadas neste trabalho serão implementadas e disponibilizadas nessa ferramenta em trabalhos futuros.

Outro ponto importante de se ressaltar é a necessidade de mecanismos inteligentes para automatizar parte do processo de execução de revisões. Mesmo que a utilização de ferramentas sem mecanismos inteligentes possibilitem uma redução do esforço e tempo, ainda existem limitações que precisam ser superadas. Por exemplo, a redução de trabalhos a serem lidos se limita em identificar artigos duplicados ou artigos que não possuem autores ou resumo.

A etapa de condução, que inclui a seleção de trabalhos, ainda é considerada a mais trabalhosa e propensa a erros. A seleção de estudos é normalmente a parte mais demorada de uma revisão, porém, existem outros passos que também demandam muito esforço, como extração e sumarização de dados. Por conta disso, mecanismos inteligentes são necessários para reduzir o esforço associado (BARBARA; CHARTERS, 2007).

Existem trabalhos focados na redução de esforço ligado à condução de revisões, que foram identificados nos trabalhos de Marshall e Brereton (2013), O'Mara-Eves et al. (2015), Olorisade et al. (2016), Feng, Chiam e Lo (2017). Esses trabalhos identificaram alguns problemas que justificam a realização de novas pesquisas. Focando apenas em trabalhos que tem como objetivo reduzir o esforço na seleção de estudos, pode-se destacar os seguintes problemas:

- A maioria dos trabalhos se concentra na Medicina. Assim, replicações em outras áreas são necessárias;

[3] Lab Of Software Technology
[4] Website TheEnd – https://easii.ufpi.br/theend/

- Detalhes necessários para replicação dos estudos são omitidos em alguns casos, dificultando a realização de estudos que comparem o desempenho de diversas abordagens;
- Poucos dos métodos propostos estão disponíveis de forma aplicada, integrando ferramentas para que a comunidade utilize;
- Algumas abordagens exigem conhecimentos de computação para serem utilizadas, dificultando seu uso por pessoas de outras áreas que não seja a computação.

Dado os fatos anteriormente comentados, é necessário frisar que o método proposto neste trabalho não visa substituir ou reimplementar ferramentas e métodos existentes na literatura. O objetivo principal é fornecer uma alternativa para reduzir o esforço necessário na seleção de estudos, que no futuro será implementado em um ferramenta de acesso livre. Além disso, o método será disponibilizado de forma que não seja necessário conhecimentos adicionais para o seu uso.

Estrutura do Trabalho

Além da introdução apresentada anteriormente e que aborda os principais aspectos deste trabalho, que são: contexto, motivação, objetivos gerais e específicos e as principais contribuições deste estudo, este trabalho está dividido em 5 capítulos.

No Capítulo 1 são descritos alguns dos conceitos fundamentais para uma boa compreensão do trabalho. O Capítulo 2 descreve dois Mapeamentos Sistemáticos e duas Revisões Sistemáticas que têm por objetivo a identificação, análise e sumarização das pesquisas disponíveis na literatura, referentes à automatização das etapas de RSLs utilizando mineração de texto e ferramentas que apoiam a sua execução.

No Capítulo 3 é apresentado o método proposto para auxiliar na redução de esforço na seleção de estudos. No Capítulo 4 são discutidas duas avaliações que foram realizadas, a primeira que visa identificar a melhor configuração do método e na segunda é realizada uma avaliação que compara o método proposto com uma ferramenta existente na literatura. Finalmente, Capítulo 5 apresenta as conclusões deste trabalho, assim como desafios e limitações do método proposto e as perspectivas para trabalhos futuros.

1 Fundamentação Teórica

Neste capítulo serão apresentados os conceitos necessários para o entendimento do presente trabalho. Primeiro são apresentadas as revisões sistemáticas na Seção 1.1, como o foco deste estudo é a etapa de condução, apenas ela é discutida em detalhes. Além das revisões outros estudos sistemáticos são importantes e serão discutidos na Seção 1.2. A mineração de texto foi utilizada para resolver o problema da demora na seleção inicial de artigos em estudos sistemáticos, na Seção 1.3 os conceitos de MT focados na classificação de documentos são discutidos. Por fim, os conceitos de aprendizagem de máquina são abordados na Seção 1.4, além disso, o processo de seleção dos algoritmos e os que foram utilizados neste estudos são explanados.

1.1 Revisão Sistemática da Literatura

Revisão Sistemática da Literatura (RSL), por vezes referenciada somente como Revisão Sistemática (RS), é um método capaz de identificar, interpretar e sumarizar os trabalhos relevantes para um determinado tópico de pesquisa, área ou fenômeno de interesse de forma não tendenciosa e replicável (BARBARA; CHARTERS, 2007). Esse método gera estudos secundários, ou seja, estudos que têm o objetivo de produzir comparações sistemáticas a partir de um conjunto de estudos primários (aqueles que investigam uma questão de pesquisa específica, por exemplo, experimentos, estudos de caso, etc.), cientificamente selecionados (BIOLCHINI et al., 2007).

Em geral, uma Revisão Sistemática é composta por três etapas: Planejamento, Condução e Publicação dos Resultados. Em Biolchini et al. (2007) foram definidas as etapas de execução de uma RS, com base em suas experiências e em outros estudos de referência. Isso pode ser resumido conforme Figura 4.A conclusão ou não da etapa é definida pelos termos aprovado ou recusado, quando todos os participantes da pesquisa entram em consenso é dada como aprovada a etapa.

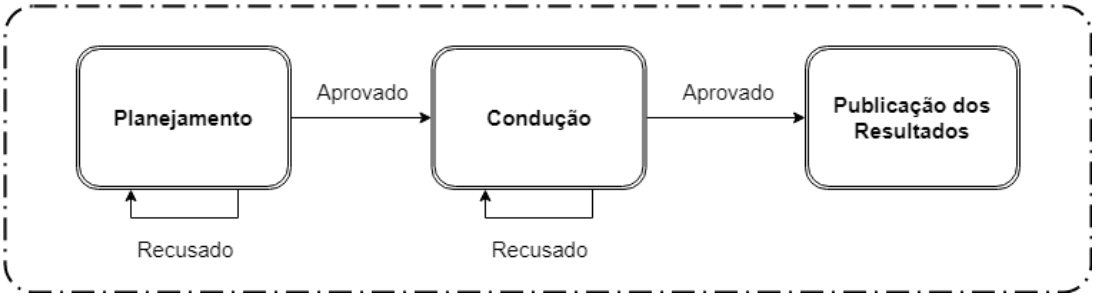

Figura 4 – Processo de execução de Revisões Sistemáticas da Literatura segundo Biolchini et al. (2007).

1. Os estágios associados a etapa de planejamento são:
    - Identificação da necessidade de uma revisão;
    - Comissionamento da revisão (quando necessário);
    - Especificação da(s) questão(ões) de pesquisa;
    - Desenvolvimento do protocolo;
    - Avaliação do protocolo.
2. Em sua condução uma Revisão Sistemática possui os seguintes passos:
    - Identificação dos estudos;
    - Seleção dos estudos primários;
    - Avaliação da qualidade dos estudos;
    - Extração de dados;
    - Síntese dos dados.
3. E por fim, em sua etapa de publicação dos resultados, é necessária a realização dos seguintes itens:
    - Especificação dos mecanismos de disseminação;
    - Formatação do relatório principal;
    - Avaliação do relatório.

Dentre os estágios citados anteriormente, alguns deles não são obrigatórios. O *Comissionamento da revisão*, só ocorre caso a revisão esteja sendo feita em uma base comercial, *Avaliação do protocolo* e *Avaliação do relatório* são opcionais e dependem dos procedimentos que foram definidos para garantir a qualidade da revisão (BARBARA; CHARTERS, 2007).

Como o foco deste trabalho é na seleção inicial de estudos, optou-se por focar somente na etapa de condução. A seguir, discute-se detalhadamente essa etapa. O trabalho de Barbara e Charters (2007), que detalha a aplicação dessa metodologia de pesquisa na área de Engenharia de Software será utilizado como referência principal. Será utilizado como exemplo para explicar todos os passos realizados na etapa de condução o trabalho de Braga et al. (2015).

1.1.1 Condução

Na etapa de condução os pesquisadores devem por em prática as definições construídas no planejamento. Essa etapa requer um maior esforço por parte dos pesquisadores,

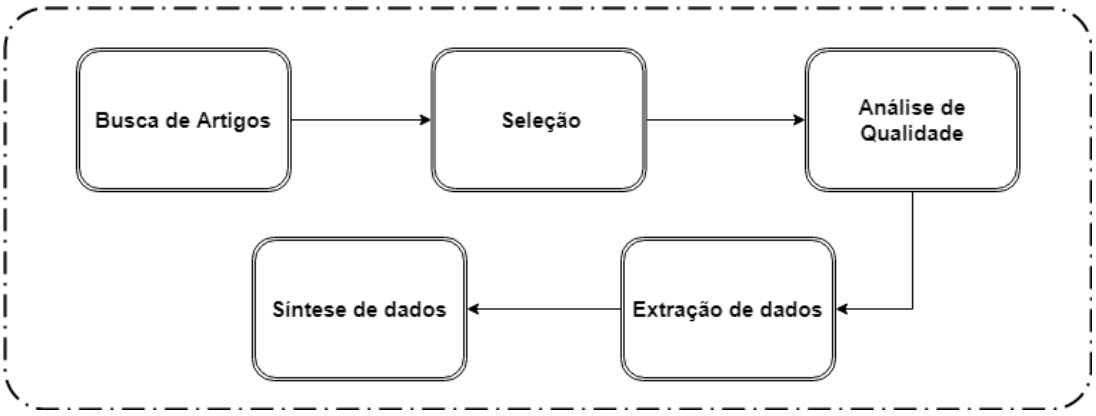

Figura 5 – Fases da etapa de condução.

pois é nela que de fato ocorre a aplicação do protocolo da Revisão Sistemática. Todas as fases que compõem essa etapa são apresentados na Figura 5.

O primeiro passo da condução é aplicar a *string* de busca nas bases de dados definidas, sempre lembrando que a *string* deve ser adaptada aos padrões de cada base de dados. Assim, obtém-se o conjunto total de estudos primários que devem ser analisados pelos pesquisadores.

Após a obtenção do conjunto de estudos primários, deve-se realizar a seleção dos trabalhos que satisfazem os critérios de inclusão definidos no planejamento. Recomenda-se que a seleção seja feita por mais de um pesquisador, pois reduz distúrbios causados pela opinião pessoal dos pesquisadores no resultado. Essa fase pode ser dividida em quatro passos, que são apresentados na Figura 6 e a seguir são discutidos:

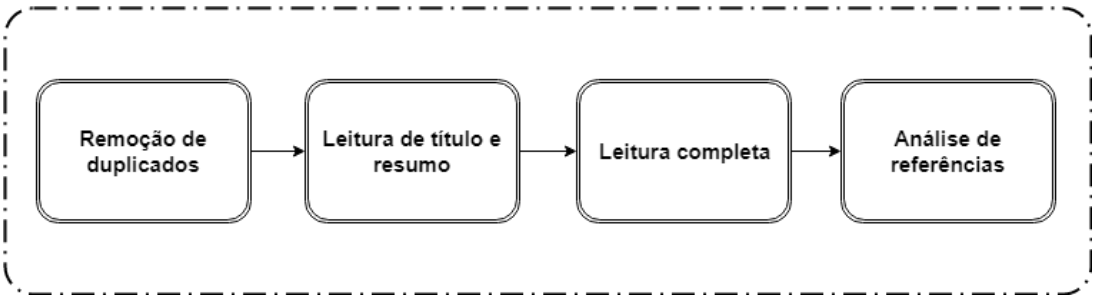

Figura 6 – Passos do processo de seleção de estudos primários.

1. **Remoção de trabalhos duplicados**: existem trabalhos que são encontrados em mais de uma base de dados. As cópias desses trabalhos devem ser removidas do conjunto de estudos primários.

2. **Leitura de título e resumo**: após a remoção dos duplicados cada pesquisador deve realizar a leitura de título e resumo dos estudos primários aplicando os critérios de inclusão/exclusão para aceitar ou rejeitar os trabalhos do conjunto de estudos primários. Ressalta-se que é importante a existência de uma justificativa para

aceitação ou rejeição, por exemplo, "*não atendeu ao critério X*". Após a conclusão dessa fase deve-se realizar uma análise de concordância entre os resultados obtidos pelos diferentes participantes do estudo. Essa análise pode ser feita utilizando o teste *Kappa* (DIXON; JR, 1957) e quanto maior o nível de concordância, maior a confiabilidade dos resultados. As divergências devem ser solucionadas por meio de uma discussão entre os pesquisadores. Recomenda-se também a construção de um relato sobre as decisões tomadas durante esse debate. Como exemplo: "dos 571 trabalhos pré-selecionados, 50 foram aceitos por dois pesquisadores (A e B) e em 8 trabalhos houve discordância quanto a seleção. Esses foram reavaliados por outros dois pesquisadores (C e D) de modo que decidiu-se pela exclusão dos artigos.";

3. **Leitura completa dos trabalhos**: os trabalhos aceitos na fase anterior devem ser lidos por completo pelos pesquisadores para que se verifique se realmente atendem aos critérios de inclusão e para que se inicie o processo de classificação dos trabalhos. Alguns autores (PETERSEN et al., 2008) sugerem classificar os estudos em relação à contribuição do trabalho (SHAW, 2003) e ao tipo de pesquisa (WIERINGA et al., 2006). No nosso exemplo base, imagine que dos 50 trabalhos aceitos após leitura de título e resumo, 13 foram removidos por estarem disponíveis somente na forma de *abstract* ou resumo expandido e 21 foram removidos após a leitura das seções de introdução e conclusão, resultando em 16 trabalhos aceitos;

4. **Análise de Referências**: essa é uma etapa opcional mas que deve ser levada em consideração, pois podem existir trabalhos relevantes nas referências dos trabalhos aceitos e que não foram retornados pela *string* de busca. Esses trabalhos devem ser incluídos no conjunto de estudos e analisados por completo. No nosso exemplo, imagine que realizou-se uma inspeção nas referências bibliográficas dos trabalhos aceitos e decidiu-se pela inclusão de mais 8 artigos.

Após seleção, é necessário avaliar a qualidade dos estudos. É importante definir os instrumentos de qualidade no protocolo e deixar claro de que maneira os dados serão utilizados.

O penúltimo passo a ser executado é a extração dos dados, o conjunto de estudos aceitos representam os trabalhos que respondem às questões de pesquisa. Nesse momento, o pesquisador deve realizar a extração das informações relevantes utilizando os formulários definidos na etapa de planejamento do estudo.

Por último, é realizada a síntese dos dados que envolve a comparação e o resumo dos resultados que foram obtidos dos estudos primários incluídos. A síntese pode ser descritiva (não quantitativa), no entanto, às vezes é possível complementar uma síntese descritiva com um resumo quantitativo.

1.2 Outros tipos de Revisão

Segundo Barbara e Charters (2007), na literatura existem dois tipos de revisões que servem de complemento para às Revisões Sistemáticas da Literatura, que são: Mapeamento Sistemática da Literatura (MSL) que na literatura também é referenciado como Mapeamento Sistemático de Estudos ou Estudo de Escopos (EE), e o segundo é a Revisão Terciária da Literatura (RTL). Em ambos os casos existe uma fase de seleção de estudos, que é o foco deste trabalho.

1.2.1 Mapeamento Sistemático da Literatura

Um Mapeamento Sistemático da Literatura é uma boa alternativa à realização de uma Revisão Sistemática, principalmente quando a sua execução não é viável, que é o caso de existirem poucas evidências ou um determinado tópico de interesse é muito amplo (BARBARA; CHARTERS, 2007). Uma de suas características mais importantes é que são capazes de obter uma visão geral de uma área de pesquisa, pois ele realiza uma análise quantitativa do tópico de interesse (BARBARA; CHARTERS, 2007; PETERSEN et al., 2008).

Apesar de um Mapeamento Sistemático ser uma alternativa à realização de Revisões Sistemáticas, esses dois métodos compartilham algumas semelhanças, sendo similares no que se refere à busca e seleção de estudos, sendo diferentes somente em suas metas e, portanto se diferenciam pelas abordagens utilizadas na análise dos dados (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Enquanto revisões sistemáticas objetivam sintetizar evidências, considerando também a força da evidência, os mapeamentos sistemáticos estão primariamente relacionados à estruturação de uma área de pesquisa (PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

Em um estudo realizado por Kitchenham et al. (2010) foram constatadas algumas diferenças entre o processo de execução de MSLs e RSLs. Essas diferenças identificadas foram em relação as questões de pesquisa, processo de pesquisa, estratégia de pesquisa, avaliação de qualidade e resultados. A Figura 7 mostra as etapas que fazem parte de um Mapeamento Sistemático baseado no trabalho de Petersen et al. (2008).

Como mostra a Figura 7, podemos dividir o MSL em 5 etapas, sendo que cada etapa ao seu final gera algum resultado. As etapas são: a definição das questões de pesquisa, que definem o escopo da pesquisa; a condução, que em seu final gera uma coleção de artigos que serão avaliados; a triagem, que ao final obtém os artigos relevantes para o trabalho; a definição de grupos (*Keywording of Abstracts*), que ao ser aplicada gera um esquema de classificação para os trabalhos; e por fim, a extração de dados, que no final gera os mapas que representam um área.

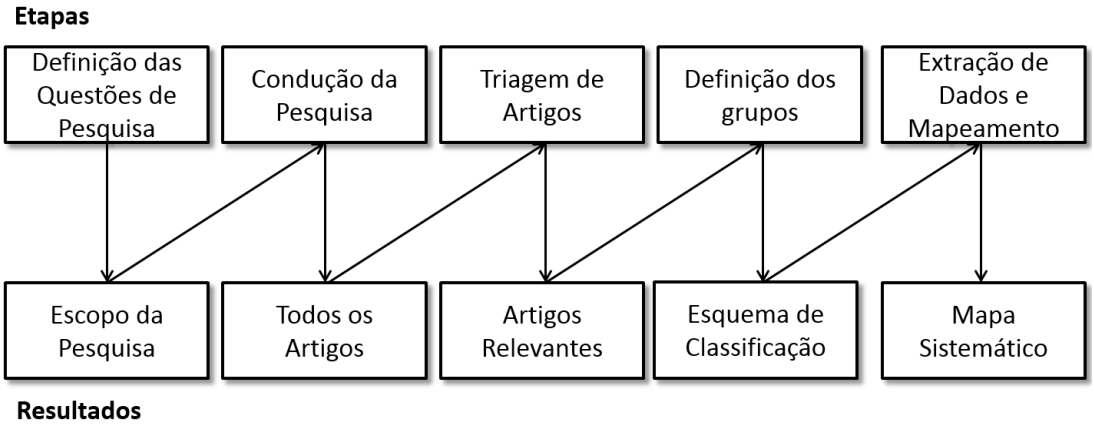

Figura 7 – Etapas Mapeamento Sistemático (adaptadas de Petersen et al. (2008))

1.2.2 Revisão Terciária da Literatura

Outro tipo de revisão existente na literatura são as Revisões Terciárias da Literatura (RTLs). Quando em um determinado domínio existe um grande número de revisões sistemáticas, pode-se realizar uma RTL para responder questões mais amplas. Uma revisão terciária usa exatamente a mesma metodologia que uma revisão sistemática. Sua execução é potencialmente menos intensiva do que conduzir uma nova revisão sistemática de estudos primários, mas isso depende da qualidade das revisões existentes (BARBARA; CHARTERS, 2007).

Com a realização de uma Revisão Terciária se obtém um estudo terciário, pois ele sintetiza estudos secundários. Esses estudos são raros e consequentemente carece de exemplos na literatura. Em Barbara e Charters (2007) que foi base para escrita desta seção, pouco se fala sobre o assunto e não foram identificados outros trabalhos para auxiliar no desenvolvimento desta seção.

Para um melhor entendimento de como são as RTLs na área de Engenharia de Software, pode-se consultar os dois exemplos a seguir: em Kitchenham et al. (2010) foi realizada uma pesquisa ampla sobre todas as revisões sistemáticas na área de ES; no segundo exemplo, Cruzes e Dybå (2011) foi realizado um estudo, que tinha como o objetivo principal, contribuir para uma melhor compreensão dos desafios na síntese de pesquisas na área de ES.

1.3 Mineração de Texto

A Mineração de Texto (MT), às vezes denominada como mineração de dados de texto ou descoberta de conhecimento a partir de banco de dados textuais, normalmente refere-se ao processo de extrair padrões ou conhecimentos interessantes e não triviais de documentos de texto não estruturados (TAN et al., 1999; FELDMAN; DAGAN, 1995; HEARST, 1997). Tais características fazem com que ela seja considerada uma extensão

da mineração de dados ou descoberta de conhecimento a partir de banco de dados (estruturados) (TAN et al., 1999; FAYYAD et al., 1996; SIMOUDIS, 1996).

A mineração de texto é considerada uma tarefa complexa (mais do que a mineração de dados), pois trabalha com dados não estruturados e confusos (TAN et al., 1999). A mineração de texto é um campo multidisciplinar, que envolve a recuperação de informações, análise de texto, extração de informações, agrupamento, categorização, visualização, aprendizado de máquina e mineração de dados (TAN et al., 1999).

