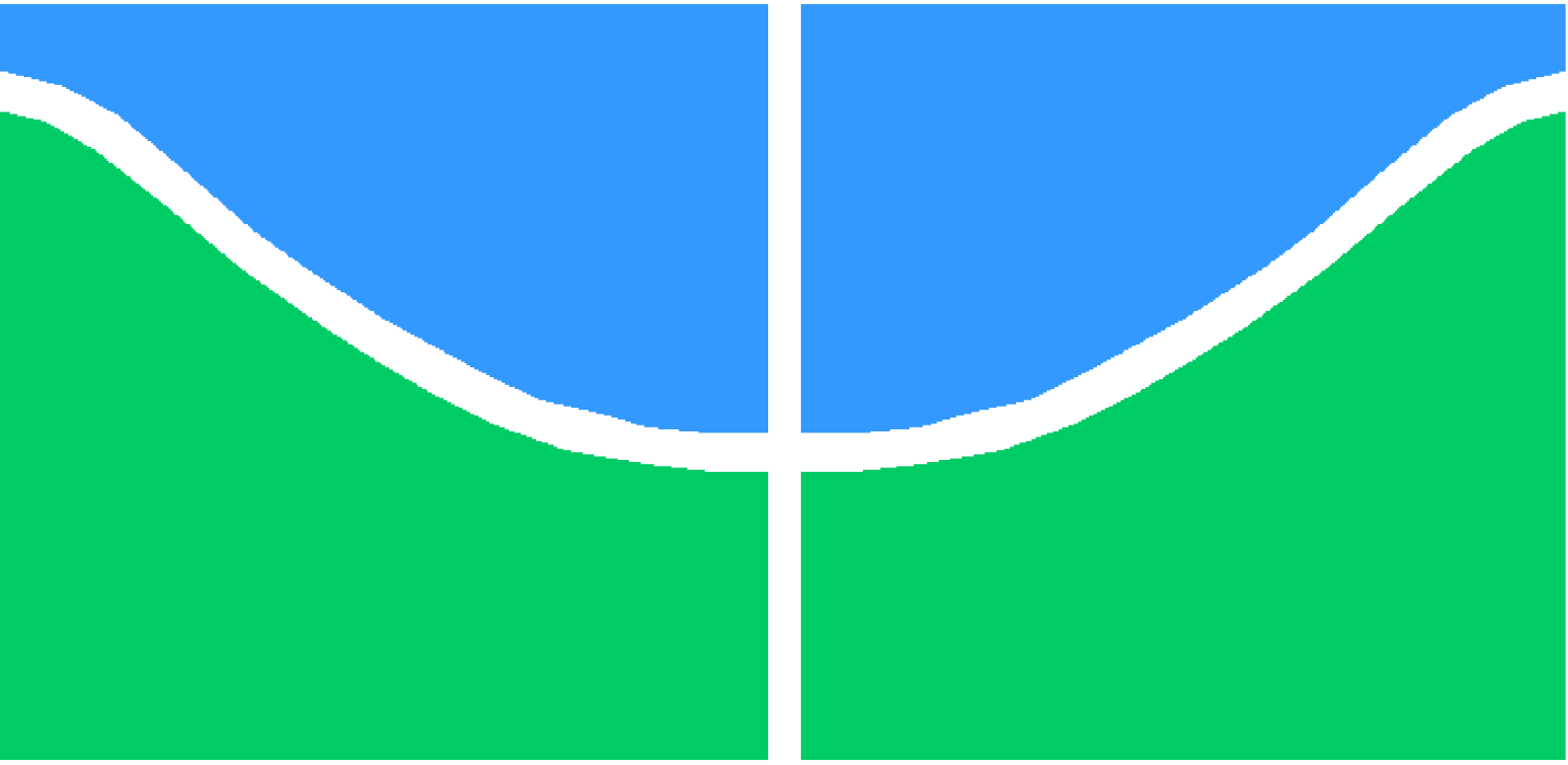

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Elicitação de Requisitos Baseada em Análise de Multidão: Uma Abordagem Orientada a Aprendizado de Máquina

Autor: Gustavo Sabino Mangabeira

Orientador: Profa. Dra. Milene Serrano

Coorientador: Prof. Dr. Maurício Serrano

Brasília, DF

2018

Gustavo Sabino Mangabeira

Elicitação de Requisitos Baseada em Análise de Multidão: Uma Abordagem Orientada a Aprendizado de Máquina

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Profa. Dra. Milene Serrano

Coorientador: Prof. Dr. Maurício Serrano

Brasília, DF

2018

Gustavo Sabino Mangabeira

Elicitação de Requisitos Baseada em Análise de Multidão: Uma Abordagem Orientada a Aprendizado de Máquina/ Gustavo Sabino Mangabeira. – Brasília, DF, 2018-

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

Orientador: Profa. Dra. Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2018.