A classificação do documento identifica os padrões subjacentes que distingue os documentos e utiliza essas informações para atribuir cada novo documento a classes conhecidas (JOACHIMS, 1998; SEBASTIANI, 2002; ANANIADOU et al., 2009). Pode-se definir de forma simples que o processo de classificação de textos consiste em receber como entrada um documento d e um conjunto fixo de classes $C = c_1, c_2, ..., c_n$. A saída é determinar a que classe o documento d está semanticamente relacionado (JURAFSKY, 2000; SEBASTIANI, 2002; AGGARWAL; ZHAI, 2012). Os modelos de classificação podem ser divididos em *single-label* ou *multi-label* (ALY, 2005). No primeiro, cada documento só pode está associado a uma classe, já no segundo ele pode ser associado a uma ou mais classes (CERRI; CARVALHO; FREITAS, 2011).

A MT além de ser associada aos métodos e técnicas utilizadas na Aprendizagem de Máquina (AM) e Estatística, em determinadas situações, pode ser também ligada à Extração e Recuperação de informação, ao Processamento de Linguagem Natural (PLN), ou ainda, ao processo que conjuga todas estas áreas (HOTHO; NÜRNBERGER; PAASS, 2005; GOMES, 2013).

Em Feng, Chiam e Lo (2017) foi realizada uma RSL com o objetivo de identificar/classificar técnicas e ferramentas de MT que auxiliam as atividades de revisões sistemáticas. Entre as informações obtidas no estudo, pode-se destacar uma sumarização das principais técnicas utilizadas de acordo com o objetivos de cada estudo analisado. A maioria dos trabalhos são focados na classificação de textos nos estágios inicias da pesquisa e seleção de estudos, essa classificação é utilizada para distinguir artigos relevantes dos irrelevantes.

A classificação de textos é constituída pelas seguintes etapas: coleta de dados, processamento, indexação, seleção de palavras, classificação e a análise dos resultados (KORDE; MAHENDER, 2012). A Figura 8 apresenta as etapas do processo de classificação de texto.

Neste trabalho serão utilizadas especificamente técnicas de mineração de texto para a classificação de documentos. As técnicas de MT são utilizadas juntamente com técnicas de PLN e AM. Nas próximas seções serão apresentadas as técnicas utilizadas em cada etapa do processo de classificação.

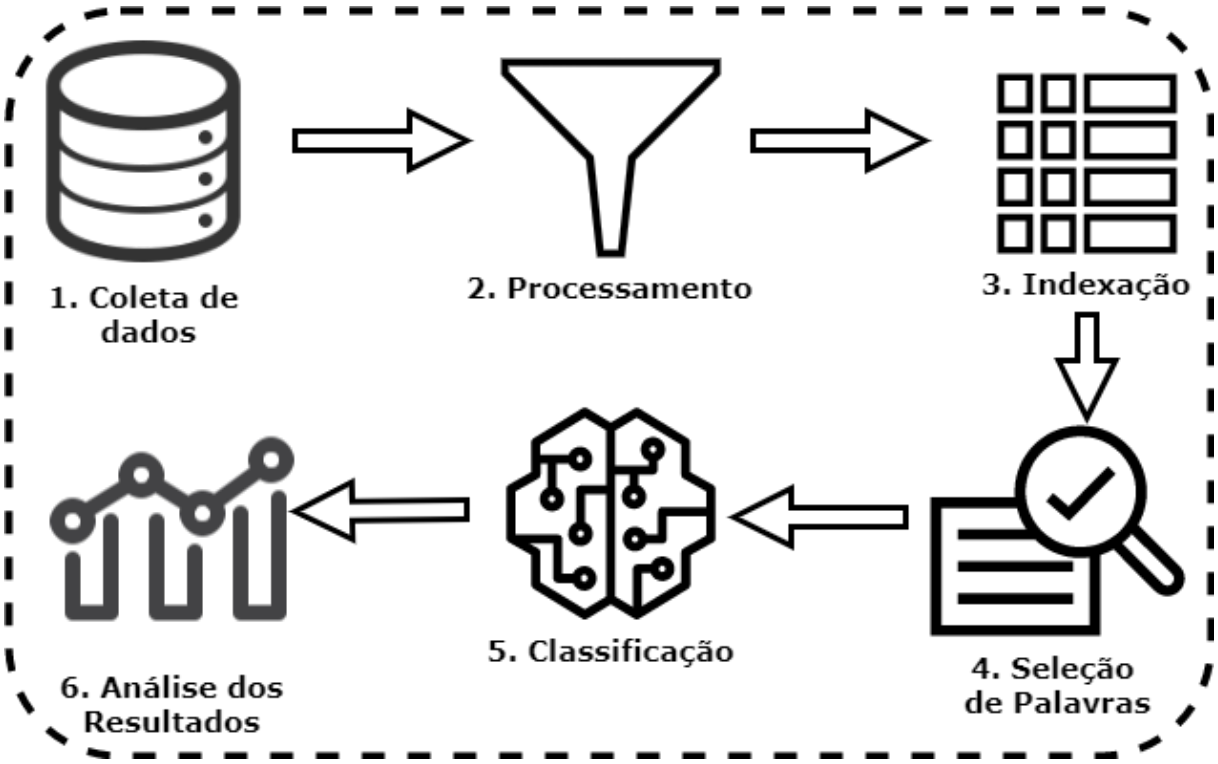

Figura 8 – Processo de classificação de documentos, obtida em Korde e Mahender (2012).

1.3.1 Coleta de dados

A primeira etapa do processo de classificação consiste em extrair textos de diversas fontes de dados Korde e Mahender (2012). Existem diversas fontes de dados textuais, tais como e-mails, textos em banco de dados, páginas web, entre outros. Neste trabalho, os dados são obtidos de arquivos no formato BibTex[5]. Este formato de arquivo possui várias informações sobre um artigo, neste trabalho foram extraídos os dados do título, resumo e palavras chaves.

1.3.2 Processamento

É necessário preparar os dados obtidos na etapa de coleta, removendo o que for desnecessário para o entendimento do texto e preparando o mesmo para a análise que será realizada. Esse processo é feito utilizando mecanismos de tokenização, *stemming*, e outras técnicas de processamento (JURAFSKY, 2000).

Na tokenização são extraídos os *tokens* (palavras) dos textos de cada documento. Isso significa quebrar o texto em uma lista de palavras. Esses *tokens* podem ser formados por mais de uma palavra, quando formado apenas por uma é denominado unigrama. Quando formado por duas palavras, é bigrama, e assim por diante (TAN; WANG; LEE, 2002). Após a geração dos *tokens*, normalmente são removidos todos os caracteres especiais e números, mantendo assim só palavras. A remoção de *stopwords* consiste em retirar os

[5] Site BibTex – <http://www.bibtex.org/>

tokens que são considerados irrelevantes para a análise textual, por não acrescentarem significado ao texto. São exemplos de *stopwords*: a, e, as, os, para, com, sem, etc.

A tokenização e remoção de *stopwords* podem ser realizadas com o auxílio de ferramentas gratuitas. O *Natural Language Toolkit* (NLTK[6]) é uma plataforma líder para a criação de programas em *Python* que trabalham com dados em linguagem humana. A Figura 9 mostra um exemplo de uso dessas funções no NLTK.

```
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords

tokenizer = RegexpTokenizer('[a-z]\w+')
stops = set(stopwords.words("english"))

def pre_process(text):
    tokens = tokenizer.tokenize(text.lower())
    tokens = [w for w in tokens if not w in stops]
    return(" ".join(tokens))

texts = ["You can't connect-me the dots looking forward; you can only connect them looking backward.",
"So you have to trust that the dots will somehow connect in your future.",
"You have to trust in something; your gut, destiny, life, karma, whatever.",
"This approach has never let me down, and it has made all the difference in my life."]

for text in texts:
    print(pre_process(text))
```

```
connect dots looking forward connect looking backward
trust dots somehow connect future
trust something gut destiny life karma whatever
approach never let made difference life
```

Figura 9 – Exemplo de código e resultado obtido com a aplicação da tokenização e remoção de *stop words* com o NLTK.

Os *Stemmers* são analisadores morfológicos que associam variantes de um mesmo termo com sua forma raiz, chamada radical (CHAVES, 2003). A raiz resultante do processo de *stemming* não é necessariamente idêntica ao radical linguístico, mas servirá como uma denotação miníma não ambígua do termo. O *Stemming* consiste em reduzir as palavras ao mesmo *stem*, por meio da retirada dos prefixos e sufixos, permanecendo apenas um radical (CHAVES, 2003). Desse modo, palavras como documentação e documentos seriam ambas transformadas em document, possibilitando a geração de um menor número de palavras e como consequência reduzindo a dimensão dos vetores de palavras.

Existem várias implementações de *stemmers*, neste trabalho foi utilizada a implementação feita em Jones (1997) e que está disponível no NLTK. A Figura 10 apresenta um exemplo[7] de uso do *PorterStemmer*[8].

[6] Site NLTK – <https://www.nltk.org/>
[7] Exemplo retirado de <http://www.nltk.org/howto/stem.html>
[8] PorterStemmer – <https://www.nltk.org/_modules/nltk/stem/porter.html>

```
from nltk.stem.porter import *
stemmer = PorterStemmer()

plurals = ['caresses', 'flies', 'dies', 'mules', 'denied', 'died',
'agreed', 'owned', 'humbled', 'sized', 'meeting', 'stating', 'siezing',
'itemization', 'sensational', 'traditional', 'reference', 'colonizer',
'plotted']

singles = [stemmer.stem(plural) for plural in plurals]

print(' '.join(singles))
```

```
caress fli die mule deni die agre own humbl size meet state siez item sensat tradit refer
colon plot
```

Figura 10 – Exemplo de código e resultado obtido com *PorterStemmer* implementado no NLTK.

1.3.3 Indexação

É comum no processamento de documentos a aplicação de técnicas que visam facilitar a sua manipulação, transformando o texto completo em um vetor de documento Korde e Mahender (2012). Um modelo de representação de palavras que é comum em aplicações de PLN é a *Bag Of Words* (BOW), que é uma representação matricial desse conjunto de termos. Esse modelo possuí algumas limitações, que são: alta dimensionalidade, perda de correlação entre palavras adjacentes e perda do relacionamento semântico entre os termos de um documento Korde e Mahender (2012). Na Seção 1.3.3.1 é discutido sobre a BOW e os tipos que foram utilizados neste trabalho.

1.3.3.1 Bag of Words

A Bag of Words é um modelo matricial de representação de palavras, que visa auxiliar na comparação e avaliação da semelhança entre documentos. A BOW é um método comumente utilizado para preparar texto que serão utilizados como entrada nos algoritmos de Aprendizagem de Máquina. A BOW pode ser construída de várias formas diferentes, neste estudo serão utilizados três. A primeira é representando a ocorrência ou não de uma palavra (*booleana*), a segunda com a quantidade de vezes que as palavras ocorrem em um documento, denominada *Term Frequency* (TF), e por fim, utilizando a*Term Frequency–Inverse Document Frequency* (TF-IDF).

Para facilitar o entendimento de cada tipo de BOW, considere os textos a seguir como exemplos de entrada:

E1 O cachorro é inimigo do gato. O cachorro adora osso.

E2 O gato é inimigo do rato. O gato adora peixe.

E3 O rato é amigo do cachorro. O rato adora queijo.

E4 O cachorro, rato e gato são animais.

A BOW é representada no formato de uma matriz. Na primeira linha ficam todas as palavras identificadas em ordem alfabética. Cada palavra está associada a uma coluna. Cada linha representa uma instância, que neste caso é cada uma das frases apresentadas acima. Por fim, o valor que se encontra no cruzamento entre linha e coluna é de acordo com o tipo de BOW, cada um deles será explicado em detalhes.

1. **Booleana**: A representação *booleana* é a mais simples. Quando uma palavra ocorre em um determinada instância, seu valor é 1, caso contrario é 0. Ambas as classes mencionadas anteriormente possuem um atributo *binary* que por padrão é desativado, quando ativo a saída final é um vetor binário.

   A Tabela 1 mostra um exemplo de BOW do tipo *booleana*. Na tabela pode-se perceber que os valores são sempre iguais a 1 ou 0, independente da quantidade de ocorrências de uma determinada palavra.

Tabela 1 – Exemplo de bag of words *booleana*.

	ador	**amig**	**animal**	**cachorr**	**gat**	**inimig**	**oss**	**peix**	**queij**	**rat**
E1	1	0	0	1	1	1	1	0	0	0
E2	1	0	0	0	1	1	0	1	0	1
E3	1	1	0	1	0	0	0	0	1	1
E4	0	0	1	1	1	0	0	0	0	1

2. ***Term Frequency***: A frequência de ocorrências de uma palavra em um documento é também simples. A principal diferença em relação a *booleana*, é que quando uma palavra ocorre para uma determina instância, ao invés do valor ser igual a 1, é adicionada a quantidade de vezes que a palavra apareceu, nesse caso a frequência é absoluta. A Tabela 2 apresenta um exemplo para esse tipo de BOW. Nela as palavras cachorro, gato e rato possuem valor igual a 2, pois essas palavras ocorreram duas vezes nas instâncias E1, E2 e E3 dos textos, respectivamente.

Tabela 2 – Exemplo de bag of words com *Term Frequency*.

	ador	**amig**	**animal**	**cachorr**	**gat**	**inimig**	**oss**	**peix**	**queij**	**rat**
E1	1	0	0	2	1	1	1	0	0	0
E2	1	0	0	0	2	1	0	1	0	1
E3	1	1	0	1	0	0	0	0	1	2
E4	0	0	1	1	1	0	0	0	0	1

3. **TF-IDF**: a diferença para esse tipo de BOW em relação aos anteriores está no valor que é armazenado para cada tupla instância/palavra. É necessário se entender para que serve o TF-IDF, que visa determinar quais palavras em um *Córpus*[9] de documentos, podem ser mais favoráveis para uso em uma consulta. Como o termo implica, o TF-IDF calcula valores para cada palavra em um documento, sendo esse cálculo feito com a proporção inversa da frequência da palavra em um documento particular, em relação a porcentagem de documentos em que a palavra aparece (RAMOS et al., 2003). Palavras com números altos de TF-IDF implicam em um forte relacionamento com o documento em que aparecem, sugerindo que, se essa palavra aparecesse em uma consulta, o documento poderia ser de interesse para o usuário (RAMOS et al., 2003).

   O TF-IDF funciona da seguinte maneira: dada uma coleção de documentos D e uma palavra w e um determinado documento $d \in D$, podemos obter o TF-IDF pela Equação 1.1. Nela, $f_{w,d}$ é o numero de vezes que a palavra w ocorre em d, $|D|$ é o tamanho do *Córpus*, e $f_{w,D}$ é o número de documentos em D que possuem w (SALTON; BUCKLEY, 1988; BERGER et al., 2000).

$$w_d = f_{w,d} \times \log\left(\frac{1 + |D|}{1 + f_{w,D}}\right) + 1 \tag{1.1}$$

   Os vetores TF-IDF resultantes são então normalizados pela norma euclidiana[10], como mostra a Equação 1.2.

$$v_{norm} = \frac{v}{\|v\|_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}} \tag{1.2}$$

Tabela 3 – Exemplo de bag of words com TF-IDF.

	ador	**amig**	**animal**	**cachorr**	**gat**	**inimig**	**oss**	**peix**	**queij**	**rat**
E1	0,32	0,0	0,0	0,63	0,32	0,39	0,5	0,0	0,0	0,0
E2	0,32	0,0	0,0	0,0	0,63	0,39	0,0	0,5	0,0	0,32
E3	0,3	0,47	0,0	0,3	0,0	0,0	0,0	0,0	0,47	0,61
E4	0,0	0,0	0,67	0,43	0,43	0,0	0,0	0,0	0,0	0,43

1.3.4 Seleção de palavras

Para garantir a efetividade dos classificadores de textos é necessário que após o processamento e indexação se realize uma Seleção de Palavras (SP) (DASGUPTA et al., 2007). A sua aplicação visa identificar um subconjunto de palavras obtidas dos documentos

[9] Córpus (plural corpora) linguístico é o conjunto de textos escritos em uma determinada língua e que serve como base de análise.

[10] Informações obtidas na documentação em <http://bit.ly/2RT2L72>

originais. Com a SP os termos com uma maior pontuação de importância de acordo com alguma medida são mantidos. Uma de suas vantagens é a redução da dimensionalidade que é um problema comum na classificação de textos.

Segundo Bird, Klein e Loper (2009) baseado no que se pretende analisar, define-se qual será à abordagem a ser utilizada. Ela pode ser semântica, que visa identificar a importância das palavras dentro do contexto, utilizando, para isto, as relações morfológicas e sintáticas do idioma. A abordagem estatística é baseada na frequência dos termos encontrados no texto.

Existem diversas técnicas que são utilizadas para a seleção de palavras em revisões. Em Olorisade et al. (2016) foram identificadas essas técnicas, na maioria dos trabalhos foi utilizada a frequência de termos para seleção de palavras. Neste trabalho foram utilizadas abordagens estatísticas nos testes realizados.

Nas próximas seções será realizada uma discussão sobre duas abordagens que foram utilizadas para selecionar as palavras relevantes nos documentos. A primeira foi utilizando a lei de Zipf com dois pontos de cortes definidos por Luhn, na Seção 1.3.4.1 são dados mais detalhes. Na Seção 1.3.4.2 é utilizada uma abordagem que foi implementada como um filtro de seleção de atributos na ferramenta WEKA. A *Waikato Environment for Knowledge Analysis* (WEKA) surgiu pela necessidade percebida de um sistema capaz de unir várias técnicas de Aprendizagem de Máquina (AM), permitindo que pesquisadores tenham um fácil acesso às técnicas avançadas (HALL et al., 2009).

1.3.4.1 Lei de Zipf e Cortes de Luhn

Mesmo após a remoção de *stopwords* que são palavras que não apresentam relevância semântica para a análise de alguns problemas, podem existir outras palavras que também são poucos representativas em um determinado *Córpus* estudado. Em (LUHN, 1958) foi proposto uma forma de identificar tais palavras, utilizando a lei de Zipf (ZIPF, 1949).

Luhn ponderou os termos relevantes para a indexação de documentos que ficam concentrados em uma região delimitada por dois pontos de cortes. Termos acima do limite superior e termos abaixo do limite inferior são considerados pouco discriminantes, por serem muito frequentes ou por serem muito raros. A Figura 11 mostra os limites superior e inferior para a curva de Zipf, segundo a abordagem de Luhn.

Podemos dividir em três regiões, os termos de maior frequência que são termos como artigos, preposições e conjunções que se encontram na região superior. A central que é onde se concentram os termos mais relevantes, que podem ser substantivos, adjetivos e verbos. Por fim, a terceira possui termos menos frequentes, que muitas vezes podem ser considerados como ruídos. Apesar dessas regiões serem de fácil visualização na Figura 11, definir esses pontos de corte não é uma tarefa trivial e a decisão sobre os limites está

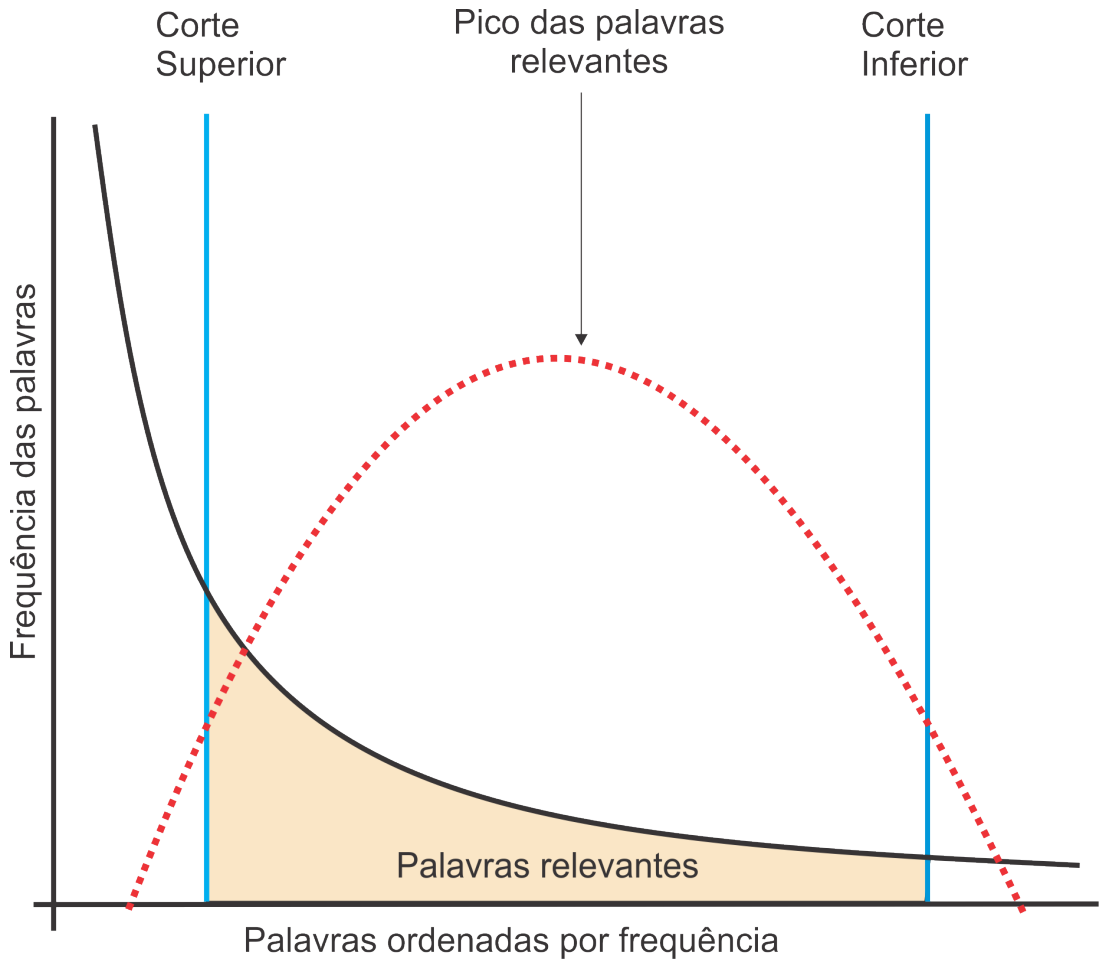

Figura 11 – Lei de Zipf e pontos de corte propostos por Luhn.

associada a um certo grau de arbitrariedade.

Neste trabalho foram feitos testes variando os pontos de cortes. A utilização desta técnica melhorou os resultados obtidos, embora não seja possível definir exatamente o ponto ideal em que se deve aplicar cada um deles.

1.3.4.2 Correlation-based feature subset selection

Para a seleção de atributos a serem usados neste trabalho foi utilizado o filtro *Correlation-based feature subset selection* (CFS) baseado no trabalho de Hall (1998). Esse filtro avalia o valor de um subconjunto de atributos, considerando a capacidade preditiva individual de cada um, juntamente com o grau de redundância entre eles. O CFS é um algoritmo de filtro simples, que classifica subconjuntos de atributos de acordo com uma função de avaliação heurística baseada em correlação. Essa função tenta encontrar subconjuntos que contêm atributos altamente correlacionados com a classe e não correlacionados entre si (HALL, 1998).

Atributos irrelevantes devem ser ignorados, por possuírem baixa correlação com a classe. Os atributos redundantes devem ser selecionados, pois estarão altamente correlacio-

nados a um ou mais atributos restantes. A aceitação de um recurso, depende da medida em que ele prevê classes em áreas do espaço que ainda não foram previstas (HALL, 1998).

O CFS foi testado empiricamente, e os experimentos realizados em conjuntos de dados artificiais mostraram que o CFS pode efetivamente selecionar atributos irrelevantes, redundantes e ruidosos. O CFS seleciona atributos relevantes, desde que eles não interajam fortemente com outros atributos (HALL, 1998).

Neste trabalho os demais filtros de seleção de atributos, não foram capazes de reduzir a quantidade de atributos (palavras). O desempenho dos algoritmos utilizados após seleção de atributos serão discutidos nas avaliações.

1.3.5 Classificação

No aprendizado supervisionado o objetivo é induzir conceitos de exemplos que estão pré-classificados, ou seja, estão rotulados com uma classe conhecida. De acordo com os tipos de valores atribuídos à classe, o problema é conhecido como classificação ou regressão. Neste trabalho o problema a ser resolvido é de classificação, pois os artigos possuem um rótulo conhecido que pode ser aceito ou rejeitado. As características ligadas a cada artigo são utilizadas em um algoritmo de aprendizagem de máquina, visando gerar um classificador para a revisão em questão.

Na literatura existem diversos algoritmos que são utilizados para a classificação de documentos. Abordagens como Árvores de Decisão (*Decision Tree*), *Naive Bayes*, *Random Forest* e SVM são exemplos de modelos de classificação. Na Seção 1.4 são dados detalhes sobre a Aprendizagem de Máquina, como foi realizada a seleção dos algoritmos e um resumo sobre cada um deles.

1.3.6 Análise dos resultados

Nesta seção são apresentados os conceitos relacionados às medidas comumente utilizadas por trabalhos encontrados na literatura para avaliação do desempenho de classificadores. Existe uma exceção que é o WSS (Seção 1.3.6.3). Ela é uma medida desenvolvida para o tipo de problema que este trabalho visa solucionar. Para facilitar o entendimento dos conceitos, são apresentadas as medidas e um exemplo de revisão para à avaliação de desempenho dos classificadores.

1.3.6.1 Matriz de Confusão

A matriz de confusão nada mais é que uma tabela para organização dos resultados obtidos da classificação realizada pelos algoritmos. Seus dados são preenchidos com o objetivo de comparar a classificação do algoritmo com o resultado esperado. Na Tabela 4 é apresentado um exemplo de uma matriz de confusão para problemas binários.

Tabela 4 – Exemplo de uma matriz de confusão para problemas binários.

		Real	
		x	y
Previsto	x	Verdadeiros Positivos	Falsos Positivos
	y	Falsos Negativos	Verdadeiros Negativos

Os valores de x e y representam as classes hipotéticas do problema. Os valores para **Verdadeiros Positivos (VP)** indicam a quantidade de elementos da classe x que realmente foram classificados como x. Os **Falsos Positivos (FP)** indicam a quantidade de elementos da classe y que foram classificados como x. Já os **Falsos Negativos (FN)** indicam a quantidade de elementos da x, mas que foram classificados como y. Finalmente, os valores para **Verdadeiros Negativos (VN)** representam os elementos da classe y que realmente foram classificados como y.

1.3.6.2 Recall

O *recall* ou sensitividade é a medida que infere a porcentagem de verdadeiros positivos em relação a soma dos elementos da classe. Assim, é possível identificar a proporção de elementos de uma classe x em relação a todos que deveriam ter sido classificados como x. O *recall* pode ser calculado por meio da Equação 1.3.

$$R = \frac{VP}{VP + FN} \tag{1.3}$$

1.3.6.3 Work Saved over Sampling

Em Cohen et al. (2006) foi definida uma nova métrica denominada *Work Saved over Sampling* (WSS). Essa métrica visa contabilizar quanto de trabalho não foi necessário ser realizado, ou seja, o trabalho salvo é a porcentagem de artigos que atendem aos critérios de pesquisa, que os revisores não precisam ler (porque foram filtrados pelo classificador). O cálculo do WSS pode ser realizado por meio da Equação 1.4 no qual N é o número total de amostras no conjunto de teste e R o *recall* da classe de artigos aceitos.

$$WSS = \frac{VN + FN}{N - (1.0 - R)} = \frac{(VN + FN)}{N - 1 + \frac{VP}{(VP+FN)}} \tag{1.4}$$

1.3.6.4 Exemplo de revisão e análise de desempenho

Suponha que uma revisão possui um total de 3000 artigos onde 200 devem ser aceitos e foram utilizados 10% para treinamento, onde 150 são aceitos e o restante rejeitado. Ao aplicar o algoritmo nos 90% restante foi obtida a matriz de confusão da Tabela 5.

Tabela 5 – Exemplo de uma matriz de confusão para uma revisão fictícia.

		Real	
		Aceito	Rejeitado
Previsto	Aceito	48	650
	Rejeitado	02	2000

Primeiro calculamos o *recall* para a classe dos artigos aceitos e obtemos $R_{aceitos} = \frac{48}{48+2} = 0,96$. Agora vamos calcular a medida WSS fixada no *recall* de 96%, assim o seguinte resultado é obtido $WSS@96 = \frac{2000+2}{2700-(1,0-0,96)} = 0,741$. Por fim, para diferenciar o WSS de um cálculo simples de percentual, vamos calcular o percentual dos artigos que não seriam lidos. Lembrando que artigos utilizados para treinamento são considerados como lidos e os classificados como aceitos ainda devem ser lidos para confirmar a sua aceitação. Desse modo a redução de esforço em percentual é calculada da seguinte forma $P_{redução} = \frac{2000+2}{3000} = 66,7\%$.

O WSS considera em seu cálculo somente os artigos do conjunto de teste, quando é necessária a leitura inicial de muitos artigos, o WSS tem uma desvantagem em relação ao percentual de artigos não lidos. Para ilustrar melhor essa afirmação, suponha que no exemplo anterior 50% dos artigos foram lidos e os resultados obtidos foram os seguintes: treinamento 1500 artigos. O valor do *recall* permanece o mesmo, mas o WSS agora passa a ser $WSS@96 = \frac{1150+2}{1500-(1,0-0,96)} = 0,768$ e o percentual de redução $P_{redução} = \frac{1150+2}{3000} = 38,4\%$.

O valor obtido para o WSS foi de 76,8%, quando visto em percentual, a redução real é de 38,4%, gerando uma diferença de quase 40% do valor real. É importante considerar no cálculo de redução todos os artigos, assim um valor real da redução é obtido, para o exemplo apresentado anteriormente, substituindo o valor de $N = 3000$ o resultado obtido é igual a 38,4% que é o mesmo do percentual. Neste trabalho são realizadas variações no tamanho do conjunto de treinamento, por conta disso o WSS foi calculado com N igual ao total de artigos da revisão analisada e com N igual ao tamanho do conjunto de teste, que é referenciado com WSS (Cohen).

1.4 Aprendizagem de Máquina

A Aprendizagem de Máquina (AM) é uma sub-área da Inteligência Artificial (IA), cujo o objetivo é desenvolver sistemas que são capazes de adquirir conhecimento de forma automática, inspirados no aprendizado humano (LUGER, 2004; KULKARNI, 2012). Um sistema capaz de aprender é aquele que se modifica automaticamente no sentido de que ele possa fazer a(s) mesma(s) tarefa(s) sobre um mesmo domínio de conhecimento, de uma maneira cada vez mais eficiente (SIMON, 1983; MITCHELL; LEARNING, 1997).

Aprendizado que ocorre com base em uma classe de exemplos é chamado de

supervisionado. Enquanto aprende, o sistema tem um conjunto de dados conhecidos (rotulados), que são os problemas de classificação. Esse é um dos métodos de aprendizado mais comuns (KULKARNI, 2012; ZHANG; ZHOU, 2014).

Este trabalho visa reduzir o esforço e tempo necessário na etapa de seleção de estudos em revisões, a partir de um conjunto de artigos rotulados (aceito e rejeitado), uso de técnicas de MT e PLN para extrair características textuais e uso de algoritmos de aprendizagem para classificar novos dados (artigos não rotulados). Assim, o problema alvo deste trabalho pode ser definido como um problema de classificação.

Nesta seção foi realizada uma divisão em duas partes. A primeira visa apresentar os algoritmos utilizados na literatura e o processo de escolha dos algoritmos que foram utilizados neste estudo, que se encontra na Seção 1.4.1. Na Seção 1.4.2 são apresentados os conceitos e motivação do uso de cada algoritmo identificado anteriormente.

1.4.1 Identificação dos algoritmos de aprendizagem

Em Olorisade et al. (2016) foi realizada uma sumarização dos algoritmos utilizados para a classificação de texto em revisões na área da Medicina. Foi apresentada a distribuição dos algoritmos com o passar dos anos (2006-2014), como mostra a Tabela 6.

Tabela 6 – Algoritmos de classificação utilizados por ano, obtido em Olorisade et al. (2016).

Algoritmo	**2006**	**2007**	**2008**	**2009**	**2010**	**2011**	**2012**	**2013**	**2014**	**Total**
SVM	1	1	3	1	4	1	4		1	16 (31%)
EvoSVM					1		1			2 (4%)
NB		1			1	1	2		2	7 (14%)
cNB					1	1	1			3 (6%)
KNN						1	1		1	3 (6%)
k-Means						1	1			2 (4%)
DT		1			1					2 (4%)
WAODE					1					1 (2%)
NN	1									1 (2%)
Ensemble	1			2	3	1	2	1	1	11 (22%)
Regression							1			1 (2%)
Rocchio									1	1 (2%)
D. S								1		1 (2%)

Existe um espaço de tempo não coberto por esse trabalho de 2015 à 2019, por conta disso a escolha dos algoritmos a serem utilizados nesse estudo não foi realizada somente observando esses dados. Primeiro foram realizados testes utilizando um *plug-in* da Weka. O AUTO-WEKA[11] é um *plug-in* da ferramenta WEKA, que realiza a seleção de algoritmos e hiper-parâmetros para os algoritmos de classificação e regressão implementados na

[11] Manual está disponível em <http://bit.ly/2YNHgag>

WEKA, ou seja, dado um específico conjunto de dados, o AUTO-WEKA utiliza diferentes configurações de hiper-parâmetros com diversos algoritmos e sugere o método que obteve melhor desempenho de generalização para o problema (THORNTON, 2014; KOTTHOFF et al., 2017).

Em um primeiro momento o processamento foi realizado utilizando a própria WEKA, após alguns testes foram identificados problemas (principal estouro de memória). Visando uma solução, foi utilizado o *Sci-kit*[12] para preparar os dados e reduzir a quantidade de palavras. Resolvido o problema da dimensionalidade, os algoritmos identificados pelo *plug-in* obtiveram um baixo *recall* para a classe dos artigos aceitos, mesmo após modificar as métricas de escolha do melhor algoritmo.

Nessa busca pelos melhores algoritmos, foram incluídos os algoritmos entre os melhores resultados do AUTO-WEKA e que são utilizados na literatura para classificação de texto. Nesse processo, foram obtidos os seguintes algoritmos: *Naive Bayes*, SMO, J48, *Random Forest*. Por conta dos resultados obtidos com *Naive Bayes*, decidiu-se incluir sua outra versão o *Complement Naive Bayes* (presente na relação de algoritmos muito utilizados na medicina). Essa inclusão foi realizada após realização de testes que comprovaram o seu desempenho superior ao *Naive Bayes* para a classificação de artigos.

Na Tabela 6 os algoritmos mais utilizados foram o SVM (31%) e o *Esemble* (22%). Respectivamente o primeiro não foi utilizado, pois existem diversos algoritmos de SVM, em muitos trabalhos os vetores de suporte utilizados não são descritos, e por fim, o *Esemble* é a combinação de vários algoritmos, que torna também difícil a replicação dessas combinações.

1.4.2 Algoritmos de aprendizagem de máquina

Nesta seção serão discutidos os algoritmos de Aprendizado de Máquina utilizados neste trabalho, sendo discutido brevemente sobre o funcionamento de cada um deles. Todos os algoritmos utilizados são uma excelente opção para a resolução de problemas de classificação com dados esparsos. Tais características os tornam uma boa opção para a classificação de texto. No total foram utilizados quatro algoritmos e um *framework*, que são: *Naive Bayes* (Seção 1.4.2.1), *Complement Naive Bayes* (Seção 1.4.2.2), SMO (Seção 1.4.2.3), J48 (Seção 1.4.2.4) e *Random Forest* (Seção 1.4.2.5).

1.4.2.1 Naive Bayes

Naive Bayes (NB) é um algoritmo popular de aprendizado de máquina para classificação, devido à sua simplicidade, alta eficiência computacional e boa precisão de classificação, especialmente para dados de alta dimensão, como textos (WU et al., 2015). Pode-se ver esse classificador como uma forma especializada de rede bayesiana, denominada

[12] Site *Sci-kit* – <http://scikit-learn.org/stable/>

ingênua, pois assume que os atributos são independentes um do outro, dado um valor da classe (JOHN; LANGLEY, 1995; LEWIS, 1998).

Dentro dos modelos de classificação bayesiana, dois são mais conhecidos: o modelo binário e o modelo multinomial. No modelo binário, também chamado de básico, cada atributo pode indicar ou não a ocorrência de um evento no documento. O modelo multinomial trabalha com o número de vezes que cada evento ocorre no documento.

Apesar da possibilidade do algoritmo perder performance de classificação, quando os atributos são condicionalmente dependentes, existem estudos comprovando que se é possível obter classificadores de qualidade baseados no NB (HAN et al., 2014). O *Naive Bayes* é um algoritmo de aprendizado que é frequentemente utilizado para lidar com problemas de classificação de texto (KIBRIYA et al., 2004).

1.4.2.2 Complement Naive Bayes

O algoritmo *Complement Naive Bayes* (CNB) é uma adaptação do *Multinomial Naive Bayes* (MNB) (MCCALLUM; NIGAM et al., 1998) que é um algoritmo que se adéqua bem em conjuntos de dados desequilibrados. O CNB utiliza estatísticas do complemento de cada classe para calcular os pesos do modelo. Seus desenvolvedores mostram empiricamente que as estimativas dos parâmetros para o CNB são mais estáveis quando comparadas às obtidas com MNB nas tarefas de classificação de texto (RENNIE et al., 2003).

1.4.2.3 Sequential Minimal Optimization

Sequential Minimal Optimization (SMO) é um *framework* de treinamento aprimorado para SVMs que foi desenvolvido por Platt (1998). As máquina de vetores de suporte (do inglês: *Support Vector Machine* - SVM), possuem um bom desempenho na generalização de uma ampla variedade de problemas, como o reconhecimento de caracteres manuscritos, detecção de faces, detecção de pedestres e categorização de textos (PLATT, 1998; PLATT, 1999). Segundo Platt (1998) ao contrário de outros métodos, o SMO resolve o menor problema possível de otimização em cada etapa. Para os problemas de Programação Quadrática (PQ) em SVM, o menor problema de otimização possível envolve dois multiplicadores de Lagrange. A Programação Quadrática (QP) é o processo de resolver um tipo especial de problema de otimização matemática. Especificamente, um problema de otimização quadrática (linearmente restrito), isto é, o problema de otimizar (minimizar ou maximizar) uma função quadrática de várias variáveis, que estão sujeitas a restrições sobre essas variáveis. A programação quadrática é um tipo particular de programação não linear. A vantagem do SMO reside no fato de que a solução para dois multiplicadores de Lagrange pode ser feita analiticamente.

Por fim, Platt (1998) afirma que o SMO funciona bem para SVMs com entradas esparsas, mesmo para SVMs não lineares, porque o tempo de computação do kernel pode

ser reduzido, tais características o tornam uma boa opção de algoritmo para a classificação de texto.

1.4.2.4 J48

Algoritmos para a construção de árvores de decisão estão entre os mais conhecidos e amplamente utilizados de todos os métodos de aprendizado de máquina (QUINLAN, 2014). O algoritmo C4.5 de árvore de decisão é uma extensão do *Interative Dichotomiser 3* (ID3), que foi implementado por Quinlan (1986). O algoritmo C4.5 gera como saída uma árvore de decisão, ou equivalentemente um conjunto de regras simbólicas (CLARE; KING, 2001). O uso de regras simbólicas permite que a saída seja interpretada, assim é facilmente comparável com o conhecimento existente, outros métodos de aprendizagem de máquina como redes neurais ou máquina de vetor de suporte, não são capazes de gerar uma saída de fácil interpretação (CLARE; KING, 2001).

Na ferramenta WEKA, o J48 é uma implementação Java de código aberto do algoritmo C4.5 (KAUR; CHHABRA, 2014). Essa ferramenta fornece várias opções associadas à poda de árvores. As opções que são disponibilizadas pela ferramenta, auxiliam os seus utilizadores na criação de um sistema de aprendizado com uma alta flexibilidade e precisão (KAUR; CHHABRA, 2014).

1.4.2.5 Random Forest

O Algoritmo *Random Forest* (RF) assim como o C4.5 é um algoritmo do tipo simbolista. O RF é um classificador que consiste em uma coleção de árvores $h(x, k), k = 1, ...,$ no qual k são vetores aleatórios independentes e igualmente distribuídos, cada árvore vota na classe mais popular na entrada x (BREIMAN, 2001).

O algoritmo proposto por Breiman (2001), além de construir cada árvore utilizando uma amostra diferente de dados (*bootstrap* com reposição), o RF altera a forma que as árvores de classificação ou regressão são construídas. Normalmente em árvores, cada nó é dividido usando o melhor entre um subconjunto de preditores escolhidos aleatoriamente nesse nó (LIAW; WIENER et al., 2002).

1.5 Considerações Finais

Neste capítulo foram introduzidos os conceitos necessário para o entendimento do desta dissertação. Alguns desses conceitos foram apresentados com o uso de ferramentas que foram utilizadas no método com exemplos. No próximo capítulo o foco é nos trabalhos relacionados com o estudo e no estado da arte.

2 Trabalhos Relacionados

Neste capítulo serão discutidos os trabalhos que são focados na automatização da seleção de estudos em Mapeamentos ou Revisões Sistemáticas. Existem estudos que fizeram o levantamento do estado da arte no formato de mapeamentos e revisões sistemáticas sobre o tema. Na Seção 2.1 são apresentados tais estudos. Após discussões sobre os estudos, serão apresentados os principais trabalhos que possuem relação a esta pesquisa (Seção 2.2). Por fim, são apresentadas considerações que ajudam a posicionar este trabalho dentro dessa linha da pesquisa.