1. CrowdSource. 2. Machine Learning. I. Profa. Dra. Milene Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Elicitação de Requisitos Baseada em Análise de Multidão: Uma Abordagem Orientada a Aprendizado de Máquina

CDU 02:141:005.6

Gustavo Sabino Mangabeira

Elicitação de Requisitos Baseada em Análise de Multidão: Uma Abordagem Orientada a Aprendizado de Máquina

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 03 de julho de 2018:

Profa. Dra. Milene Serrano
Orientador

Prof. Dr. Maurício Serrano
Coorientador

Prof. Dr. Fabrício Ataides Braz
Convidado 1

Brasília, DF
2018

Este trabalho é dedicado a todos que sonham e persistem

Agradecimentos

Agradeço a minha família, que sempre me ajudou durante essa fase de estudos. Especialmente, à minha mãe, Lindsey Nunes, que sempre procurou meu bem-estar. Agradeço aos meus amigos Rafael Akiyoshi, Francisco Allyson, Macário Soares e Anna Larissa, entre outros que me acompanharam durante os anos de faculdade.

Agradeço também aos meus orientadores, Profa. Dra. Milene Serrano e Prof. Dr. Maurício Serrano, pela dedicação e por acreditar que esse trabalho valeria a pena.

"You pile up enough tomorrows,
and you'll find you are left with nothing
but a lot of empty yesterdays."
(Meredith Willson, The Music Man: A Musical Comedy)

Resumo

Entender corretamente as requisições dos usuários é um dos fatores necessários para o sucesso de um projeto. Atualmente, a comunicação entre usuário e o mantenedor do software está mais acessível, principalmente através de micro-blogs ou lojas de aplicativos. Entretanto, com um número razoável de comentários gerados por essas plataformas, a comunicação, e consequentemente, a elicitação de requisitos com esses usuários torna-se uma atividade impraticável. Visando contribuir para o entendimento das necessidades dos usuários, esse trabalho tem como objetivo propor uma aplicação que identificará, automaticamente, quais os tipos de requisições estão sendo feitas pelos usuários, e quais são as palavras-chave dessas requisições, utilizando *Machine Learning* e *Crowd-Based Requirements Engineering*.

Palavras-chaves: Engenharia de Requisitos de Software, *Crowd-Based Requirements Engineering*, Aprendizado Supervisionado, Aprendizado Não Supervisionado, *Naive Bayes*, *KMeans*.

Abstract

Understanding correctly the user requisitions is one of the factors for a project to succeed. Currently, communication between user and software maintainer is more accessible, especially through micro-blogs or application stores. However, with a huge number of comments generated by these platforms, communication and requirements elicitation with users becomes and impractical activity. In order to contribute to the understanding of users needs, this work aims to propose an application that will automatically identify which types of requests are being made by users and which is the must commented subject, using Machine Learning and Crowd-Based Requirements Engineering.

Key-words: Requirements Engineering, *Crowd-Based Requirements Engineering*, Supervised Learning, UnSupervised Learning, *Naive Bayes*, *KMeans*.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CrowdRE	*Crowd-Base Requirements Engineering*
TCC	Trabalho de Conclusão de Curso
RNF	Requisitos não-funcionais
BOW	*Bag of Words*
TF	*Term Frequency*
IDF	*Inverse Document Frequency*
TF-IDF	*Term Frequency - Inverse Document Frequency*
ML	*Machine Learning*
$P(A\|B)$	Probabilidade de A dado B
KNN	*K-nearest neighbor*
Tp	*True Positive*
Fp	*False Positive*
Tn	*True Negative*
Fn	*False Negative*
NLTK	*Natural Language Toolkit*
API	*Application Programming Interface*
REST	*Representational State Transfer*
SADT	*Structured Analysis and Design Technique*

Sumário

1 Introdução

Neste capítulo, será descrito o contexto, no qual este trabalho está envolvido. O capítulo está organizado em Seções: Na Seção 1.1, é apresentado o contexto; Na Seção 1.2, é descrita a questão de pesquisa; Na Seção 1.3, as justificativas. Na Seção 1.4, é descrito o objetivo geral e, adicionalmente, os objetivos específicos. E por fim, na Seção 1.5, é apresentada a organização dessa monografia.

1.1 Contextualização

Requisitos de software são descrições e limitações do que o sistema e seus serviços devem fornecer. Essas limitações refletem as necessidades dos *stakeholders* durante o projeto. Requisitos podem ser separados de acordo com seu detalhamento. Descrições com alto nível de abstração e escritas em linguagem natural são chamadas de Requisitos de Usuário. Já descrições detalhadas sobre o sistema são Requisitos do Sistema (SOMMERVILLE, 2011).

Requisitos de software são comumente classificados entre Funcionais e Não-funcionais. Requisitos funcionais são afirmações sobre o funcionamento, os serviços e o comportamento esperados do sistema. É esperado que a descrição de um requisito funcional seja mais completa e consistente possível. Isso significa que todos os serviços desejados pelo usuário devem ser definidos e não contraditórios (SOMMERVILLE, 2011) .

Requisitos não-funcionais são características e restrições aplicadas às suas funcionalidades, sendo, geralmente mais críticos que os requisitos funcionais. Essa criticidade é justificada, pois os Requisitos Não-funcionais são mais abstratos em termos de especificação. Por exemplo, em um requisito não-funcional de segurança tem-se que: o que é seguro em um sistema, pode não ser para outros. Adicionalmente, erros na especificação podem tornar as funcionalidades existentes inutilizáveis (SOMMERVILLE, 2011).

Para obter as informações necessárias, é preciso realizar a atividade de elicitação. Durante esse processo, os profissionais da organização, como engenheiros e desenvolvedores, trabalham juntos com os usuários para obter informações a respeito do sistema (GUERSES et al., 2005). Diferentes *stakeholders*, como os patrocinadores, os clientes, os gestores e os usuários têm opiniões diferentes a respeito do produto. Entretanto, todos os grupos são fontes de informação importantes (KUJALA et al., 2005).

A literatura, (KEIL; CARMEL, 1995) e (KUJALA, 2003), mostra que a interação direta de usuários e desenvolvedores, sem agentes intermediários, contribui para efeitos

positivos no sucesso da aplicação e na satisfação do usuário. Métodos como introspecção, entrevistas, questionários, análise de protocolos e *brainstorming* são técnicas conhecidas para envolver *stakeholders* durante o processo de elicitação (GOGUEN; LINDE, 1993) e (GUNDA, 2008).

Entretanto, com o aumento da demanda por software (BASOLE; KARLA, 2011) e um nível de competição sem precedentes, fábricas de software são encorajaradas a buscarem meios, além dos tradicionais, para se conectarem com os usuários finais de uma maneira mais efetiva (WILLIAMS; MAHMOUD, 2017). (GROEN et al., 2017) cita que os métodos tradicionais perdem a oportunidade de envolver um número grande de usuários. Tais métodos demandam muito tempo e esforço da equipe, tornando-os inadequados para produtos de software que precisam evoluir rapidamente, no intuito de não perder oportunidades de mercado (CARREñO; WINBLADH, 2013).

Para suprir a limitação de engajar muitos usuários, técnicas avançadas de engenharia de requisitos, como *Crowd-Base Requirements Engineering* (CrowdRE), surgem. CrowdRE é um termo utilizado para descrever abordagens automatizadas de coleta e análise de informações vindas da multidão, visando obter requisitos de usuários validados (GROEN; DOERR; ADAM, 2015). CrowdRE mobiliza o maior número de usuários possível para discutir suas necessidades, sendo essa comunicação chamada de *user-feedback* (GROEN et al., 2017).

User-feedback tem como base a percepção do usuário sobre a experiência durante o uso do software. Essa comunicação pode ser expressa implicitamente, através do comportamento do usuário, o qual pode utilizar o software constantemente ou deixar de usar por completo. Pode ainda ser explícita, expressa como um comentário ou um relatório de erro (MORALES-RAMIREZ; PERINI; GUIZZARDI, 2015).

Requisitos de software, através da opinião de multidão, têm o potencial de aumentar a qualidade e a compreensão dos envolvidos (HOSSEINI et al., 2014), uma vez que permite a participação de interessados, principalmente, aqueles que costumam ter o papel mais marginalizado no processo de elicitação: os usuários(GROEN et al., 2017). (HOSSEINI et al., 2014) mostra que informações da multidão possuem diversas características e qualidades, como por exemplo: possuir vasto número de informações, conferindo maior precisão e menos informações tendenciosas; garantindo diversidade de fonte de informação, e contribuindo para as informações terem maior relevância.

Segundo (GROEN et al., 2017), metodologias tradicionais de elicitação têm problemas quando tratam de requisitos com um número grande de usuários. Tal atividade consome tempo e, consequentemente, dinheiro. Para ajudar no entendimento dos requisitos e aproveitar o benefício do envolvimento de usuários, soluções que utilizam *feedback* do usuário, como propostos por (GROEN et al., 2017), sugerem buscar por informações em plataformas de distribuição de aplicações como Apple Store®, Google Play® e Amazon

Appstore®.