2.1 Estado da Arte

Em Marshall e Brereton (2013) foi realizado um mapeamento sobre ferramentas que apoiam a realização de Revisões Sistemáticas na área de Engenharia de Software. Foram utilizadas as bases de dados *Google Scholar*, *ACM DL* e *IEEE Xplore*, além da utilização da técnica *snowballing* (nos artigos incluídos). A pesquisa incluiu trabalhos entre os anos de 2004 à 2012 e foram identificados um total de 21 artigos. Nesse estudo não foi reportado de que forma as ferramentas identificadas automatizam as etapas de uma RSL, apenas foram apresentadas as principais abordagens subjacentes utilizadas nos estudos (Figura 12). Outro ponto, foi a resposta da primeira questão de pesquisa que apresenta as ferramentas identificadas, apenas as de interesse deste estudo serão apresentadas posteriormente. A Tabela 7 apresenta a quantidade de estudos identificados por etapa de uma RSL.

Tabela 7 – Trabalhos por abordagem subjacente obtidos em Marshall e Brereton (2013).

RSL Fases	**RSL Estágios**	**Total**
Planejamento	Identificação da necessidade de realização da revisão	-
	Desenvolvimento do protocolo	-
Condução	Identificação dos estudos	1
	Seleção	5
	Análise de qualidade	-
	Extração de dados	3
	Síntese de dados	4
Publicação da revisão		1
Todas as fases		3

Foi realizado um Mapeamento Sistemático em Olorisade et al. (2016) que tem como objetivo realizar uma análise crítica dos estudos obtidos em uma Revisão Sistemática anterior executada por O'Mara-Eves et al. (2015). Nesse estudo foram definidas questões de pesquisa que se concentram em informações sobre as técnicas Mineração de Texto e como

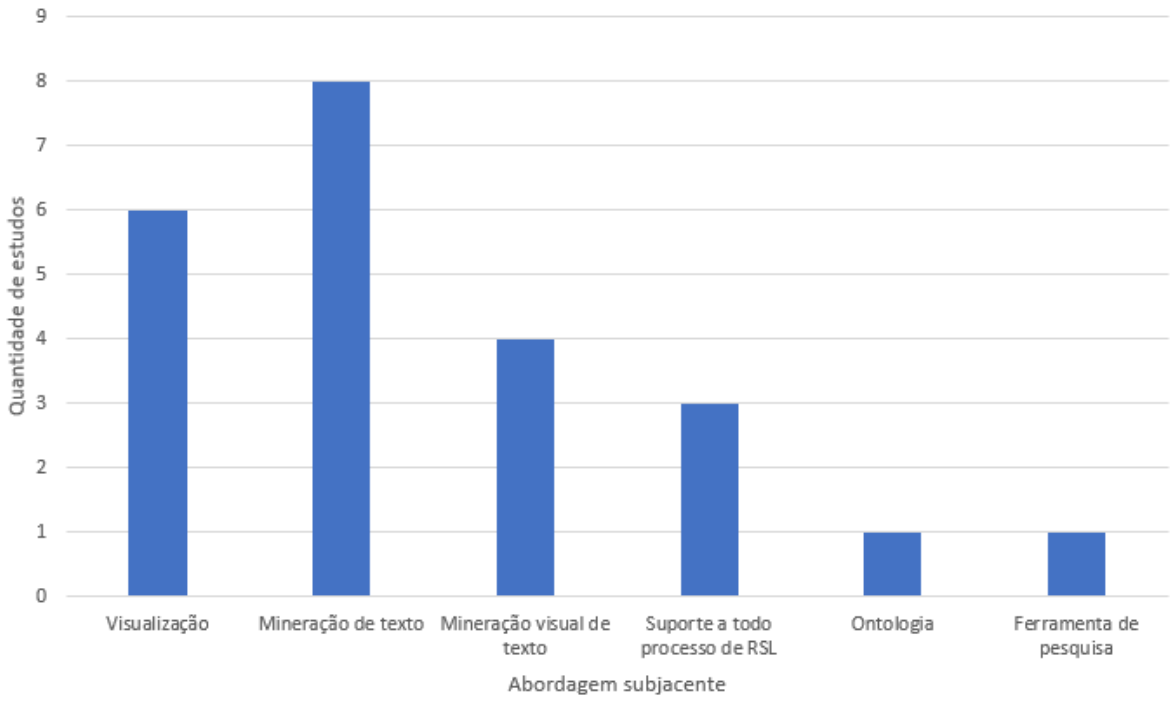

Figura 12 – Abordagens subjacentes identificadas em Marshall e Brereton (2013).

elas foram utilizadas. Em 70% dos casos, os estudos relataram o uso de *frameworks* que implementam algoritmos de AM, utilizando ferramentas como a WEKA, com diferentes configurações, em sua maioria a padrão, sem discutir por que elas são adequadas dentro do contexto que foram aplicadas (OLORISADE et al., 2016).

Em Olorisade et al. (2016), foi identificado que em sua maioria, as revisões utilizadas nos estudos possuem a proporção de 10% de artigos aceitos e 90% rejeitados. Problemas relacionados ao desequilíbrio das classes foram tratados de várias maneiras, em O'Mara-Eves et al. (2015) esse assunto é tratado com mais detalhes. Em sua maioria, os estudos buscam manter a mesma proporção de artigos aceitos e rejeitados no conjunto de teste e treinamento. As medidas de avaliação de classificadores em sua maioria foram utilizadas as mesmas apresentadas na Seção 1.3.6. O *recall* para a classe dos artigos aceitos, na maioria dos trabalhos foi superior a 95%, enquanto a precisão ficou em cerca de 10% em sete trabalhos. Por fim, na terceira e última questão, foi observado que a maioria dos estudos utiliza conjuntos de dados diferentes, dificultando qualquer comparação com os resultados obtidos. Dois estudos utilizaram o mesmo conjunto de dados em seus experimentos, mas não ficou claro se a réplica do conjunto de treinamento e teste em um experimento foi o mesmo no outro (OLORISADE et al., 2016).

Uma Revisão Sistemática da Literatura foi realizada por O'Mara-Eves et al. (2015). Com o objetivo de informar aos revisores o estado atual dos métodos de mineração de texto, como o seu uso reduz a carga de trabalho na fase de triagem (seleção), quais os seus potenciais benefícios e que desafios são enfrentados para a implementação de tais tecnolo-

gias (O'MARA-EVES et al., 2015). Os estudos foram realizados em diversos mecanismos de pesquisa. Foram incluídos estudos de 2005 a dezembro de 2013. Detalhes da estratégia de busca eletrônica, incluindo bancos de dados pesquisados e termos utilizados, podem ser encontrados nos seguintes endereços https://goo.gl/uYP6d4 e https://goo.gl/CtNGK2.

Dentre as informações discutidas em O'Mara-Eves et al. (2015), foram identificados alguns artigos que utilizam algoritmos de classificação com uma abordagem de aprendizado ativo. Esse processo é interativo e a precisão das predições feitas pelos algoritmos é melhorada de acordo com as interações feitas com os revisores. Em todos os casos os autores relataram que os sistemas testados levaram a uma redução na carga de trabalho. No entanto, dada a diversidade de abordagens e a falta de sobreposição (replicação) entre as avaliações, é impossível concluir se uma abordagem é melhor que a outra em termos de desempenho. Nesse trabalho foram identificados os seguintes estudos: Wallace et al. (2010), Jonnalagadda e Petitti (2013), Wallace et al. (2010), Ma (2007), Miwa et al. (2014), Wallace et al. (2012b) e Wallace et al. (2012a).

Outra preocupação do estudo que também e de nosso interesse, é saber quais métodos foram implementados para que possam ser utilizados pela comunidade. Segundo O'Mara-Eves et al. (2015), apenas seis sistemas diferentes (relatados em 12 artigos) estão atualmente disponíveis em alguma ferramenta. Alguns são sistemas de revisão sistemática sob medida, enquanto outros são softwares mais genéricos para a análise preditiva e podem ser usados em RSL. Os sistemas identificados foram *Abstrackr* (WALLACE et al., 2012b; WALLACE et al., 2012a), *EPPI-Reviewer* (SHEMILT et al., 2014; THOMAS; O'MARA-EVES, 2011), *GAPScreener* (YU et al., 2008) e *Revis* (FELIZARDO et al., 2011) e os considerados genéricos são: *Pimiento* (ADEVA et al., 2014) e *RapidMiner* (BEKHUIS; DEMNER-FUSHMAN, 2010; BEKHUIS et al., 2014).

Outro ponto que O'Mara-Eves et al. (2015) apontou em relação aos sistemas, foi que para se utilizar as ferramentas citadas, os revisores necessitam realizar algum treinamento para poder utilizá-las. Ao final desse estudo foram apontadas algumas recomendações, dentre elas as voltadas para a pesquisa, que são:

1. É necessário que métodos diferentes, passem a utilizar o mesmo conjunto de dados, somente assim será possível comparar genuinamente uns com os outros;

2. Para facilitar o que precede, os dados sobre os quais as avaliações são baseadas, devem ser divulgados com maior frequência possível;

3. É necessário analisar a eficácia dos métodos que foram propostos em outras disciplinas. Por exemplo, o campo de Estudo de Desenvolvimento pode ser mais complexo e, portanto, exigir mais da mineração de texto (promovendo mais inovações para superar novos obstáculos).

Em Feng, Chiam e Lo (2017) foi realizada uma RSL com o objetivo de identificar/classificar técnicas e ferramentas de MT que auxiliam as atividades de revisões sistemáticas. Essa revisão se difere da anterior por investigar a adoção de técnicas de MT no domínio da Engenharia de Software. Os estudos buscados estão compreendidos entre janeiro de 2004 à dezembro de 2016. Foram buscados artigos em 6 bases de dados, além da execução de uma busca manual, foram obtidos um total de 4056 artigos, ao final da seleção restaram apenas 32 estudos relevantes para a pesquisa.

No domínio da ES, os pesquisadores propuseram as técnicas de Mineração Visual de Texto (MVT) que combinam técnicas de agrupamento de documentos e visualização de informação. Essas técnicas foram identificados nos seguintes estudos: Malheiros et al. (2007), Felizardo et al. (2010), Felizardo et al. (2014). Foi identificada também uma estratégia de seleção de estudos semiautomática, SCAS (*Score Citation Automation Selection*) que foi proposta pelos pesquisadores com base na técnica de pontuação de 50% e com árvore de decisão, especificamente o algoritmo J48 (Weka) em Octaviano, Silva e Fabbri (2016), Fabbri et al. (2016).

No campo da medicina, os pesquisadores utilizaram a classificação automatizada de documentos, particularmente aplicando algoritmos AM para construir classificadores, esses estudos foram Ramampiaro et al. (2010), Aphinyanaphongs et al. (2005), Cohen et al. (2006), Kouznetsov et al. (2009) e Hassler et al. (2014). No entanto, o desempenho da classificação depende muito da qualidade dos estudos primários. Trabalhos na medicina estão focados na redução da carga de trabalho dos revisores durante a triagem inicial, eliminando o maior número possível de documentos não relevantes, incluindo nada menos que 95% dos documentos relevantes em Wong et al. (2003), Cohen et al. (2006).

2.2 Principais Trabalhos Relacionados

Nesta seção serão apresentados os principais trabalhos relacionados que foram identificados nos estudos apresentados na Seção 2.1. Dois trabalhos focados em automatizar a seleção na área de Engenharia de Software não foram identificados nos trabalhos mencionados por terem sido publicados recentemente e serão discutidos nesta seção. A maioria dos estudos identificados nesses trabalhos estão ligados à mineração de texto e aprendizagem de máquina. No entanto, apesar desses estudos estarem relacionados a este trabalho, apenas alguns possuem uma maior compatibilidade de ideias com esta dissertação.

Foram considerados trabalhos que visam reduzir a quantidade de estudos que devem ser lidos na etapa inicial de seleção. Os trabalhos que serão discutidos utilizam aprendizagem ativa (apesar da abordagem apresentada nesta dissertação não utilizar esse tipo de aprendizado). Existe uma exceção, para os trabalhos de Octaviano et al. (2015),

Fabbri et al. (2016), Octaviano, Silva e Fabbri (2016), pois é aplicável sem a necessidade de uma leitura inicial dos trabalhos e foi considerada uma abordagem interessante a ser discutida.

Em Fabbri et al. (2016) é apresentada a ferramenta que suporta todas as etapas de uma revisão, denominada *StArt*. Nesse estudo foi apresentada um nova estratégia de seleção de artigos denominada *Score Citation Automatic Selection* (SCAS). Essa estratégia foi avaliada em Octaviano et al. (2015), Octaviano, Silva e Fabbri (2016). O SCAS consiste em duas etapas: a primeira realiza a atribuição de pontuação de score e citação, na segunda é utilizada uma técnica de quadrantes para pré-classificar os artigos. O algoritmo J48 foi utilizado para apoiar na definição das regras, detalhes não serão discutidos aqui. Por fim, o estudo realizado aponta uma redução média de esforço de até 58,2% com um erro de 12,98% em Octaviano et al. (2015). Enquanto, em Octaviano, Silva e Fabbri (2016) a redução foi de 22,33% e a taxa de erro 3,95%, além disso o *recall* foi de 90,25% com uma precisão de 65,49%.

Jonnalagadda e Petitti (2013) apresentou um sistema que reduz a carga de trabalho de revisões sistemáticas. A abordagem tem semelhanças com o aprendizado ativo, mas é projetada para se adaptar com uma maior facilidade. Primeiramente, um documento é selecionado aleatoriamente e apresentado ao revisor, depois o artigo deve ser classificado como aceito ou rejeitado. O modelo semântico é utilizado para apresentar o próximo modelo ao revisor, sendo essa escolha feita baseada na similaridade dos documentos. Esse documento é anotado e classificado como relevante ou não. O *feedback* do revisor é utilizado para apresentar o próximo documento. O revisor pode optar por encerrar o processo de classificação, reconhecendo que ao parar, é feita a troca de um *recall* alto, por uma redução no esforço. Foi realizada uma avaliação com 15 revisões sistemáticas de medicamentos publicadas. O número de artigos a serem revisados foi reduzido, com uma variação de redução de 6% a 30% com um *recall* de 95%.

Aprendizagem ativa também foi utilizada no trabalho de Miwa et al. (2014), em revisões sistemáticas de medicina clínica (foram três) e em ciências sociais (especificamente, saúde pública, com um total de quatro estudos). Foram utilizadas SVMs para aprendizado e na modelagem de tópicos o *Latent Dirichlet Allocation* (LDA). Foi constatado que influenciou na melhoria do aprendizado, configurado com 300 tópicos e 30 iterações. Diversos testes foram realizados, como critério de escolha de artigos. Foi aplicado um critério denominado critério de certeza e a ponderação de instâncias positivas foi promissora para superar o problema do desequilíbrio dos dados. A forma que os dados foram apresentados, dificultou uma apresentação da taxa de redução de esforço e o *recall* obtido nos testes realizados.

Em Wallace et al. (2010) foram utilizadas SVMs para a aprendizagem ativa, foi utilizada uma estratégia personalizada para lidar com o desbalanceamento de dados. A

estratégia é denominada *Patient Active Learning* (PAL). Basicamente o PAL funciona da seguinte forma: primeiro explora o espaço de citações, solicitando rótulos para as instâncias aleatoriamente; esse processo encerra quando uma representação razoável da classe minoritária (artigos aceitos) é obtida. Além disso, as estratégias comuns de representação de dados, UMLS e MeSH (detalhes não serão apresentados, para saber mais veja Wallace et al. (2010)), são gerados a partir do título e resumo dos artigos. Os resultados foram positivos nas três revisões utilizadas para avaliar a abordagem: em duas a redução foi de 50% e na outra foi de 40%. Além disso, a redução foi obtida sem perda de artigos relevantes.

Wallace possui vários trabalhos focados em aprendizagem ativa. Em Wallace et al. (2010) foi descoberto que o conceito alvo (critérios de elegibilidade de um artigo) pode sofrer alteração ao longo do tempo, tornando assim necessária a participação do revisor com outras entradas, além da definição do rótulo. É necessário identificar quando, durante o processo de treinamento, os rótulos provavelmente se tornam confiáveis. Foi utilizado um modelo denominado *CoFeature*, que é capaz de identificar alterações de critérios. Dessa forma, quando um novo artigo é rotulado, é verificado se trabalhos já avaliados passaram a possuir relação com outro rótulo (por exemplo, se um artigo aceito passou a ter maior relação com a classe dos rejeitados).

No trabalho realizado por Ma (2007), o aprendizado ativo foi utilizado para selecionar instâncias com maior ganho de informação para serem rotuladas, isso gerou uma redução no conjunto de treinamento. Diversos algoritmos de seleção de características foram utilizados. O melhor desempenho foi apresentado pelo BNS modificado (ver Ma (2007)). A seleção de amostras foi feita utilizando a amostragem por clusterização. Diversos algoritmos de aprendizagem de máquina foram testados: *Naive Bayes* (NB), *Decision Tree* e SVM. O algoritmo que obteve o melhor desempenho foi o NB. A redução no esforço de rotulagem foi de 86%, o *recall* obtido foi de 100%, a precisão de 58,43%, e por fim, a redução de esforço WSS foi de 53,4%.

Em Yu e Menzies (2017), Yu, Kraft e Menzies (2018) foi realizada uma pesquisa que explorou o uso de aprendizagem ativa com 32 variações, destacando-se como o primeiro a realizar esse tipo de estudo no domínio da Engenharia de Software. Seus resultados foram comparados a outros trabalhos existentes na literatura que utilizam aprendizado ativo e que foram identificados nos estudos mencionados anteriormente. Entre suas descobertas se destaca que as soluções obtidas em outros domínios precisam ser adaptadas para se adequarem a ES e não aplicadas diretamente, pois seu desempenho é reduzido. A ferramenta desenvolvida denominada *FASTREAD* foi capaz de reduzir a quantidade de artigos a serem lidos de 20% à 50% com um *recall* de até 95%.

O trabalho anterior foi continuado em Yu e Menzies (2019), algumas melhorias foram feitas em relação a ferramenta anterior e uma nova foi proposta denominada

FAST^2. Destacam-se como melhorias: (1) uma nova técnica para auxiliar na detecção de estudos relevantes a serem lidos, ou seja, conjunto inicial de estudos; (2) um estimador da quantidade de artigos relevantes para auxiliar o pesquisador a decidir quando parar a leitura de artigos; (3) um algoritmo de correção de erro humano, ou seja, detecta quando um artigo é rotulado de maneira incorreta (aceito ou rejeitado). Os resultados obtidos para *recall* de 95% foram e uma redução de 10% à 20% com uma maior robustez.

A Tabela 8 apresenta um resumo dos principais trabalhos relacionados. Apesar deste trabalho não utilizar o aprendizado ativo algumas semelhanças podem ser identificadas com os trabalhos supracitados. Entre as diferenças pode-se destacar que os trabalhos focaram em melhorar o aprendizado ativo e como consequência reduziram o esforço de leitura. O trabalho apresentado aqui focou em aplicar diversas variações no pré-processamento (técnicas de PLN) visando reduzir o esforço.

Tabela 8 – Resumo dos principais trabalhos relacionados.

Estudo	**Algoritmos**	**Técnicas de PLN**	**Recall**	**Redução**	**Dados**
Fabbri et al. (2016)	J48 (C4.5)	Nenhuma	0,902	58%	-
Jonnalagadda e Petitti (2013)	Não se aplica	Hyperspace Analogue to Language (HAL)	0,950	30%	15 revisões
Miwa et al. (2014)	SVMs	BOW com TF-IDF+LDA	-	-	07 revisões
Wallace et al. (2010)	SVMs	BOW+TF-IDF com auxilio de UMLS/MeSH para identificar termos relevantes.	1,000	50%	03 revisões
Wallace et al. (2010)	SVM	BOW Binária	-	-	03 revisões
Ma (2007)	*Naive Bayes*, *Decision Tree* e SVM	BOW	1,000	86%	48000 artigos
Yu, Kraft e Menzies (2018)	SVM	BOW com TF-IDF	0,950	50%	05 revisões
Yu e Menzies (2019)	SVM	BOW com TF-IDF	0,950	70%	05 revisões

Na FAST^2 a seleção de palavras foi realizada utilizando as 4000 palavras com o melhor TF-IDF. Neste trabalho, identificamos que é possível melhorar os resultados

aplicando os cortes de Luhn. Para isso, foram feitos testes todos com os possíveis pontos de corte.

2.3 Considerações Finais

Neste capítulo foram apresentados os principais trabalhos existentes na literatura. Isso foi possível com o uso de dois Mapeamentos Sistemáticos, que apresentam uma visão ampla da área no período de 2004 à 2014. As duas Revisões Sistemáticas identificadas apresentaram uma análise mais rigorosa da área, apresentando as principais lacunas passíveis de investigação e que compreendem o período de 2004 à 2016. Além disso, foram identificados estudos relevantes a serem discutidos que foram publicados recentemente com o mesmo objetivo desta dissertação.

3 Método Proposto

Este trabalho apresenta uma abordagem para reduzir o esforço e tempo na execução de estudos sistemáticos, com foco na fase inicial de seleção dos trabalhos relevantes. A abordagem utiliza as características textuais dos artigos, obtidos a partir do título, resumo e palavras-chave, para aprender sobre a seleção realizada pelos revisores e para reduzir o número de trabalhos que devem ser lidos na fase de seleção. A Figura 13 apresenta a abordagem. Até a escrita desta dissertação o método não foi implementado em uma ferramenta, o próximo passo será disponibiliza-lo na ferramenta *TheEnd*[13].