Essas plataformas permitem que os usuários expressem suas satisfações com determinado software, através de comentários. Possibilitam ainda que esses usuários deem informações a respeito da qualidade do software (PAGANO; BRUEGGE, 2013), proponham novas funcionalidades (PAGANO; MAALEJ, 2013), e reportem erros nos aplicativos. Por essa razão, os comentários mostram-se uma valiosa fonte de informação para determinar novos requisitos de software em futuras implementações e evoluções (GUZMAN; MAALEJ., 2014).

Entretanto, mesmo ao utilizar plataformas como as citadas anteriormente para obter opiniões, o tempo, os recursos humanos e o próprio viés do engenheiro de requisitos mostram-se fatores necessários para a criação de uma técnica capaz de classificar automaticamente os desejos do usuário em requisitos.

Uma das tarefas mais importantes na Engenharia de Requisitos é a classificação dos requisitos encontrados (LI et al., 2017). Requisitos bem categorizados contribuem para melhorar a qualidade da especificação (POHL; RUPP, 2011). Muito esforço é alocado para categorizar esses requisitos e uma abordagem automática faz-se necessária (LI et al., 2017). Durante a elicitação de requisitos através da multidão, podem existir de centenas a milhares de *stakeholders* e o número de requisitos coletados pode ser imenso (LI et al., 2017). (LI et al., 2017) e (CLELAND-HUANG et al., 2006) propoem o uso de processamento de linguagem natural e aprendizado de máquina para identificar e classificar requisitos vindos dos usuários.

1.2 Questão de Pesquisa

Este trabalho pretende responder a seguinte questão: Há a possibilidade de auxiliar gerentes de software na priorização de funcionalidades e de melhorias utilizando grande quantidade de informações?

1.3 Justificativa

De acordo com (THE STANDISH GROUP REPORT, 2015), uma das razões que um projeto será bem sucedido é o envolvimento dos usuários. Esse mesmo relatório mostra que a falta de envolvimento do usuário é também uma das principais razões do fracasso do projeto junto com a falta de especificação adequada dos requisitos de software. Metodologias de desenvolvimento e gestão de software como *eXtreme programming* (WELLS, 1996) e *Scrum* (SCHWABER; SUTHERLAND, 2017) visam aumentar o envolvimento do usuário, colocando-o diretamente em contato com o time de desenvolvimento e pro-

curando obter *feedback* a cada iteração. A intenção é reduzir os erros causados por mau entendimento durante a fase de elicitação. Contudo, durante o ciclo de vida do software, suas evoluções dependem altamente do *feedback* dos usuários do software, não somente dos investidores e gestores. É nesse momento que os comentários realizados nas plataformas de distribuição de aplicações mostram-se poderosas ferramentas para priorizar solicitações que mais agregam valor ao usuário final. Uma vez que usuários não são treinados, suas requisições são escritas em linguagem natural e informal (LI et al., 2017). Dessa forma, seria um trabalho árduo classificar as milhares de requisições obtidas pelos comentários. Com a intenção de apoiar essa classificação bem como ajudar gerentes de software a tomar decisões, este trabalho visa criar um classificador automático de requisições de usuário com base em regras de processamento de linguagem natural, definições das classes de requisitos não-funcionais e algoritmos de aprendizado de máquina.

1.4 Objetivo Geral

Visando auxiliar os gerentes de software, principalmente na priorização de funcionalidades e de melhorias dada uma grande quantidade de informações envolvidas nesse processo, criar um classificador de requisições de usuários, expressas em linguagem natural, automatizando essa solução com processamento de linguagem natural e algoritmos de aprendizado de máquina. Como essas requisições podem ser funcionalidades ou critérios de qualidade, adicionalmente, há a necessidade de classificar essas requisições de usuários em requisitos funcionais e requisitos não-funcionais.

1.4.1 Objetivos Específicos

Objetivos específicos foram definidos para auxiliar na obtenção do objetivo geral, sendo eles:

1. Identificar regras para pré-processar os dados coletados;
2. Propor um processo ou um fluxo de atividades para pré-processamento de dados;
3. Identificar um algorítmo de *Machine Learning* para utilização no trabalho;
4. Classificar algumas avaliações para cada classe de requisitos selecionada para o trabalho. O número de avaliações será definido ao longo da pesquisa;
5. Desenvolvimento da aplicação proposta, e
6. Coletar métricas de desempenho do classificador.

1.5 Organização do Documento

Este documento está dividido em capítulos, sendo eles:

Capítulo 2 - Referencial Teórico: apresenta as áreas e os conceitos estudados para o desenvolvimento do trabalho;

Capítulo 3 - Referencial Tecnológico: apresenta as tecnologias envolvidas no desenvolvimento desse trabalho;

Capítulo 4 - Metodologia: apresenta a metodologia de pesquisa e desenvolvimento adotada para esse trabalho;

Capítulo 5 - Proposta: apresenta a solução proposta para a questão de pesquisa levantada anteriormente;

Capítulo 6 - Resultados: apresenta os resultados alcançados no presente trabalho, considerando, principalmente, os objetivos específicos e o objetivo geral.