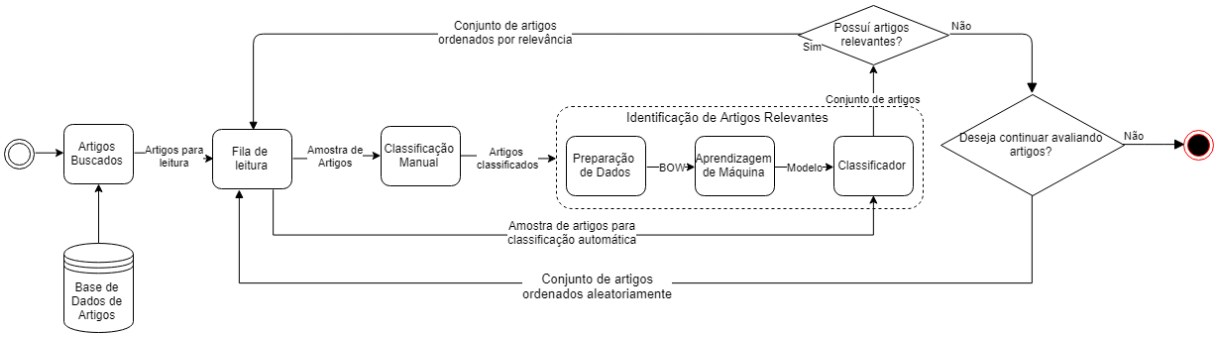

Figura 13 – Etapas da abordagem proposta.

A etapa de Preparação de Dados na Figura 13 foi adicionada como passo da etapa Identificação de Artigos Relevantes para facilitar o entendimento. Na aplicação real esse passo é executado somente uma vez no início do método, pois não é necessária a sua execução diversas vezes.

3.1 Artigos Identificados

No início de uma revisão bibliográfica, utilizando os métodos de MSL ou RSL, é realizada uma busca dos estudos em bases de dados científicas definidas no planejamento do estudo. Os artigos obtidos nesta pesquisa são utilizados em várias outras etapas, a fim de selecionar os relevantes. Após a pesquisa, os documentos duplicados são removidos e, em seguida, o processo de leitura inicial deve ser iniciado.

Para facilitar o entendimento da abordagem, considere um cenário simulado, contendo uma revisão com 100 artigos a serem avaliados.

[13] *TheEnd* – <https://easii.ufpi.br/theend/>

3.2 Fila de Leitura

Inicialmente, os artigos a serem lidos não possuem uma ordenação de acordo com a sua possibilidade de aceitação. Porém, conforme a abordagem é executada, artigos considerados relevantes pelo classificador (3.4.3) passam a ser priorizados nessa fila.

Uma amostra de artigos que estão no início da fila devem ser classificados manualmente (3.3). Os demais estudos da fila são classificados de forma automática pelo classificador na etapa Classificador.

Em nosso cenário simulado, inicialmente, é retirada do início da fila uma amostra de 10% para uma classificação manual e 90% para classificação automática.

3.3 Classificação Manual de Artigos

É necessário que alguns artigos possuam uma classe definida para que seja possível aprender como classificá-los, uma vez que os algoritmos de aprendizagem supervisionada necessitam de um conjunto com classe definida para que sejam treinados. Nesta etapa é feita a leitura dos artigos, considerando somente título, resumo e palavras-chave, depois é feita a avaliação utilizando os critérios de inclusão e exclusão. Ao final do trabalho alguns artigos passam a ter uma classe, que pode ser aceito ou rejeitado.

Aqui será feita uma leitura inicial de 10% dos artigos, vamos supor que dos 10 avaliados, a classificação foi de 5 aceitos e 5 rejeitados. Após a revisão é gerada uma BOW com os 10 artigos e sua classificação, que será utilizada na etapa seguinte. Lembrando que essa BOW foi gerada no início, então aqui é feita somente a adição da classe que ele pertence.

3.4 Identificação de Artigos Relevantes

3.4.1 Preparação de Dados

É necessário preparar os documentos para serem utilizados pelos algoritmos de aprendizagem. A Figura 14 mostra os passos a serem executados na preparação dos dados. Todo esse processo foi discutido em detalhes na Seção 1.3.2, exceto o primeiro. Em *Obtendo dados de interesse*, para cada instância (artigo) obtido via pesquisa nas bases de dados utilizadas, devem ser obtidos dados referentes ao título, resumo e palavras chaves. Esses dados devem ser agrupados. Além dos textos, uma informação adicional é necessária: a avaliação final sobre aceitação do trabalho, que pode ser aceito ou não, caso exista tal informação disponível sobre o artigo.

Mesmo após a aplicação do *stemming*, um grande número de palavras é obtido. A

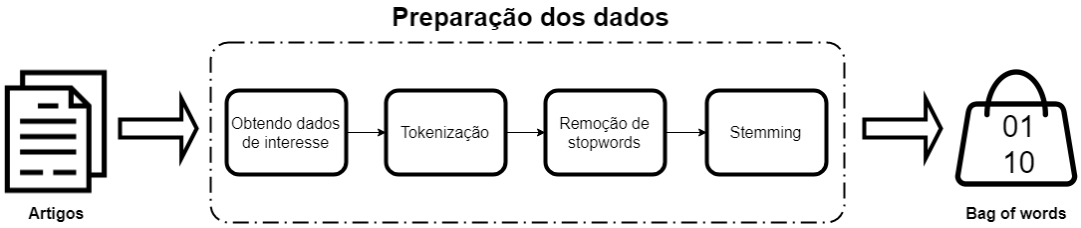

Figura 14 – Processo de preparação dos dados.

maioria delas não é representativa, o que torna necessária a seleção das palavras com uma melhor representatividade. Depois de alguns testes, os melhores resultados foram obtidos utilizando os pontos de corte de Luhn (Seção 1.3.4.1). Após esses tratamentos, os recursos selecionados são convertidos em uma BOW.

Continuando o nosso exemplo, imagine que no início do método essa etapa foi realizada, para os 100 artigos pesquisados, foi identificado um total de 200 palavras relevantes para o tópico da revisão da literatura. Então uma BOW com 100 linhas e 200 colunas deve ser gerada. Nessa BOW, as linhas representam os 100 artigos encontrados no estudo e as colunas as palavras relevantes identificadas. Após a avaliação dos artigos, a BOW gerada no início é consultada e os artigos classificados são adicionados a uma BOW que além das palavras possuem a classificação dos artigos e então é enviada para o treinamento do algoritmo de aprendizagem.

3.4.2 Aprendizagem de Máquina

No aprendizado de máquina, os artigos com status de classificação definido são utilizados em um algoritmo de AM, visando gerar um classificador para o MSL ou RSL em questão. É importante lembrar que para cada MSL ou RSL um novo classificador é treinado e, além disso, aprende-se como a equipe de pesquisadores classificou os trabalhos encontrados. Este método não dispensa a execução de uma etapa de consenso após a seleção.

Continuando nosso exemplo, o algoritmo recebe como entrada a BOW com os artigos que possuem classificação já definida para ser treinado, após sua execução é obtido um classificador de artigos.

3.4.3 Classificador

Após a etapa anterior (Seção 3.4.2) foi obtido um classificador de artigos, que foi treinado baseado nos artigos que já foram avaliados inicialmente. Nessa etapa os trabalhos que ainda não foram lidos serão submetidos ao classificador para serem classificados em aceitos ou rejeitados. A BOW gerada no início da execução do método é consultada e os artigos que não possuem classificação são utilizados nesta etapa.

Artigos classificados como aceitos pelo classificador são chamados de candidatos. Esses estudos precisam ter sua aceitação confirmada por um revisor. Isso é feito priorizando a leitura desses estudos no Passo *Fila de Leitura*.

Quando as sugestões de aceitação não são mais obtidas, essa etapa é encerrada. Se o revisor desejar continuar lendo artigos mesmo sem ter sugestões de artigos relevantes, é possível executar o método inteiro a partir do Passo *Fila de Leitura*, quantas vezes forem consideradas necessárias.

Para continuar a explicação, deve-se continuar com o exemplo iniciado. Agora, deve-se considerar que dos 90 artigos não avaliados, 15 foram classificados como aceitos. Nesse caso o processo continua e agora esses 15 artigos devem ter sua seleção priorizada, na etapa *Classificação Manual de Artigos*. Nessa etapa, após leitura dos 15 trabalhos, 10 foram aceitos e 5 rejeitados. Agora tem-se um conjunto com 25 artigos avaliados, sendo 15 aceitos e 10 rejeitados. Os trabalhos aceitos serão utilizados para um novo aprendizado, para então gerar uma nova lista de possíveis aceitos. Se o sistema não for capaz de identificar novos artigos relevantes, a etapa pode ser dada como encerrada.

Para calcular o desempenho da abordagem, considere que dos 100 artigos existentes, 17 deveriam ser aceitos e o restante rejeitado. Foram aceitos 15 artigos, então o *recall* obtido foi de 88,23% e foram lidos um total de 25 artigos. A redução obtida nesse caso seria de 75%.

Os estudos classificados como rejeitados não serão lidos posteriormente pelos revisores. Portanto, é importante classificar corretamente quando um artigo deve ser rejeitado. No entanto, os trabalhos classificados como aceitos serão lidos pelos revisores para confirmação de sua aceitação. Assim, é importante não rejeitar artigos relevantes para o estudo sistemático. Além disso, os trabalhos rejeitados que não estão relacionados com o propósito do MSL ou RSL terão impacto na redução geral do esforço, pois não precisarão ser lidos.

3.5 Considerações Finais

Neste capítulo foi apresentado o método proposto e suas etapas foram discutidas. A ideia por trás do método é minimizar o trabalho dos revisores, sugerindo artigos para revisão de forma a potencializar o modelo de aprendizado utilizado no contexto em questão. No próximo capítulo serão apresentadas as avaliações realizadas seguindo a ideia chave proposta e que auxiliaram à proposição da abordagem na versão em que ela se encontra neste trabalho.

4 Avaliação

Este capítulo apresenta avaliações que foram realizadas visando identificar a adequabilidade do método proposto para redução de trabalho na seleção de artigos em revisões.

Primeiro foi realizada uma avaliação da aprendizagem. Nessa avaliação foram realizadas diversas variações de técnicas de PLN e algoritmos de AM, visando aumentar o nível de acerto na seleção de artigos em mapeamentos para assim, identificar um conjunto de técnicas e um algoritmo que quando utilizado obtivesse os melhores resultados para aplicação no método aqui proposto. Posteriormente, foi realizada uma nova avaliação que além de acrescentar quatro revisões sistemáticas na base de dados para experimentação, adicionou novos algoritmos e incluiu a aplicação dos cortes de Luhn. Por fim, a última avaliação foi realizada comparando o método proposto com um método existente na literatura e que possui excelentes indicadores de avaliação.

Este capítulo está organizado da seguinte forma: na Seção 4.1 as bases de dados utilizadas nas avaliações são apresentadas, juntamente com as informações extraídas em cada avaliação e o processo de geração dos conjuntos de treinamento. Na Seção 4.2 é discutida a avaliação do aprendizado. Por fim, a avaliação que compara o método proposto neste trabalho com um existente na literatura se encontra na Seção 4.3.

4.1 Informações das avaliações

Antes de discutir sobre as avaliações realizadas, serão descritas as bases de dados utilizadas, que são mapeamentos e revisões que foram disponibilizadas para esse estudo. Os mapeamentos utilizados na avaliação foram selecionados por conveniência, pois seus executores permitiram o seu uso. A Tabela 9 apresenta as principais informações sobre cada um deles. Além dos mapeamentos, em Yu, Kraft e Menzies (2018) foi disponibilizada uma base de dados com quatro revisões sistemáticas na área da Engenharia de Software. As informações básicas sobre cada uma delas se encontra na Tabela 10.

Tabela 9 – Mapeamentos sistemáticos utilizados na avaliação.

ID	Mapeamento	Aceitos	Rejeitados
M1	Silva, Moura e Soares (2017)	218 (07%)	2985 (93%)
M2	Carvalho Thasciano e Castro (2019)	84 (06%)	1284 (94%)
M3	Braga et al. (2015)	63 (11%)	531 (89%)
M4	Braga (2019)	88 (17%)	415 (83%)
M5	Alves (2019)	51 (19%)	223 (81%)

Tabela 10 – Revisões sistemáticas utilizadas na avaliação.

ID	Revisão	Aceitos	Rejeitados
R1	Wahono (2015)	62 (01%)	7002 (99%)
R2	Radjenović et al. (2013)	104 (01%)	8911 (99%)
R3	Hall et al. (2011)	48 (01%)	6000 (99%)
R4	Kitchenham et al. (2010)	45 (03%)	1704 (97%)

Os dados das revisões que foram disponibilizadas por Yu, Kraft e Menzies (2018) não são os originais, os três conjuntos de dados foram criados pela engenharia reversa das RSLs existentes. Por conta disso, a quantidade total de artigos identificados e relevantes se difere dos trabalhos originais com exceção do trabalho de Kitchenham et al. (2010) que disponibilizou os dados. Todos os conjuntos de dados das revisões estão disponíveis *on-line* na *Seacraft*, *Zenodo*[14].

Com o objetivo de simular o mesmo processo de leitura dos artigos realizados por revisores em um mapeamento/revisão, os artigos de cada estudo foram divididos em dois conjuntos, teste e treinamento. Essa divisão foi realizada gerando um conjunto de treinamento com intervalos de 10% do conjunto total, até atingir 90%. A Planilha[15] apresenta a quantidade de artigos aceitos e rejeitados em cada conjunto gerado. Esses conjuntos foram gerados com o auxilio da biblioteca Pandas que é uma biblioteca do *Python* que é utilizada para a analise de dados. O método utilizado foi o *sample*[16] que gera amostras aleatórias.

Para uma melhor análise dos resultados obtidos, algumas informações foram apresentadas sobre cada experimentação realizada, além das métricas discutidas na Seção 4.2.1. A Tabela 11 apresenta as informações extraídas, com a chave utilizada para representá-la e uma explicação do que se trata cada uma delas. Essas chaves podem variar de acordo com a avaliação.

4.2 Avaliação do Aprendizado

Nesta seção é apresentada uma avaliação que visa validar o método proposto utilizando cinco mapeamentos (veja a Tabela 9). Cada mapeamento foi dividido em nove conjuntos de treinamento aleatórios, variando de 10% à 90%, simulando assim a seleção inicial de trabalhos feita por revisores. Além da variação no tamanho do conjunto de treinamento, foram realizadas as seguintes variações:

[14] <https://doi.org/10.5281/zenodo.1162952>

[15] Arquivo com informações de revisões e mapeamentos. Disponível em <https://drive.google.com/drive/folders/1Gkwq89JUX05DXtD1mzk-mA5rnb_OrXlx?usp=sharing>

[16] Documentação Pandas método sample, disponível em: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html>

Tabela 11 – Informações extraídas de cada avaliação.

Chave	**Informações**
Estudo	Identificador dos estudos avaliados de acordo com as Tabelas 9 e 10.
Superior	Ponto de corte superior de Luhn aplicado. Quando seu valor for igual a 1 é equivalente a não aplicação de corte.
Inferior	Ponto de corte inferior de Luhn aplicado. Quando seu valor for igual a 1 é equivalente a não aplicação de corte.
CFS	Define se foi aplicado o filtro CFS da Weka.
Palavras	Quantidade de palavras na BOW.
Treinamento	Quantidade de artigos no conjunto de treinamento.
Teste	Quantidade de artigos no conjunto de teste.
VN	Verdadeiros Negativo, artigos que deveriam ser rejeitados e foram rejeitados pelo algoritmo.
FN	Falsos Negativo, artigos que deveriam ser aceitos e foram rejeitados pelo algoritmo.
VP	Verdadeiros Positivo, artigos que deveriam ser aceitos e foram aceitos pelo algoritmo.
FP	Falsos Positivo, artigos que deveriam ser rejeitados e foram aceitos pelo algoritmo.
Recall	O *recall* obtido para a classe dos artigos aceitos.
WSS (Cohen)	Valor do WSS calculado considerando a variável N como o total de artigos no conjunto de teste.
WSS	Valor do WSS calculado considerando a variável N como o total de artigos em todo o estudo (revisão ou mapeamento).
Perdidos	Artigos perdidos na avaliação automática. Equivalente ao FN.
Não Lidos	Artigos que não seriam lidos com a utilização do algoritmo. Equivalente a soma do VN com FN.

- Foram testados 5 algoritmos (NB, CNB, SMO, J48, RF);
- Foram feitas variações na seleção de palavras: (i) utilizando todas as palavras; (ii) limitando em 2000 com o melhor (TF-IDF); (iii) aplicando o filtro CFS; e (iv) variando os cortes superior e inferior de Luhn, de 1% à 100% em incrementos de 1%. Para os cortes de Luhn, foram testados diversos incrementos. Primeiro os pontos de cortes foram testados em intervalos de 10%, depois esse valor foi reduzido até uma taxa de 1%. Além disso, para ser possível testar o incremento de 1%, apenas os algoritmos NB e CNB foram utilizados.

Essa avaliação pode ser dividida em duas partes. A primeira que utiliza todos os algoritmos e variações, enquanto na segunda os cortes de Luhn são aplicados em um menor

conjunto de variações.

Podemos obter a quantidade de resultados da primeira parte da avaliação pelo o produto das seguintes variáveis: $P_1 = M \times T \times A \times L \times CFS$. Onde $M = 5$ é o total de mapeamentos avaliados, $T = 9$ quantidade de conjuntos de treinamento (10% à 90% do total), $A = 5$ quantidade de algoritmos utilizados, $L = 2$ aplicação da limitação de palavras, e por fim, $CFS = 2$ que é a aplicação ou não do filtro CFS. As variáveis L e CFS possuem valor dois e são consideradas no produto, pois foram aplicadas juntas, ou seja, foram feitas limitações na quantidade de palavras com e sem o uso do filtro CFS e vice-versa. Foram obtidos um total de 900 resultados. Essa parte da avaliação foi realizada conforme mostra o Algoritmo 1. Todas as variáveis testadas estão presentes no algoritmo, mas somente são utilizadas as variações citadas anteriormente, por exemplo, para as variáveis *tipo_bow* e *tipo_selecao* foi utilizado somente o TF-IDF (TF e Binário foram testados mas seu desempenho foi inferior ao TF-IDF). Quando aplicado o limite na quantidade de palavras, o *tipo_selecao* é a métrica utilizada para gerar o *rank* de palavras, por exemplo, se utilizado TF-IDF com limite de 2000, são retornadas as 2000 palavras com maior TF-IDF.

Algoritmo 1: Algoritmo de avaliação sem o corte de Luhn.

```
1  Avaliacao(tipo_selecao, tipo_bow, CFS, L)
2  início
3      para cada M ∈ Mapeamentos faça
4          para cada T ∈ {10%, ..., 90%} faça
5              teste, treinamento ← BUSCAR_DADOS(M, T);
6              vocabulario ←
                SELECIONAR_PALAVRAS(teste, treinamento, tipo_selecao, L);
7              bow_treinamento ← GERAR_BOW(teste, vocabulario, tipo_bow);
8              bow_teste ←
                PRE_PROCESSAMENTO(treinamento, vocabulario, tipo_bow);
9              para cada A ∈ Algoritmos faça
10                 modelo ← TREINAR(A, bow_treinamento, CFS);
11                 matriz_confusao ← TESTAR(modelo, bow_teste);
12                 metricas ← CALCULAR_METRICAS(matriz_confusao);
13                 SALVAR_RESULTADOS(M, T, A, L, CFS, metricas);
14             fim
15         fim
16     fim
17 fim
```

Além disso, a partir dos testes com os cortes de Luhn, foi necessário limitar essas variações. Os cortes foram divididos em duas parte. Na primeira, é possível obter o total de variações pela seguinte formula: $P_{2.1} = M \times T \times A \times CFS \times V_{max}$. Os valores de M, CFS e T permanecem os mesmos; $A = 2$, uma vez que foram utilizados somente os algoritmos *Complement Naive Bayes* e *Naive Bayes*. A variável $V_{max} = 4851$ representa a quantidade

máxima de vocabulários gerados, que será explicado mais a frente. O total de resultados obtidos com os cortes é de aproximadamente 873.180, mesmo considerando que esse valor pode variar para menos, pois nem sempre os pontos de corte geram vocabulário. Essa parte da avaliação foi realizada conforme mostra o Algoritmo 2.

Além do que foi mencionado anteriormente, novos testes foram realizados com os cortes. No formato anterior foram acrescentados os trabalhos da Tabela 10, com incremento de 5%. Não foi possível até a escrita deste trabalho avaliar os pontos de cortes em um menor intervalo. É importante ressaltar que os cortes foram realizados nas palavras identificadas no conjunto de treinamento na primeira avaliação dos cortes.