Capítulo 7 - Considerações Finais: apresenta a conclusão desse trabalho. Adicionalmente, são acordados os aspectos chave da pesquisa, sua contribuição e a possibilidade de trabalhos futuros.

2 Referencial Teórico

Neste capítulo, são apresentadas as bases teóricas que apoiam este trabalho. O capítulo está organizado em Seções, sendo: na Seção 2.1, são abordados os Requisitos de software e suas definições; na seção 2.2, são apresentadas as técnicas de Elicitação de Requisitos de Software; na seção 2.3, é descrito o conceito de *CrowdSourcing Requirements*, um novo modelo de Elicitação de Requisitos de Software; na Seção 2.4, aborda-se o Processamento de Linguagem natural, e na Seção 2.5 são introduzidos o conceito de Aprendizado de máquina e os principais algoritmos utilizados em problemas envolvendo texto. Por fim, são apresentadas as considerações finais do capítulo.

2.1 Requisitos de Software

Segundo (SOMMERVILLE, 2011), todos os processos de software devem incluir quatro atividades fundamentais da engenharia de software. Dentre essas atividades, tem-se a Especificação de Software.

Especificação de Software, ou Engenharia de Requisitos, é o processo de entender e definir quais serviços são exigidos do sistema. Essa etapa é crítica, uma vez que a interpretação errada pode levar a problemas durante a fase de implementação. A Engenharia de Requisitos possui quatro atividades principais (SOMMERVILLE, 2011):

1. Estudo de Viabilidade

   Uma estimativa é realizada para identificar se o usuário está satisfeito com a tecnologia atual. Esse estudo considera se o sistema a ser construído terá uma relação boa de custo-benefício e dentro dos limites financeiros da empresa.

2. Elicitação e Análise de Requisitos

   Durante esta fase, a observação dos produtos de software concorrentes bem como conversas com usuários em potencial são realizadas para que sejam obtidos requisitos de software derivados. Essa atividade ajuda a entender o sistema a ser especificado.

3. Especificação de Requisitos

   É a atividade de transformar a informação obtida durante os processos anteriores em documentos que definem o conjunto de requisitos.

4. Validação dos Requisitos

É o momento de verificar a veracidade, consistência e completude dos requisitos. Durante esse processo, são feitas mudanças de correção e adaptação das descrições.

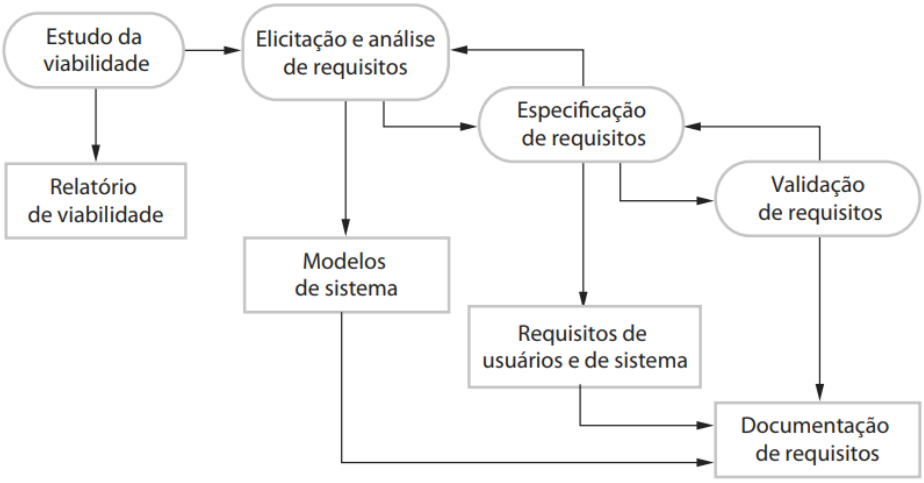

Figura 1 – Ciclo de Engenharia de Requisitos (SOMMERVILLE, 2011)

Ao longo do processo de Engenharia de Requisitos, podem surgir problemas que são resultados de não realizar uma separação clara entre descrições de usuário e descrições de sistema, sendo essas duas separações básicas definidas por (SOMMERVILLE, 2011):

- Requisitos de Usuário: São declarações em linguagem natural sobre quais serviços e limitações são esperados que o sistema provenha para o usuário, e
- Requisitos de Sistema: São descrições mais detalhadas das funcionalidades e limites do sistema. A especificação de requisitos deve descrever exatamente o que deve ser implementado.

2.1.1 Requisitos Funcionais

Requisitos Funcionais são descrições sobre o que o sistema deveria fazer. Esses requisitos dependem do tipo de software que está sendo desenvolvido, da expectativa dos usuários e da abordagem adotada pela organização. Quando escritos como requisitos de usuário, os requisitos funcionais são abstraídos de maneira que possam ser facilmente entendidos por diversos *stakeholders* (SOMMERVILLE, 2011).

Requisitos Funcionais podem descrever como o sistema deve reagir a determinadas entradas, ou até mesmo descrever a maneira que os usuários que utilizarão o software devem agir quando interagirem com o sistema (SOMMERVILLE, 2011).

2.1.2 Requisitos Não-Funcionais

Requisitos não-funcionais (RNF) são especificações que não estão diretamente concentradas em descrever os serviços para seus usuários. Entretanto, os RNF descrevem as características de implementação, tais como segurança, usabilidade e desempenho. Geralmente, são tidos como mais críticos na aplicação, pois a implementação inadequada desses requisitos impossibilita o uso correto da aplicação, mesmo se as funcionalidades estiverem corretas. Como mostra a Figura 2, Requisitos Não-funcionais podem surgir de diversas fontes (SOMMERVILLE, 2011), tais como:

1. *Requisitos de Produto* especificam o comportamento do sistema, como desempenho, segurança, usabilidade e taxa de erro tolerável;
2. *Requisitos Organizacionais* especificam características que definem as políticas e os procedimentos adotados pelas organizações nas quais o software irá atuar, e
3. *Requisitos Externos* cobrem todas as demais especificações não descritas anteriormente. Geralmente, abrangem casos, nos quais o software precisa obedecer uma série de regulações.

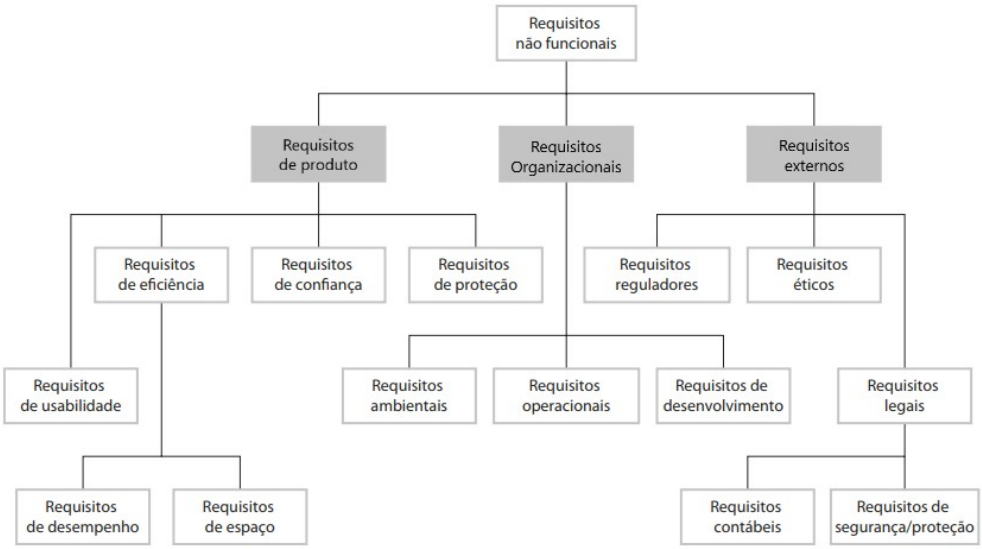

Figura 2 – Tipos de Requisitos Não-Funcionais - (SOMMERVILLE, 2011)

2.2 Elicitação de Requisitos de Software

A fase de elicitação de requisitos é a primeira do processo de Engenharia de Requisitos. Durante essa fase, as informações são coletadas, interpretadas, analisadas, modeladas

e validadas. Um dos objetivos mais importantes da elicitação é descobrir qual problema precisa de solução, e portanto, identificar os limites do sistema tanto como seus usuários (NUSEIBEH; EASTERBROOK, 2000).

Para auxiliar o processo de identificação, existem diversas técnicas, tais como as apresentadas nas seções 2.2.1 a 2.3.

2.2.1 Entrevistas

Entrevista é o método mais popular de elicitação de requisitos de software. Nesse método, o Engenheiro de Requisitos fica encarregado de conversar com diferentes *stakeholders* para entender as necessidades do sistema. Tipicamente, entrevistas dividem-se em: entrevistas fechadas, que necessitam certo nível de preparação anterior com perguntas predefinidas; entrevistas abertas, que o entrevistador tenta extrair informação do *stakeholder* sem um roteiro pré-definido (GUNDA, 2008); e estrevistas semi-estruturadas que combinam perguntas abertas e fechadas, onde o informante tem a possibilidade de discorrer sobre o tema proposto (BONI; QUARESMA, 2005).