Na segunda parte dos cortes, foram utilizados todos os artigos dos mapeamentos e revisões (Tabelas 9 e 10). Assim como nas avaliações anteriores, pode-se obter a quantidade de resultados pela seguinte formula: $P_{2.2} = M \times T \times V_{max}$, onde M representa a quantidade de revisões e mapeamentos, os valores de cada variável são: $M = 9$, $T = 9$ e $V_{max} = 4851$. No total foram obtidos 392.931 resultados. O Algoritmo 2 também é válido para esses cortes com algumas ressalvas. Só foi utilizado o algoritmo CNB implementado no *Sci-kit*[17], como consequência dessa alteração, o filtro CFS não foi aplicado.

As variações dos testes geraram como saída muitos dados e para uma melhor organização, tais dados estão disponíveis em duas planilhas[18]. Suas informações estão de acordo com as chaves definidas na Tabela 11. Esses arquivos com resultados estão separados por estudo no formato *Comma Separated Values* (CSV) e unidos em uma única planilha. Os resultados dos cortes são extensos, com tamanho em torno de 66MB. A primeira parte da avaliação se encontra nas planilhas denominadas **resultados com/sem limitação palavras**. A segunda parte, em que foram realizados os cortes, estão divididas em dois arquivos: Cortes Luhn V1 e Cortes Luhn V2.

O restante desta seção está dividida em duas seções, sendo a Seção 4.2.1 responsável por apresentar os principais resultados e discussões sobre a avaliação com foco na aprendizagem e na Seção 4.2.2 serão discutidas as ameaças à validade.

4.2.1 Resultados e Discussão

Antes de iniciar a apresentação dos resultados e discuti-los é necessário dividi-los. Para um melhor entendimento, serão definidas três questões de pesquisa, que serão respondidas a partir dos resultados obtidos. As questões são descritas a seguir:

- **Q1**: Qual processo de seleção de palavras melhorou os resultados da aprendizagem?

[17] Documentação disponível em <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB.html>

[18] Disponível em <https://drive.google.com/drive/folders/1Gkwq89JUX05DXtD1mzk-mA5rnb_OrXlx?usp=sharing>

Algoritmo 2: Algoritmo de avaliação com cortes de Luhn.

```
1  Avaliacao(tipo_bow, CFS)
2  início
3      para cada M ∈ Estudos faça
4          para cada T ∈ 10%, ..., 90% faça
5              teste, treinamento ← BUSCAR_DADOS(M, T);
6              para cada corte_superior ∈ {100%, ..., 1%} faça
7                  para cada corte_inferior ∈ {1%, ..., 100%} faça
8                      se corte_superior > corte_inferior então
9                          para cada A ∈ Algoritmos faça
10                             vocabulario ←
                                CORTES_LUHN(teste, treinamento, corte_superior, corte_inferior);
11                             se vocabulario não vazio então
12                                 bow_treinamento ←
                                   GERAR_BOW(teste, vocabulario, tipo_bow);
13                                 bow_teste ←
                                   PRE_PROCESSAMENTO(treinamento, vocabulario, tipo_bow);
14                                 modelo ← TREINAR(A, bow_treinamento, CFS);
15                                 matriz_confusao ← TESTAR(modelo, bow_teste);
16                                 metricas ←
                                   CALCULAR_METRICAS(matriz_confusao);
17                                 SALVAR_RESULTADOS(M, T, A, L, CFS, corte_superior,
18                                 corte_inferior, metricas);
19                         fim
20                     fim
21                 fim
22             fim
23         fim
24     fim
25 fim
```

Para auxiliar no entendimento, para cada tipo de seleção são necessárias questões específicas, que são:

- **Q1.1**: A limitação das palavras, além de reduzir a dimensionalidade da BOW, auxiliou na melhoria dos resultados? Qual a melhor métrica para limitação?
- **Q1.2**: Os cortes propostos por Luhn, melhoraram os resultados? Qual a melhor forma de aplicar os cortes? É possível definir um ponto de corte superior e inferior que possa ser utilizado em todas as revisões?
- **Q1.3**: O filtro CFS melhora os resultados?

- **Q2**: Quantos artigos são necessários para leitura inicial e assim garantir a eficiência dos algoritmos de aprendizagem? É possível definir uma valor em percentual mínimo?

- **Q3**:Qual algoritmo obteve o melhor desempenho, considerando a redução de esforço alcançada e a perda de artigos relevantes?

Q1: Qual processo de seleção de palavras melhorou os resultados da aprendizagem?

Na Tabela 12, tanto a limitação das palavras em 2.000, quanto o uso por completo das palavras, não gerou um *recall* satisfatório e uma redução de esforço expressiva, tornando necessária a exploração de novos métodos para à seleção de palavras. Os cortes propostos por Luhn se mostraram uma boa alternativa para selecionar quais palavras são mais representativas. A seguir serão discutidas questões especificas para cada método de seleção utilizado.

Tabela 12 – Melhores resultados por algoritmo com e sem limitação de palavras na avaliação de validação do método.

	Algoritmo	**Limite**	**Recall**	**WSS**	**Perdidos**	**Não lidos**	**WSS (Cohen)**
M1	CNB	Sim*	0,895	0,075	02	7,52%	0,751
		Não*	0,842	0,082	03	08,24%	0,823
	J48	Sim	0,263	0,094	14	9,43%	0,943
		Não*	0,316	0,096	13	09,62%	0,962
	NB	Sim*	0,895	0,081	02	8,12%	0,810
		Não*	**0,947**	0,082	01	08,21%	0,819
	RF	Sim*	0,110	0,688	130	68,81%	0,983
		Não*	0,081	0,196	34	19,58%	0,980
	SMO	Sim	0,316	0,096	13	9,55%	0,955
		Não	0,368	0,095	12	09,46%	0,946
M2	CNB	Sim*	0,667	0,076	03	07,59%	0,761
		Não*	0,500	0,169	10	16,88%	0,845
	J48	Sim*	0,343	0,390	23	38,94%	0,975
		Não*	0,340	0,566	31	56,52%	0,943
	NB	Sim	0,778	0,083	02	8,30%	0,831
		Não	**0,889**	0,074	01	07,38%	0,738
	RF	Sim*	0,212	0,294	26	29,36%	0,978
		Não*	0,333	0,096	06	09,65%	0,969
	SMO	Sim*	0,245	0,683	40	68,23%	0,975
		Não	0,350	0,192	13	19,15%	0,960

Quando possuir * na coluna limitação, indica que o filtro CFS foi utilizado.

Continua na próxima página.

Tabela 12 – *Continua na página anterior.*

	Algoritmo	**Limite**	**Recall**	**WSS**	**Perdidos**	**Não lidos**	**WSS (Cohen)**
M3	CNB	Sim*	0,833	0,079	01	07,91%	0,786
		Não*	0,796	0,650	11	64,98%	0,722
	J48	Sim*	0,571	0,180	06	18,01%	0,902
		Não	0,655	0,436	10	43,60%	0,873
	NB	Sim*	**0,833**	0,083	01	08,25%	0,819
		Não*	0,786	0,153	03	15,32%	0,766
	RF	Sim*	0,500	0,094	03	09,43%	0,941
		Não*	0,500	0,096	03	09,60%	0,958
	SMO	Sim	0,667	0,360	09	36,03%	0,900
		Não	0,630	0,366	10	36,53%	0,913
M4	CNB	Sim*	0,294	0,251	12	25,05%	0,838
		Não	0,294	0,255	12	25,45%	0,852
	J48	Sim*	0,529	0,251	08	25,05%	0,837
		Não*	0,588	0,243	07	24,25%	0,810
	NB	Sim*	0,647	0,227	06	22,66%	0,757
		Não*	0,647	0,229	06	22,86%	0,763
	RF	Sim*	0,294	0,275	12	27,44%	0,918
		Não*	0,353	0,277	11	27,63%	0,924
	SMO	Sim	**0,667**	0,082	03	08,15%	0,809
		Não	0,667	0,080	30	07,95%	0,789
M5	CNB	Sim*	0,727	0,122	03	12,14%	0,621
		Não*	0,727	0,122	03	12,14%	0,621
	J48	Sim*	0,705	0,371	06	37,09%	0,763
		Não	0,750	0,075	01	07,45%	0,730
	NB	Sim*	0,909	0,118	01	11,79%	0,601
		Não*	0,909	0,114	01	11,43%	0,583
	RF	Sim*	0,500	0,085	02	08,51%	0,842
		Não*	0,500	0,085	02	08,51%	0,842
	SMO	Sim	**1,000**	0,078	00	07,80%	0,759
		Não	**1,000**	0,078	00	07,80%	0,759

Q1.1: A limitação das palavras, além de reduzir a dimensionalidade da BOW, auxiliou na melhoria dos resultados? Qual a melhor métrica para limitação?

É importante ressaltar que as discussões feitas aqui estão relacionadas à limitação das palavras, em que são calculados os valores TF (Binário e TF-IDF). O processo é realizado com os seguintes passos: (i) são extraídos os *tokens*; (ii) remoção de símbolos e números; (iii) remoção de *stopwords*; (iv) aplicação do *stemmer*; (v) cálculo da métrica (TF, Binário, TF-IDF) e; (vi) extração das "melhores" palavras.

Cada métrica tem um critério para definir as melhores palavras. A binária tem como critério apenas a ordem alfabética das palavras. A TF é utilizada a frequência de documentos, ou seja, palavras que aparecem em mais documentos são as que possuem o melhor *ranking*. O TF-IDF utiliza o seu próprio valor como critério, de forma que as que possuem o maior valor são as escolhidas, e foram utilizadas na avaliação.

Foram realizados testes para verificar a influência da quantidade de palavras utilizadas para se selecionar trabalhos. Nos testes inicialmente realizados, notou-se que o uso de todas as palavras existentes nas informações dos artigos, especialmente nos resumos, era muito numerosa, e a sua utilização por completo prejudicou a classificação, uma vez que os algoritmos tiveram um menor *recall* em relação aos testes com limitação. Outro problema que ocorreu ao se utilizar todas as palavras foi a demora na execução. Sendo assim, foi escolhido empiricamente que se deveria utilizar as 2000 palavras com melhor TF-IDF, pois os resultados preliminares foram bons e houve uma redução no tempo de execução.

Foi utilizado o mapeamento M1 para escolha da quantidade de palavras. Foi identificado que a classe dos artigos aceitos possuía um total de 3139 palavras e rejeitados 20994. Desse total, 2433 eram comuns entre as classes, consequentemente, o vocabulário único da classe dos aceitos possui 706 palavras e dos rejeitados 18561, resultando em um total de 19267 (únicas por classe). A limitação em 2000 foi fundamental para o equilíbrio das palavras por classe. A Tabela 13 apresenta os valores obtidos por classe com variação na quantidade de palavras. Realizando variações em intervalos de 1000 palavras, ao utilizar 3000 palavras identifica-se um equilíbrio. Assim, como o seu uso não melhorou os resultados e aumentou o tempo de execução, optou-se por utilizar 2000 palavras como um padrão nesse quesito. Esse valor impactou na melhoria da classificação e na redução do tempo de execução.

Posteriormente, os bons resultados foram validados com os demais mapeamentos. Quando aplicada a limitação nos demais estudos, os bons resultados com limitação não foram unânimes, em alguns foram superiores em outros não. Os resultados obtidos mostram que é necessário limitar as palavras, mas a utilização do TF-IDF não foi capaz de melhorar os resultados em todos os casos, tornando necessária a utilização de outras técnicas. Além

Tabela 13 – Quantidade de palavras por classe, variando de 1000 a 10000 palavras mais frequentes em M1.

Quantidade	**Aceitos**	**Rejeitados**	**Comuns**
1000	298	298	702
2000	780	780	1220
3000	1360	1360	1640
4000	1293	2154	1846
5000	1173	3034	1966
6000	1086	3947	2053
7000	1017	4878	2122
8000	969	5830	2170
9000	926	6787	2213
10000	885	7746	2254
20994 (Todas)	706	18561	2433

disso, é importante ressaltar que o CFS foi utilizado nessa análise.

Q1.2: Os cortes propostos por Luhn, melhoraram os resultados? Qual a melhor forma de aplicar os cortes? É possível definir um ponto de corte superior e inferior que possa ser utilizado em todas as revisões?

A aplicação dos cortes melhorou bastante os resultados obtidos. Para sua descrição, será considerado que a primeira parte dessa avaliação de Luhn V1 e a segunda de V2. Uma vez que existem muitos resultados nessa avaliação, serão apresentados apenas os que possuem o melhor *recall* e percentual de redução de estudos que não serão lidos. A Tabela 14 apresenta o melhor resultado por estudo para ambos os cortes.

Tabela 14 – Melhores resultados por corte na avaliação de validação do método.

	Corte	**Superior**	**Inferior**	**Recall**	**WSS (Cohen)**	**Não lidos**	**Treinamento**
M1	**V1**	17	03	1,000	0,519	26%	50%
	V2	72	18	0,978	0,600	36%	40%
M2	**V1**	07	02	1,000	0,709	07%	90%
	V2	08	02	1,000	0,738	07%	90%
M3	V1	30	20	1,000	0,721	22%	70%
	V2	37	25	1,000	0,680	34%	50%
M4	**V1**	98	93	1,000	0,245	22%	10%
	V2	01	94	1,000	0,238	21%	10%
M5	**V1**	83	19	1,000	0,606	36%	40%
	V2	02	01	0,923	0,321	19%	40%

Continua na próxima página.

Tabela 14 – *Continua na página anterior.*

	Corte	Superior	Inferior	Recall	WSS (Cohen)	Não lidos	Treinamento
R1	V1	46	11	1,000	0,773	46%	40%
	V2	13	05	1,000	0,707	30%	36%
R2	**V1**	21	01	1,000	0,743	44%	40%
	V2	09	03	1,000	0,822	25%	70%
R3	V1	66	06	1,000	0,868	61%	30%
	V2	52	06	1,000	0,858	69%	20%
R4	V1	21	11	1,000	0,634	25%	60%
	V2	21	09	1,000	0,681	27%	60%

Primeiramente será discutido os resultados comparando os dados com e sem a utilização dos cortes. Analisando os dados apresentados nas Tabelas 12 e 14 é possível visualizar que os resultados obtidos com os cortes foram melhores, obtendo um *recall* de 1,000 em todos os resultados em V1. Além disso, foi possível alcançar melhores reduções no esforço de leitura. Enquanto sem os cortes foi possível reduzir de 7,38% a 8,25%, com os cortes foi possível reduzir de 7,00% a 36,00%, considerando os melhores resultados.

Antes de comparar os resultados obtidos com as duas versões testadas de cortes, é necessário reforçar que foram utilizadas implementações diferentes do CNB, no primeiro a versão disponível na Weka e no segundo do *Sci-kit*. Ficou faltando a realização de testes cruzando as implementações do CNB para descartar ou comprovar a influência da implementação utilizada.

Considerando que as implementações do CNB não influenciaram nos resultados e considerando apenas os obtidos nos dados associados às bases de mapeamentos, a aplicação dos cortes no conjunto de treinamento (V1) foi capaz de identificar as "melhores" palavras. O equilíbrio das classes é um fator que pode contribuir para esses resultados. Equilíbrio, nesse caso é a quantidade de artigos aceitos e rejeitados no conjunto de treinamento. Ambos os cortes foram testados para os mesmos conjuntos de treinamento, ficando a quantidade de artigos rejeitados e aceitos descritos na Tabela 9 e na Tabela 15. Na Tabela 9 são mostradas a quantidade de artigos por classe e na Tabela 15 em cada conjunto de treinamento associado aos melhores resultados.

Como pode ser visto na Tabela 15, os conjuntos de teste gerados na maioria dos casos possuem a mesma proporção de artigos aceitos e rejeitados de todo conjunto. Por exemplo, no M1 07% são aceitos e o restante rejeitados. Para o conjunto de teste apresentado na tabela, a quantidade de aceitos é 06%, bem próximo do percentual de

Tabela 15 – Artigos aceitos e rejeitados por melhores resultados do corte V1 no conjunto de treinamento e teste dos mapeamentos e revisões.

	Treinamento		**Teste**	
ID	**Aceitos**	**Rejeitados**	**Aceitos**	**Rejeitados**
M1	122 (07,62%)	1479 (92,38%)	96 (05,99%)	1506 (94,01%)
M2	75 (05,91%)	1194 (94,09%)	09 (06,38%)	132 (93,62%)
M3	71 (20,17%)	281 (79,83%)	17 (11,26%)	134 (88,74%)
M4	09 (18,00%)	41 (82,00%)	79 (17,44%)	374 (82,56%)
M5	25 (22,32%)	87 (77,68%)	26 (15,76%)	139 (84,24%)
R1	22 (00,79%)	2778 (99,21%)	40 (00,95%)	4160 (99,05%)
R2	27 (01,13%)	2373 (98,88%)	21 (00,59%)	3568 (99,41%)
R3	34 (01,27%)	2639 (98,73%)	70 (01,12%)	6153 (98,88%)
R4	25 (02,45%)	997 (97,55%)	20 (02,94%)	660 (97,06%)

aceitos do conjunto original. Com isso, a possibilidade da quantidade de artigos por classe influenciar no resultado dos experimentos é quase nula, restando apenas o acaso de existir no conjunto de treinamento reduzido os artigos com as melhores palavras para aprendizado, obtidos de forma aleatória.

Os resultados obtidos em ambos os cortes foram semelhantes. Em relação ao *recall* o V1 obteve 1,000 em todos os resultados, enquanto o V2 para M1 e M5 obteve um menor *recall*. Em redução de artigos a serem lidos, para o M2 ambos tiveram o mesmo resultado e em M1/M3 foram melhores em V2, enquanto os dois restantes foram melhores em V1. Por conta do *recall*, o V1 se saiu melhor no geral.

Para as revisões presentes na Tabela 14, no corte V1, só foram testados cortes variando em 5%. Não foi necessário realizar testes com um menor incremento, pois apenas com esse teste o *recall* de 1,000 foi obtido. Como os demais pontos de cortes não foram testados e sua não execução favorece os resultados de V2, optou-se por avaliar as revisões separadamente.

O *recall* obtido em ambos os cortes foi o mesmo, sendo 1,000 para todas as revisões. A principal diferença é na redução de artigos a não serem lidos, para V1 foi de 35% a 61%. Em Luhn V2, foi possível reduzir de 25% à 69%.

Para responder o último questionamento, até a escrita deste trabalho não foi possível identificar os pontos de corte superior e inferior que retorne os melhores resultados para todos os estudos avaliados. Além disso, não foram identificados outros fatores que influenciam no desempenho do ponto de corte. São necessárias mais análises dos dados para identificar que outros fatores influenciadores neste quesito.

Q1.3: O filtro CFS melhora os resultados?

O filtro CFS da Weka se mostrou uma boa alternativa quando não se utiliza os cortes de Luhn. Esse filtro foi aplicado em todos os casos, ou seja, quando foram utilizadas todas as palavras, ao limitá-las em 2000, e por fim, ao aplicar os cortes de Luhn V1. A Figura 15 apresenta quatro gráficos de dispersão que ilustram bem a situação supracitada.

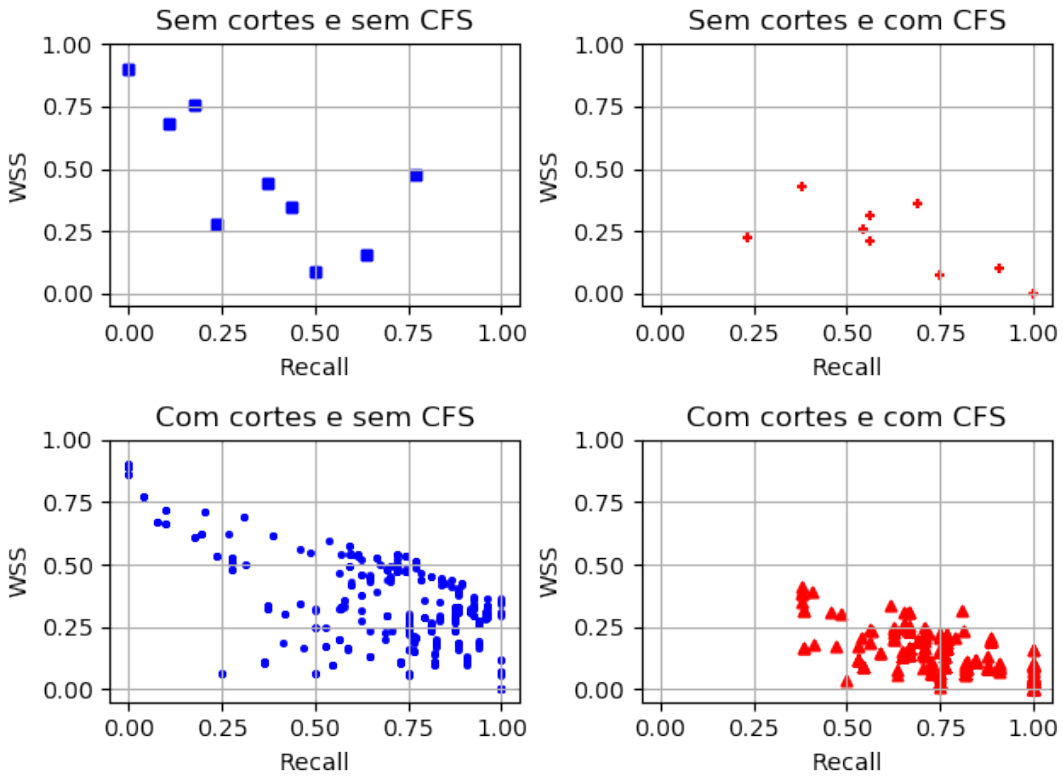

Figura 15 – Gráfico de dispersão do WSS obtido por *recall* nos testes realizados com/sem os cortes de Luhn V1 e com/sem o filtro CFS.