2.2.2 Questionários

Questionários são amplamente utilizados e possuem o benefício de terem apelo científico devido ao uso, por exemplo, de análise estatística (GOGUEN; LINDE, 1993). Adicionalmente, costumam custar menos e podem ser aplicados para um maior número de pessoas (GUNDA, 2008). O resultado da aplicação de um questionário depende, principalmente, de dois fatores: boa estruturação do questionário e honestidade das respostas.

Os dados coletados por questionários podem ser usados para analisar os resultados de modo sistemático e quantitativo. (BURGESS, 2001) define os processos básicos de uma pesquisa feita por questionário. Uma breve descrição é apresentada a seguir:

1. **Definir os propósitos de pesquisa:** é realizada uma pesquisa na literatura para saber alguns aspectos preliminares do assunto que se deseja pesquisar;

2. **Identificar a população:** define-se a parcela da população, a qual o questionário será realizado, o público-alvo;

3. **Decidir como coletar as respostas:** Decide-se qual será a maneira que o questionário será aplicado. Seja por meio de uma entrevista com auxilio de um pesquisador ou adota-se a liberdade do público de responder, por *email*, por carta ou outros meios. Para ambos os casos, deve-se tentar convencer os participantes a responder o questionário;

4. **Estruturar o questionário:** são realizadas três atividades para estruturar o questionário, sendo: determinar as questões que serão respondidas; escolher quais serão os tipos de cada questão, se serão abertas, ou fechadas, ou de múltipla escolha, e então, definir qual a ordem das questões bem como o *layout* geral;

5. **Executar um teste piloto:** é realizado um teste piloto com uma pequena população. O objetivo é identificar falhas, corrigindo-as antes de executar a pesquisa em grande escala;

6. **Executar a pesquisa verdadeira:** é realizada a pesquisa verdadeira. Adicionalmente, são definidos responsáveis bem como é realizado o acompanhamento dos questionários que aplicados, e

7. **Analisar os dados coletados:** é feita a análise dos dados, preferencialmente, com uso de software;

2.2.3 Prototipação

Prototipação é a representação visual de componentes do sistema que está sendo construída. O protótipo provê uma ideia geral de como será o uso do sistema pelo *stakeholder*, sendo utilizado para obter requisitos a partir da perspectiva de usabilidade (GUNDA, 2008). Protótipos têm a vantagem de tornar ideias e conceitos intangíveis em especificações tangíveis, além de permitir o rápido *feedback* sobre os requisitos coletados para criar essa representação (MCCLENDON; REGOT; AKERS, 2012). Apesar de promover o envolvimento do usuário e facilitar o entendimento dos requisitos, protótipos podem custar muito tempo, e dinheiro, para serem desenvolvidos.

Técnicas como a introspecção, o grupo focal, a discussão e a análise de protocolos são apresentadas detalhadamente por (GOGUEN; LINDE, 1993), e podem ajudar na elicitação de requisitos. Mais modernamente, e considerando a necessidade de prover software para um grande número de usuários, com diferentes pontos de vista, demandas e preferências, surgem estudos mais emergentes como *CrowdSourcing Requirements*. Tal tópico é debatido na Seção 2.3 desse capítulo, sendo tema e foco de estudo no presente trabalho.

2.3 CrowdSourcing Requirements

Crowdsourcing descreve um novo modelo de negócio que aproveita as soluções criativas de uma rede distribuída de indivíduos. Com grupos motivados, a multidão é capaz de solucionar problemas com qualidade e quantidade superior até mesmo em comparação com times específicos de negócios tradicionais (BRABHAM, 2008).

Adicionalmente, *crowdsourcing* permitiu que uma nova abordagem de Engenharia de Requisitos, *CrowdRE*, surgisse. *CrowdRE* descreve atividades de coleta e análise automática de informações da multidão, permitindo transformar essas informações em requisitos de software válidos (GROEN et al., 2017). A multidão, nesses casos, compreende um grande número de usuários, ou potenciais usuários. Tal abordagem permite que dados sejam coletados mais facilmente, uma vez que não é mais necessária a presença física dos *stakeholders* (LEVY; HADAR; TE'ENI, 2015).

Apesar da ausência dos *stakeholders*, ainda existe comunicação entre o Engenheiro de requisitos e os *stakeholders*. (GROEN et al., 2017) chama essa comunicação de *feedback* do usuário, a qual é explicada na Seção 2.3.1

2.3.1 *Feedback* do Usuário

A comunicação entre os desenvolvedores e os usuários do software é chamada de *feedback*. O *feedback* pode ser expresso de diversas formas, entre as mais comuns estão: a documentação linguística, da opinião do usuário, e a documentação não-linguística, da opinião do usuário. Uma breve descrição dessas formas é apresentada a seguir:

Documentação Linguística consiste na disponibilização de sua opinião através de textos em linguagem natural e áudios (GROEN et al., 2017), e

Documentação Não-Linguística consiste na opinião expressa em formatos mais abstratos como: imagens, *emoticons* e avaliações, sendo representados, por exemplo, com número de estrelas e quantidade de *curtidas* no *Facebook*® (MORALES-RAMIREZ; PERINI; GUIZZARDI, 2015).