A Figura 15 apresenta os resultados para o mapeamento M5. No primeiro gráfico de dispersão que possui os quadrados, não foi aplicado o corte e o filtro CFS. Nesse caso, o melhor resultado é um *recall* de 0,760, com um WSS de 0,500. No segundo, que possui o símbolo de +, o filtro CFS foi aplicado e o melhor resultado foi um *recall* de 0,950 com um WSS de 0,100.

Nos próximos dois gráficos de dispersão, os cortes foram aplicados. No primeiro, com círculos, sem o uso do filtro CFS, os resultados obtidos foram bons, além de um *recall* alto (1,000), foi possível obter um WSS de aproximadamente 0,370. Após aplicação do filtro CFS, exibido no gráfico com triângulos, o *recall* foi mantido, mas o WSS foi reduzido para aproximadamente 0,200.

Com os cortes de Luhn, os resultados foram melhores, e a aplicação do filtro CFS com os cortes reduziu o valor do WSS. Nesse caso, foi considerado que o CFS não melhora

os resultados. Sem uma definição concreta dos pontos de corte, o CFS é importante para auxiliar na identificação de palavras relevantes.

Q2: Quantos artigos são necessários para leitura inicial e assim garantir a eficiência dos algoritmos de aprendizagem? É possível definir uma valor em percentual mínimo?

Considerando os resultados da Tabela 14, a variação do tamanho do conjunto de treinamento foi grande. Em alguns trabalhos, foi necessária a leitura de 90% dos artigos, enquanto em outros a leitura de 10% foi o suficiente. Calculando a mediana para os percentuais de treinamento, em V1 foi necessária a leitura de 40% com IQR 30%, enquanto em V2 a mediana foi a mesma com o IQR 40%. Considerando esses percentuais, pode-se considerar que 40% é um percentual mínimo para garantir bons resultados.

Q3: Qual algoritmo obteve o melhor desempenho, considerando a redução de esforço alcançada e a perda de artigos relevantes?

Para responder essa pergunta é necessário analisar os melhores resultados de cada algoritmo para todos os mapeamentos utilizados nesta avaliação. A Tabela 12 mostra o melhor resultado para cada algoritmo com e sem a limitação na quantidade de palavras. Na Tabela 12 os melhores desempenhos foram definidos baseados no *recall* para a classe dos artigos aceitos e depois a melhor redução em percentual. Apesar da baixa taxa de acerto obtida nesta avaliação, foi possível observar que algoritmos se adequam melhor ao problema de seleção de estudos. Analisando os resultados neste ponto de vista, os algoritmos que se destacam são o *Naive Bayes* e *Complement Naive Bayes*. Para o M5, SMO obteve o melhor resultado, chegando a um *recall* de 100%, no entanto esse foi um caso isolado e seu desempenho foi inferior nos demais testes realizados.

Ao realizar a segunda parte desta avaliação, que consiste em aplicar os cortes de Luhn apenas os algoritmos NB/CNB foram utilizados. Nos testes realizados o CNB obteve os melhores resultados. Essa parte da avaliação será discutida em maiores detalhes na **Q2.2** (Seção 4.2.1). Os melhores resultados apresentados até o momento não tiveram perda de artigos relevantes. Para analisar a questão de redução de esforço, será considerado que um bom resultado é aquele capaz de reduzir em pelo menos 30% os artigos a serem lidos. A Tabela 16 apresenta os melhores resultados para um redução de pelo menos 30%, nos testes com corte de Luhn.

Como exibido na Tabela 16, ambos os cortes tiveram bons resultados de redução, vale ressaltar que foram listados somente os resultados com pelo menos 30% de redução. O Corte V1 obteve reduções de 31% à 61%, já os resultados do V2, variaram de 34% à 69%. Em contrapartida a esses percentuais de redução, o *recall* em V1 variou de 0,812 à

Tabela 16 – Melhores resultados por redução de pelo menos 30% nos cortes de Luhn dos mapeamentos e revisões.

ID	Corte	Redução	Recall	WSS (Cohen)
M1	V1	32%	0,979	0,647
M2	V1	46%	0,936	0,768
M3	V2	34%	1,000	0,680
M4	V1	31%	0,812	0,448
M5	V1	36%	1,000	0,606
R1	V2	50%	1,000	0,000
R2	V1	44%	1,000	0,773
R3	V2	69%	1,000	0,858
R4	V2	46%	0,889	0,572

1,000 e em V2 de 0,797 à 1,000. Os melhores resultados obtidos nos cortes sem limitar aos que atingiram 30% de redução, o *recall* obtido em V1 foi de 1,000 e para V2 de 0,946 à 1,000. Por fim, a redução obtida para V1 foi de 07% à 61% e em V2, foi possível reduzir de 25% à 69%.

4.2.2 Ameaças a Validade

A realização de uma avaliação está sujeita a várias ameaças a sua validade (FELDT; MAGAZINIUS, 2010). Nas conclusões feitas a partir desta avaliação devem ser considerados os seguintes problemas:

Todas as conclusões deste estudo foram tiradas dos testes realizados em cinco conjuntos de dados de mapeamentos sistemáticos (Tabela 9). Como consequência, as conclusões podem não ser aplicáveis em outros conjuntos de dados. Visando reduzir essa ameaça, após proposição do método, foram utilizados quatro novos conjuntos de dados de revisões sistemáticas para avaliar o método (Tabela 10).

Os trabalhos foram divididos em 9 conjuntos de treinamento com percentuais variando em 10% do conjunto original, a quantidade de artigos aceitos e rejeitados no conjunto de treinamento pode ter influenciado nos resultados obtidos. Para evitar isso, a cada conjunto gerado, os artigos utilizados foram obtidos de forma aleatória do conjunto original.

Outro ponto que deve ser levado em consideração, são as implementações dos algoritmos utilizados. Em todos os casos não foram testadas variações nos parâmetros dos algoritmos. Assim, podem existir parâmetros que se alterados seriam capazes de melhorar o desempenho do algoritmos testado, as implementações também devem ser consideradas, pois podem influenciar nos resultados.

4.3 Avaliação comparativa com a FAST^2

Nesta seção será apresentada uma avaliação que foi realizada com o objetivo de comparar a redução de esforço na seleção de artigos obtida na abordagem proposta neste estudo, com o trabalho de Yu e Menzies (2019), em sua última versão disponível[19].

Reduzir o esforço na seleção inicial de artigos em revisões é um problema multiobjetivo. Então, para uma melhor análise de seu desempenho é necessário avaliar quantos artigos relevantes foram identificados e o quanto de esforço de leitura foi reduzido.

Assim como no trabalho de Yu e Menzies (2019), foi fixado o *recall* dos aceitos em 0,950. No entanto, podem existir situações em que esse valor será relaxado. Esse valor é utilizado em outros estudos presentes na literatura, como o que foi discutido em Cohen (2011), Cohen et al. (2006), O'Mara-Eves et al. (2015). Segundo Shemilt et al. (2014), Yu, Kraft e Menzies (2018), o *recall* de 1,000 não pode ser garantido com a aplicação de mineração de texto, sem que seja necessário realizar a leitura de todos os artigos, pois não se pode garantir que todos os estudos considerados relevantes foram identificados.

Para a ferramenta FAST^2, proposta em Yu e Menzies (2019), é feita a avaliação da mesma forma que foi descrito em Yu e Menzies (2019). O Algoritmo 3 apresenta os passos do teste realizado. Para a entrada *Estudos* foram utilizados os mapeamentos sistemáticos da Tabela 9 e em *Repetições* foram realizadas 30, e são avaliados 10 artigos a cada *loop*, assim como em seu trabalho. Não foi possível utilizar as revisões da Tabela 10. Por conta disso, serão utilizados os resultados obtidos em Yu e Menzies (2019). O Algoritmo 3 representa somente o processo de execução, para o algoritmo da FAST^2 é utilizada a sua versão atual no GitHub.

Para o método proposto nesta dissertação, serão utilizados os 30 melhores resultados obtidos na avaliação anterior, ou seja, os que foram discutidos na Seção 4.2.1 para o corte de Luhn V2. Entre os testes realizados, o corte V2 é o que possui menos variações, ou seja, a utilização dos demais resultados iria gerar contradições ao que foi considerado até o momento como melhor resultado da abordagem. A Tabela 17 apresenta a relação entre os estudos e as abordagens. Onde refinamento, significa que o estudo foi utilizado para refinar a abordagem, enquanto o teste, indica que o estudo foi utilizado para testar a abordagem.

Tabela 17 – Utilização dos estudos por abordagem.

	Abordagem	**FAST^2**
Refinamento	Mapeamentos	Revisões
Teste	Revisões	Mapeamentos

[19] *Último commit em 16 de Março de 2019* – <https://github.com/fastread/src>

Algoritmo 3: Algoritmo utilizado para avaliar a FAST^2, conforme o trabalho de Yu e Menzies (2019).

```
1  avaliacao(Estudos, Repetições)
2  início
3      para cada E ∈ Estudos faça
4          para cada R ∈ {1, ..., Repetições} faça
5              artigos ← CARREGA_ARTIGOS(10);
6              parar_avaliação ← Não;
7              repita
8                  para cada artigo ∈ artigos faça
9                      artigo_classe ← CARREGA_CLASSE(artigo);
10                     quantidade_aceitos, quantidade_lidos, total_artigos ←
                        AVALIAR_ARTIGO(artigo, artigo_classe);
11                     se quantidade_aceitos ≥ (total_aceitos × recall) então
12                         parar_avaliação ← Sim;
13                 fim
14                 artigos ← CARREGA_ARTIGOS(10);
15             até parar_avaliação = Sim;
16             GERAR_ARQUIVO_RESULTADOS();
17             REINICIAR_AVALIAÇÃO();
18         fim
19     fim
20 fim
```

4.3.1 Resultados e Discussão

Para a FAST^2 foram salvos os dados de total de artigos, quantidade de aceitos estimados e total de artigos lidos (artigos que foram buscados e lidos em intervalos de 10) para todas as 30 repetições realizadas, em alguns momentos o algoritmo não foi capaz de continuar com a avaliação e foi necessário executar mais de 30 repetições. Baseado nesses dados foram calculados o *recall* dos artigos aceitos e redução de esforço em percentual (quantidade de artigos não lidos). O WSS (Seção 1.3.6.3) não foi utilizado na comparação, pois o valor de N no cálculo em alguns trabalhos não fica claro se é utilizado o total de artigos ou a quantidade de artigos do conjunto de teste. Com a redução em percentual é possível analisar a capacidade das ferramenta em reduzir o esforço. Para a abordagem proposta, os dados obtidos são os já mencionados na Tabela 11, mas apenas os em comum serão utilizados para comparação. Todos os resultados serão reportados em termos de mediada (percentil 50) e IQRs (percentil (75-25)).

Para uma melhor avaliação da redução de esforço obtida, a Tabela 18 mostra a mediana e o IRQ obtido nos resultados em ambas as abordagens obtidos nos mapeamentos. Para o FAST^2, foi possível obter em seu melhor resultado uma redução de 65,03% dos artigos a serem lidos. A abordagem proposta neste trabalho obteve resultados inferiores de redução, obtendo 36,03% em seu melhor resultado. A FAST^2 obteve um bom *recall*, que

foram maiores ou igual a 0,950 em todos mapeamentos, enquanto a abordagem proposta obteve valores inferiores em três estudos e nos demais obteve resultados superiores.

Tabela 18 – Redução de esforço e *recall* obtidos pelo método proposto e a FAST^2 nos mapeamentos sistemáticos.

	Abordagem		**FAST^2**	
Estudo	**Recall**	**Redução**	**Recall**	**Redução**
M1	0,978 (0,005)	36,03% (22,82%)	0,954 (0,001)	65,03% (01,87%)
M2	0,900 (0,036)	13,97% (22,59%)	0,952 (0,000)	60,28% (07,10%)
M3	1,000 (0,000)	32,15% (10,44%)	0,952 (0,000)	52,86% (05,05%)
M4	0,944 (0,059)	11,53% (02,98%)	0,955 (0,011)	36,38% (01,99%)
M5	0,923 (0,096)	26,71% (03,43%)	0,961 (0,020)	46,81% (11,52%)

Os valores estão em termos de mediada (percentil 50) e os entre parênteses são os IRQs (percentil (75-25)).

Os resultados referentes as revisões estão presentes na Tabela 19. A abordagem proposta obteve um bom *recall* em todos os seus resultados, que foram de 1,000 para todos, na FAST^2 o *recall* obtido variou de 0,910 à 0,980. Quanto a redução de esforço a abordagem proposta foi capaz de reduzir em até 68,23%, enquanto a FAST^2 na revisão R3 chegou a reduzir o esforço em 94,39%.

Tabela 19 – Redução de esforço e *recall* obtidos pelo método proposto e a FAST^2 nas revisões sistemáticas.

	Abordagem		**FAST^2**	
Estudo	**Recall**	**Redução**	**Recall**	**Redução**
R1	1,000 (0,000)	47,09% (00,24%)	0,950 (0,000)	83,15% (01,00%)
R2	1,000 (0,000)	32,88% (30,76%)	0,940 (0,000)	87,33% (00,60%)
R3	1,000 (0,000)	68,23% (00,87%)	0,980 (0,000)	94,39% (00,30%)
R4	1,000 (0,000)	25,73% (08,53%)	0,910 (0,020)	70,07% (01,20%)

Os valores estão em termos de mediada (percentil 50) e os entre parênteses são os IRQs (percentil (75-25)).

Para o desenvolvimento da abordagem proposta, foram feitos diversos testes utilizando técnicas de PLN e alguns algoritmos de AM. Foi observado que a melhora de desempenho sempre foi maior quando foram aplicadas técnicas para melhorar a escolha das palavras. Entre os testes realizados, foram feitas variações na quantidade de palavras utilizadas baseado no maior valor de TF-IDF obtido, assemelhando-se a FAST^2, que utilizava as 4000 palavras com maior TF-IDF. Quando utilizado os cortes de Luhn (Seção 1.3.4.1), os resultados obtidos foram superiores a todos os outros. Baseando-se nesses testes, acredita-se que essa seleção de palavras tem um grande impacto na redução de esforço, tornando-se o principal ponto de distinção desta abordagem com a ferramenta FAST^2.

A forma como a FAST^2 foi avaliada favoreceu os seus resultados. Com a utilização da aprendizagem na priorização de leitura, foi possível encerrar o processo mais cedo, assim contribuindo para a redução de esforço. Uma avaliação utilizando a abordagem proposta na priorização de leitura é necessária para que seja possível realizar uma avaliação mais justa.

4.3.2 Ameaças a Validade

Uma das questões fundamentais a respeito de uma avaliação refere-se ao grau de validade dos resultados obtidos. Assim, é importante analisar os riscos relacionados à avaliação e propor alternativas para tentar contornar os problemas que possam ameaçar à validade dos resultados (WOHLIN et al., 2012). Quaisquer conclusões feitas a partir desta avaliação devem ser consideradas com os seguintes problemas em mente:

É importante enfatizar que nos experimentos realizados, assumiu-se que os revisores dos estudos (revisões e mapeamentos) estavam sempre certos de suas avaliações. Em contextos reais, podem ocorrer problemas de discordância entre revisores ou mesmo erros de contexto.

Os mapeamentos utilizados para comparar a FAST^2 com a abordagem proposta podem ter favorecido a abordagem, pois eles foram utilizados para refina-la. Para validar a eficiência da abordagem e descartar essa possibilidade, foi realizada uma avaliação do método proposto com as quatro revisões disponibilizadas em Yu, Kraft e Menzies (2018). Para evitar favorecimento da abordagem proposta neste estudo ou da FAST^2, a abordagem proposta foi refinada com o uso dos mapeamentos e validada com as revisões, para FAST^2 foi realizado o inverso, foram utilizados os dados das revisões obtidos em Yu e Menzies (2019) (Tabela 19) e foi validada com a execução da ferramentas nos mapeamentos (Tabela 18).

Para avaliar a FAST^2, foi utilizado o código fonte disponível no repositório GitHub. A execução foi feita de forma simples, entretanto, talvez existam algumas configurações que se utilizadas melhoram os resultados obtidos pela ferramenta. A falta de um maior conhecimento sobre a sua utilização pode ter prejudicado os testes realizados e influenciado nos resultados.

4.4 Considerações Finais

Este capítulo apresentou uma avaliação da abordagem proposta neste trabalho. Essa avaliação foi realizada em duas partes, a primeira foram realizadas diversas variações de técnicas de PLN e algoritmos de aprendizagem visando identificar o que melhor se adequam ao problema. Na segunda parte, o método proposto foi comparado a uma ferramenta existente na literatura com o mesmo objetivo. Por meio dos resultados obtidos pôde-se

perceber que a abordagem proposta obteve um bom desempenho na identificação de artigos relevantes e redução de esforço. Quando comparada a ferramenta FAST^2, sua capacidade de reduzir esforço foi menor, apesar de que em alguns casos foi capaz de identificar uma maior quantidade de artigos relevantes.

5 Conclusões e Trabalhos Futuros

Quando se deseja iniciar uma nova pesquisa é essencial que se faça uma análise do atual estado da arte do tema de interesse. Esta é a principal justificativa para a realização de Revisões e Mapeamentos Sistemáticos da Literatura (RSL e MSL). Uma RSL é um método científico capaz identificar, interpretar e sumarizar os trabalhos relevantes para determinada linha de pesquisa, área ou fenômeno de interesse, de forma não tendenciosa e replicável. O grande e crescente número de trabalhos publicados torna a identificação de estudos relevantes de uma área uma tarefa complexa e demorada.

Esta Dissertação apresentou um método que visa reduzir o esforço necessário na seleção inicial de estudos em Revisões e Mapeamentos Sistemáticos, pois é uma das etapas que exigem mais esforço e, portanto, requer mais tempo para ser concluído. Primeiro foi realizado um estudo das principais técnicas utilizadas para automatizar a seleção de estudos, as informações obtidas foram utilizadas para definir o método proposto.

Foi realizada uma avaliação visando identificar que configurações foram capazes de reduzir a quantidade de artigos a serem lidos, mas considerando também eventuais perdas de trabalhos importantes. Desta avaliação destacam-se as seguintes contribuições:

- Foram apresentadas algumas técnicas de PLN e a influência de cada uma na resolução do problema de seleção de estudos em revisões sistemáticas;
- Alguns algoritmos de aprendizagem de máquina foram utilizados e seu desempenho comparado em relação a capacidade de resolver o problema que é foco deste estudo;
- As investigações dos atributos (palavras), principalmente com a utilização dos cortes de Luhn;
- Com a conclusão dos itens anteriores, foi possível propor um método que reduzir o esforço na seleção inicial de artigos em estudos sistemáticos da literatura;
- Um conjunto de dados formado por mapeamentos sistemáticos que podem ser utilizados por outras pesquisas. Disponível em <https://doi.org/10.5281/zenodo.3229373>.

Além disso, o método proposto foi comparado a uma ferramenta existente na literatura. Os resultados obtidos apontam que com uma pequena perda de estudos relevantes é possível se reduzir em até 65% a quantidade de artigos a serem lidos. Apesar desse valor de redução, a FAST^2 obteve um melhor desempenho que o método proposto. A FAST^2

tem como principal desvantagem não estar disponível para comunidade, sua utilização depende de conhecimentos de computação para se utilizar a ferramenta localmente.

5.1 Desafios e Limitações

Apesar dos bons resultados obtidos até o momento, este trabalho ainda apresenta alguns desafios e limitações. O maior deles refere-se a avaliação do método proposto. Conforme discutido no Capítulo 4, as avaliações foram realizadas apenas com cinco Mapeamentos Sistemáticos e quatro Revisões Sistemáticas da Literatura. Assim, ao utilizar outros estudos, os resultados obtidos podem se diferenciar, ocasionando uma menor taxa de redução de esforço ou uma maior perda de artigos relevantes. Outra limitação relacionada a avaliação, foi que o método proposto, diferente da FAST^2 não foi avaliado visando a priorização de leitura, isso ocasionou um desempenho de redução de esforço inferior ao obtido com o uso da FAST^2.

Outra limitação está relacionada à escolha do algoritmo de aprendizagem de máquina. Existe uma grande quantidade de algoritmos que podem ser utilizados e infinitas configurações de hiper-parâmetros. Tentando mitigar essa ameaça, foi utilizado um *plugin* da ferramenta WEKA, denominado AUTO-WEKA, que realiza testes com diferentes algoritmos e diversas configurações em um intervalo de tempo determinado pelo usuário.

Por fim, pode-se notar que mesmo com todo trabalho realizado até o momento, ainda há espaço para melhorias e trabalhos futuros. Dessa forma, tais limitações representam novas oportunidades de continuidade da pesquisa.

5.2 Trabalhos Futuros

Com base nos desafios e limitações apresentadas na Seção 5.1 e outros aspectos identificados, foram definidas algumas direções para trabalhos futuros:

- Realização de uma nova avaliação do método proposto, utilizando-o na priorização de leitura de artigos considerados relevantes e assim comprar os seus resultados a FAST^2;
- Realizar novas avaliações com outros métodos que tem como objetivo a redução de esforço em estudos sistemáticos e com isso realizar uma análise comparativa mais robusta com os resultados obtidos com cada um desses métodos;
- Avaliar o uso do método como um outro pesquisador, assim auxiliando na validação das seleções. Para isso, o método deve aprender baseando-se na avaliação dos demais participantes do estudo;

- Implementar o método proposto na ferramenta *TheEnd*;
- Realizar novas avaliações utilizando novos estudos e disponibilizando esses dados para a comunidade;
- Realizar novos experimentos com o uso de novos algoritmos ou explorando mais a fundo os parâmetros que cada um possui visando melhorar o seu desempenho;
- Utilizar outras técnicas de PLN mais robustas, como por exemplo, Word Embeddings (WE).

Referências

ADEVA, J. G. et al. Automatic text classification to support systematic reviews in medicine. *Expert Systems with Applications*, Elsevier, v. 41, n. 4, p. 1498–1508, 2014. Citado na página 33.

AGGARWAL, C. C.; ZHAI, C. A survey of text classification algorithms. In: *Mining text data*. [S.l.]: Springer, 2012. p. 163–222. Citado na página 15.

ALVES, F. V. d. M. *Uma Análise da Familiaridade de Código entre as Perspectivas de Módulo e Funcionalidade*. 2019. Dissertação (Mestrado em Ciência da Computação), UFPI (Universidade Federal do Piauí), Teresina, Brasil. Disponível em: <http://hdl.handle.net/123456789/1775>. Citado na página 43.

ALY, M. Survey on multiclass classification methods. *Neural networks*, Citeseer, v. 19, p. 1–9, 2005. Citado na página 15.

ANANIADOU, S. et al. Supporting systematic reviews using text mining. *Social Science Computer Review*, Sage Publications Sage CA: Los Angeles, CA, v. 27, n. 4, p. 509–523, 2009. Citado na página 15.

APHINYANAPHONGS, Y. et al. Text categorization models for high-quality article retrieval in internal medicine. *Journal of the American Medical Informatics Association*, BMJ Group BMA House, Tavistock Square, London, WC1H 9JR, v. 12, n. 2, p. 207–216, 2005. Citado na página 34.

BABAR, M. A.; ZHANG, H. Systematic literature reviews in software engineering: Preliminary results from interviews with researchers. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. [S.l.], 2009. p. 346–355. Citado na página 2.

BARBARA, K.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. *Keele University, UK*, v. 9, 2007. Citado 8 vezes nas páginas 1, 2, 3, 6, 9, 10, 13 e 14.

BEKHUIS, T.; DEMNER-FUSHMAN, D. Towards automating the initial screening phase of a systematic review. In: *MedInfo*. [S.l.: s.n.], 2010. p. 146–150. Citado na página 33.

BEKHUIS, T. et al. Feature engineering and a proposed decision-support system for systematic reviewers of medical evidence. *PloS one*, Public Library of Science, v. 9, n. 1, p. e86277, 2014. Citado na página 33.

BERGER, A. et al. Bridging the lexical chasm: statistical approaches to answer-finding. In: ACM. *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.], 2000. p. 192–199. Citado na página 20.

BIOLCHINI, J. C. de A. et al. Scientific research ontology to support systematic review in software engineering. *Advanced Engineering Informatics*, Elsevier, v. 21, n. 2, p. 133–151, 2007. Citado 2 vezes nas páginas 13 e 9.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. [S.l.]: "O'Reilly Media, Inc.", 2009. Citado na página 21.

BRAGA, R. et al. Ferramentas para desenvolvimento de sistemas baseados em inteligência computacional: Um mapeamento sistemático. *Anais do Simpósio Brasileiro de Automaçao Inteligente*, v. 12, p. 6, 2015. Citado 2 vezes nas páginas 10 e 43.

BRAGA, R. D. *Um Estudo Prático sobre a Automação em Oráculos de Testes*. 2019. Dissertação (Mestrado em Ciência da Computação), UFPI (Universidade Federal do Piauí), Teresina, Brasil. Disponível em: <http://hdl.handle.net/123456789/1768>. Citado na página 43.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 29.

BRERETON, P. et al. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software*, Elsevier, v. 80, n. 4, p. 571–583, 2007. Citado na página 2.

CARVALHO THASCIANO E CASTRO, O. Técnicas de inteligência artificial aplicadas ao melhoramento genético na pecuária. UFPI. 2019. Citado na página 43.

CERRI, R.; CARVALHO, A. C. P. de; FREITAS, A. A. Adapting non-hierarchical multilabel classification methods for hierarchical multilabel classification. *Intelligent Data Analysis*, IOS Press, v. 15, n. 6, p. 861–887, 2011. Citado na página 15.

CHAVES, M. S. Um estudo e apreciação sobre algoritmos de stemming para a língua portuguesa. *IX Jornadas Iberoamericanas de Informática*, p. 92, 2003. Citado na página 17.

CLARE, A.; KING, R. D. Knowledge discovery in multi-label phenotype data. In: SPRINGER. *European Conference on Principles of Data Mining and Knowledge Discovery*. [S.l.], 2001. p. 42–53. Citado na página 29.

COHEN, A. M. Performance of support-vector-machine-based classification on 15 systematic review topics evaluated with the wss@ 95 measure. *Journal of the American Medical Informatics Association: JAMIA*, American Medical Informatics Association, v. 18, n. 1, p. 104, 2011. Citado na página 58.

COHEN, A. M. et al. Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, BMJ Group BMA House, Tavistock Square, London, WC1H 9JR, v. 13, n. 2, p. 206–219, 2006. Citado 3 vezes nas páginas 24, 34 e 58.

CORDEIRO, A. M. et al. Revisão sistemática: uma revisão narrativa. SciELO Brasil, 2007. Citado na página 2.

CRUZES, D. S.; DYBÅ, T. Research synthesis in software engineering: A tertiary study. *Information and Software Technology*, Elsevier, v. 53, n. 5, p. 440–455, 2011. Citado na página 14.

DASGUPTA, A. et al. Feature selection methods for text classification. In: ACM. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2007. p. 230–239. Citado na página 20.

DIXON, W. J.; JR, F. J. M. Introduction to statistical analysis . McGraw-Hill, 1957. Citado na página 12.

FABBRI, S. et al. Improvements in the start tool to better support the systematic review process. In: ACM. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2016. p. 21. Citado 3 vezes nas páginas 34, 35 e 37.

FAYYAD, U. M. et al. Advances in knowledge discovery and data mining. the MIT Press, 1996. Citado na página 15.

FELDMAN, R.; DAGAN, I. Knowledge discovery in textual databases (kdt). In: *KDD*. [S.l.: s.n.], 1995. v. 95, p. 112–117. Citado na página 14.

FELDT, R.; MAGAZINIUS, A. Validity threats in empirical software engineering research-an initial survey. In: *Seke*. [S.l.: s.n.], 2010. p. 374–379. Citado na página 57.

FELIZARDO, K. R. et al. An approach based on visual text mining to support categorization and classification in the systematic mapping. In: *EASE*. [S.l.: s.n.], 2010. Citado na página 34.

FELIZARDO, K. R. et al. A visual analysis approach to update systematic reviews. In: ACM. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2014. p. 4. Citado na página 34.

FELIZARDO, K. R. et al. Using visual text mining to support the study selection activity in systematic literature reviews. In: IEEE. *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*. [S.l.], 2011. p. 77–86. Citado na página 33.

FENG, L.; CHIAM, Y. K.; LO, S. K. Text-mining techniques and tools for systematic literature reviews: A systematic literature review. In: IEEE. *Asia-Pacific Software Engineering Conference (APSEC), 2017 24th*. [S.l.], 2017. p. 41–50. Citado 4 vezes nas páginas 4, 6, 15 e 34.

GOMES, H. J. C. *Text Mining: análise de sentimentos na classificação de notícias*. Tese (Doutorado), 2013. Citado na página 15.

HALL, M. et al. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, ACM, v. 11, n. 1, p. 10–18, 2009. Citado na página 21.

HALL, M. A. Correlation-based feature subset selection for machine learning. *Thesis submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy at the University of Waikato*, 1998. Citado 2 vezes nas páginas 22 e 23.

HALL, T. et al. A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, IEEE, v. 38, n. 6, p. 1276–1304, 2011. Citado na página 44.

HAN, J. et al. Probabilistic models for classification. In: *Data Classification*. [S.l.]: Chapman and Hall/CRC, 2014. p. 91–112. Citado na página 28.

HASSLER, E. et al. Outcomes of a community workshop to identify and rank barriers to the systematic literature review process. In: ACM. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2014. p. 31. Citado na página 34.

HEARST, M. A. Text data mining: Issues, techniques, and the relationship to information access. In: *Presentation notes for UW/MS workshop on data mining*. [S.l.: s.n.], 1997. p. 112–117. Citado na página 14.

HIGGINS, J. P.; GREEN, S. et al. *Cochrane handbook for systematic reviews of interventions*. [S.l.]: Wiley Online Library, 2008. v. 5. Citado na página 1.

HOTHO, A.; NÜRNBERGER, A.; PAASS, G. A brief survey of text mining. In: CITESEER. *Ldv Forum*. 2005. v. 20, n. 1, p. 19–62. Disponível em: <http://hdl.handle.net/10362/9182>. Citado na página 15.

JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. In: SPRINGER. *European conference on machine learning*. [S.l.], 1998. p. 137–142. Citado na página 15.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. [S.l.], 1995. p. 338–345. Citado na página 28.

JONES, K. S. *Readings in information retrieval*. [S.l.]: Morgan Kaufmann, 1997. Citado na página 17.

JONNALAGADDA, S.; PETITTI, D. A new iterative method to reduce workload in systematic review process. *International journal of computational biology and drug design*, Inderscience Publishers Ltd, v. 6, n. 1-2, p. 5–17, 2013. Citado 3 vezes nas páginas 33, 35 e 37.

JURAFSKY, D. *Speech & language processing*. [S.l.]: Pearson Education India, 2000. Citado 2 vezes nas páginas 15 e 16.

KAUR, G.; CHHABRA, A. Improved j48 classification algorithm for the prediction of diabetes. *International Journal of Computer Applications*, Citeseer, v. 98, n. 22, 2014. Citado na página 29.

KIBRIYA, A. M. et al. Multinomial naive bayes for text categorization revisited. In: SPRINGER. *Australasian Joint Conference on Artificial Intelligence*. [S.l.], 2004. p. 488–499. Citado na página 28.

KITCHENHAM, B. Procedure for undertaking systematic reviews. *Computer Science Depart-ment, Keele University (TRISE-0401) and National ICT Australia Ltd (0400011T. 1), Joint Technical Report*, 2004. Citado na página 1.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering–a tertiary study. *Information and Software Technology*, Elsevier, v. 52, n. 8, p. 792–805, 2010. Citado 3 vezes nas páginas 13, 14 e 44.

KORDE, V.; MAHENDER, C. N. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, Academy & Industry Research Collaboration Center (AIRCC), v. 3, n. 2, p. 85, 2012. Citado 4 vezes nas páginas 13, 15, 16 e 18.

KOTTHOFF, L. et al. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research*, JMLR. org, v. 18, n. 1, p. 826–830, 2017. Citado na página 27.

KOUZNETSOV, A. et al. Classifying biomedical abstracts using committees of classifiers and collective ranking techniques. In: SPRINGER. *Canadian Conference on Artificial Intelligence*. [S.l.], 2009. p. 224–228. Citado na página 34.

KULKARNI, P. *Reinforcement and systemic machine learning for decision making*. [S.l.]: John Wiley & Sons, 2012. v. 1. Citado 2 vezes nas páginas 25 e 26.

LEWIS, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. In: SPRINGER. *European conference on machine learning*. [S.l.], 1998. p. 4–15. Citado na página 28.

LIAW, A.; WIENER, M. et al. Classification and regression by randomforest. *R news*, v. 2, n. 3, p. 18–22, 2002. Citado na página 29.

LUGER, G. F. *Inteligência Artificial-: Estruturas e estratégias para a solução de problemas complexos*. [S.l.]: Bookman, 2004. Citado na página 25.

LUHN, H. P. The automatic creation of literature abstracts. *IBM Journal of research and development*, IBM, v. 2, n. 2, p. 159–165, 1958. Citado na página 21.

MA, Y. *Text classification on imbalanced data: Application to Systematic Reviews Automation*. Tese (Doutorado) — University of Ottawa (Canada), 2007. Citado 3 vezes nas páginas 33, 36 e 37.

MALHEIROS, V. et al. A visual text mining approach for systematic reviews. In: IEEE. *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*. [S.l.], 2007. p. 245–254. Citado na página 34.

MARSHALL, C.; BRERETON, P. Tools to support systematic literature reviews in software engineering: A mapping study. In: IEEE. *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. [S.l.], 2013. p. 296–299. Citado 5 vezes nas páginas 13, 15, 6, 31 e 32.

MARSHALL, C.; BRERETON, P.; KITCHENHAM, B. Tools to support systematic reviews in software engineering: a feature analysis. In: ACM. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2014. p. 13. Citado na página 6.

MCCALLUM, A.; NIGAM, K. et al. A comparison of event models for naive bayes text classification. In: CITESEER. *AAAI-98 workshop on learning for text categorization*. [S.l.], 1998. v. 752, n. 1, p. 41–48. Citado na página 28.

MITCHELL, T. M.; LEARNING, M. Mcgraw-hill science. *Engineering/Math*, v. 1, p. 27, 1997. Citado na página 25.

MIWA, M. et al. Reducing systematic review workload through certainty-based screening. *Journal of biomedical informatics*, Elsevier, v. 51, p. 242–253, 2014. Citado 3 vezes nas páginas 33, 35 e 37.

OCTAVIANO, F.; SILVA, C.; FABBRI, S. Using the scas strategy to perform the initial selection of studies in systematic reviews: an experimental study. In: ACM. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2016. p. 25. Citado 2 vezes nas páginas 34 e 35.

OCTAVIANO, F. R. et al. Semi-automatic selection of primary studies in systematic literature reviews: is it reasonable? *Empirical Software Engineering*, Springer, v. 20, n. 6, p. 1898–1917, 2015. Citado 2 vezes nas páginas 34 e 35.

OLORISADE, B. K. et al. A critical analysis of studies that address the use of text mining for citation screening in systematic reviews. In: ACM. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2016. p. 14. Citado 7 vezes nas páginas 15, 4, 6, 21, 26, 31 e 32.

O'MARA-EVES, A. et al. Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Systematic reviews*, BioMed Central, v. 4, n. 1, p. 5, 2015. Citado 6 vezes nas páginas 4, 6, 31, 32, 33 e 58.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: SN. *12th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2008. v. 17. Citado 4 vezes nas páginas 13, 1, 12 e 14.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, Elsevier, v. 64, p. 1–18, 2015. Citado na página 13.

PLATT, J. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998. Citado na página 28.

PLATT, J. C. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, p. 185–208, 1999. Citado na página 28.

QUINLAN, J. R. Induction of decision trees. *Machine learning*, Springer, v. 1, n. 1, p. 81–106, 1986. Citado na página 29.

QUINLAN, J. R. *C4. 5: programs for machine learning*. [S.l.]: Elsevier, 2014. Citado na página 29.

RADJENOVIĆ, D. et al. Software fault prediction metrics: A systematic literature review. *Information and software technology*, Elsevier, v. 55, n. 8, p. 1397–1418, 2013. Citado na página 44.

RAMAMPIARO, H. et al. Supporting evidence-based software engineering with collaborative information retrieval. In: IEEE. *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*. [S.l.], 2010. p. 1–5. Citado na página 34.

RAMOS, J. et al. Using tf-idf to determine word relevance in document queries. In: *Proceedings of the first instructional conference on machine learning*. [S.l.: s.n.], 2003. v. 242, p. 133–142. Citado na página 20.

RENNIE, J. D. et al. Tackling the poor assumptions of naive bayes text classifiers. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*. [S.l.: s.n.], 2003. p. 616–623. Citado na página 28.

RIAZ, M. et al. Experiences conducting systematic reviews from novices' perspective. In: *EASE*. [S.l.: s.n.], 2010. Citado na página 2.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, Elsevier, v. 24, n. 5, p. 513–523, 1988. Citado na página 20.

SEBASTIANI, F. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, ACM, v. 34, n. 1, p. 1–47, 2002. Citado na página 15.

SHAW, M. Writing good software engineering research papers: minitutorial. In: IEEE COMPUTER SOCIETY. *Proceedings of the 25th international conference on software engineering*. [S.l.], 2003. p. 726–736. Citado na página 12.

SHEMILT, I. et al. Pinpointing needles in giant haystacks: use of text mining to reduce impractical screening workload in extremely large scoping reviews. *Research Synthesis Methods*, Wiley Online Library, v. 5, n. 1, p. 31–49, 2014. Citado 2 vezes nas páginas 33 e 58.

SILVA, M.; MOURA, I.; SOARES, A. Uso de tecnologias computacionais para o ensino de crianças com transtorno do espectro autista: um mapeamento sistemático da literatura. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2017. v. 28, n. 1, p. 173. Citado na página 43.

SIMON, H. A. Search and reasoning in problem solving. *Artif. Intell.*, v. 21, n. 1-2, p. 7–29, 1983. Citado na página 25.

SIMOUDIS, E. Reality check for data mining. *IEEE Intelligent Systems*, IEEE, n. 5, p. 26–33, 1996. Citado na página 15.

TAN, A.-H. et al. Text mining: The state of the art and the challenges. In: SN. *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*. [S.l.], 1999. v. 8, p. 65–70. Citado 2 vezes nas páginas 14 e 15.

TAN, C.-M.; WANG, Y.-F.; LEE, C.-D. The use of bigrams to enhance text categorization. *Information processing & management*, Elsevier, v. 38, n. 4, p. 529–546, 2002. Citado na página 16.

THOMAS, J.; O'MARA-EVES, A. How can we find relevant research more quickly? *NCRM Newsletter: MethodsNews*, 2011. Citado na página 33.

THORNTON, C. *Auto-WEKA: combined selection and hyperparameter optimization of supervised machine learning algorithms*. Tese (Doutorado) — University of British Columbia, 2014. Citado na página 27.

WAHONO, R. S. A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks. *Journal of Software Engineering*, v. 1, n. 1, p. 1–16, 2015. Citado na página 44.

WALLACE, B. C. et al. Active learning for biomedical citation screening. In: ACM. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2010. p. 173–182. Citado 3 vezes nas páginas 33, 36 e 37.

WALLACE, B. C. et al. Deploying an interactive machine learning system in an evidence-based practice center: abstrackr. In: ACM. *proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. [S.l.], 2012. p. 819–824. Citado na página 33.

WALLACE, B. C. et al. Toward modernizing the systematic review pipeline in genetics: efficient updating via data mining. *Genetics in medicine*, Nature Publishing Group, v. 14, n. 7, p. 663, 2012. Citado na página 33.

WALLACE, B. C. et al. Semi-automated screening of biomedical citations for systematic reviews. *BMC bioinformatics*, BioMed Central, v. 11, n. 1, p. 55, 2010. Citado 4 vezes nas páginas 33, 35, 36 e 37.

WIERINGA, R. et al. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, Springer, v. 11, n. 1, p. 102–107, 2006. Citado na página 12.

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 61.

WONG, S. S.-L. et al. Developing optimal search strategies for detecting sound clinical prediction studies in medline. In: AMERICAN MEDICAL INFORMATICS ASSOCIATION. *AMIA Annual Symposium Proceedings*. [S.l.], 2003. v. 2003, p. 728. Citado na página 34.

WU, J. et al. Self-adaptive attribute weighting for naive bayes classification. *Expert Systems with Applications*, Elsevier, v. 42, n. 3, p. 1487–1502, 2015. Citado na página 27.

YU, W. et al. Gapscreener: an automatic tool for screening human genetic association literature in pubmed using the support vector machine technique. *BMC bioinformatics*, BioMed Central, v. 9, n. 1, p. 205, 2008. Citado na página 33.

YU, Z.; KRAFT, N. A.; MENZIES, T. Finding better active learners for faster literature reviews. *Empirical Software Engineering*, Springer, v. 23, n. 6, p. 3161–3186, 2018. Citado 6 vezes nas páginas 36, 37, 43, 44, 58 e 61.

YU, Z.; MENZIES, T. *fastread/src: Core Algorithm Update*. 2017. Disponível em: <https://doi.org/10.5281/zenodo.837861>. Citado na página 36.

YU, Z.; MENZIES, T. Fast2: An intelligent assistant for finding relevant papers. *Expert Systems with Applications*, v. 120, p. 57–71, 2019. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417418307413>. Citado 6 vezes nas páginas 17, 36, 37, 58, 59 e 61.

ZHANG, M.-L.; ZHOU, Z.-H. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, IEEE, v. 26, n. 8, p. 1819–1837, 2014. Citado na página 26.

ZIPF, G. K. Human behavior and the principle of least effort. addison-wesley press, 1949. Citado na página 21.