CrowdRE depende de uma quantidade suficientemente grande de *feedback* dado pelos usuários. Essa quantidade é suprida, uma vez que atualmente há diversos canais de comunicação disponíveis para o usuário, como por exemplo: a sessão de comentários na *AppStore*®, na *Google Play*® e na *Amazon AppStore*®, bem como nas redes sociais *Facebook*® e *Twitter*®.

2.3.2 Oportunidades Através de *CrowdRE*

(HOSSEINI et al., 2014) mostra alguns dos dados que podem ser coletados através do *Crowdsourcing*. Esses dados podem ser vistos como oportunidades conquistadas através do uso de *CrowdRE*.

2.3.2.1 Descoberta de Requisitos

Atualmente, produtos de software, como as aplicações de *smartphones*, possuem um grande grupo de usuários. Tais usuários possuem perfis diversos, o que dificulta o entendimento das funcionalidades e dos atributos que devem ser atendidos para determinado software. A fim de aumentar o desempenho durante a elicitação, e criar produtos de software capazes de atender tal demanda, pode-se utilizar o entendimento da multidão durante a fase de elicitação (HOSSEINI et al., 2014).

2.3.2.2 Validação Empírica

Validação do software junto aos usuários de teste é uma atividade que demanda tempo e pode levar a resultados que são válidos apenas temporariamente, especialmente em sistemas muito dinâmicos. Os usuários podem não dividir as mesmas opiniões com o passar do tempo, especialmente quando outras soluções de software competitivas aparecem ou há o uso do software em um contexto, o qual não foi pensado inicialmente (HOSSEINI et al., 2014).

2.3.2.3 Adaptação Social Orientada por Requisitos

Identificar se o software é capaz de atender às necessidades que seus usuários requerem é uma atividade que demanda tempo. *CrowRE* pode aliviar tal necessidade ao analisar a percepção dos usuários sobre produtos de software alternativos. A capacidade e a qualidade de realizar tal tarefa são medidas e, então são propostas mudanças de adaptação (HOSSEINI et al., 2014).

2.3.2.4 Engenharia de Requisitos Baseada em *Feedback* do Usuário

(PAGANO; MAALEJ, 2013) mostram a importância do *feedback* do usuário que realiza comentários na plataforma de distribuição da aplicação. *Feedback* permite aos desenvolvedores entenderem mais adequadamente as necessidades do usuário para uma próxima versão do sistema (HOSSEINI et al., 2014).

2.3.2.5 Descoberta de *Stakeholders*

Engajar uma grande quantidade de usuários no processo de Engenharia de Requisitos é um desafio enfrentado pelos métodos de elicitação tradicionais. Essas abordagens perdem a oportunidade de envolver um grupo grande e heterogêneo de usuários que estão dispostos a dar *feedback* sobre a aplicação considerando diferentes meios (GROEN et al., 2017). Em sistemas complexos, é díficil identificar todos os *stakeholders*, seus papéis,

suas experiências e suas solicitações. *CrowdRE*, nesses casos, pode ajudar a identificar, e compreender, essas dificuldades (HOSSEINI et al., 2014).

Uma vez que o processamento manual dos artefatos gerados pelos *feedbacks* é uma atividade dispendiosa, surgem técnicas para auxiliar a coleta e análise dos dados, algumas, das quais, são apresentadas a seguir.

2.4 Processamento de Linguagem Natural

Segundo (MANNING; RAGHAVAN; SCHüTZE, 2008), Recuperação de Informação é a ação de achar conteúdo através de documentos não-estruturados que satisfazem às necessidades de informação, dada uma grande coleção de documentos. Com um número imenso de dados sendo produzido diariamente pelo uso de computadores e *smartphones*, tal atividade é necessária para facilitar a busca por informação em texto não-estruturado.

De acordo com (MANNING; RAGHAVAN; SCHüTZE, 2008), para realizar a Recuperação de Informação de maneira adequada, deve-se adotar passos, chamados de pré-processamento. Alguns passos são descritos a seguir:

2.4.1 Tokenização

"Tokenização"é definido como a separação do documento em pedaços, chamados de *tokens* (MANNING; RAGHAVAN; SCHüTZE, 2008). Durante a "tokenização", sinais de pontuação podem ser removidos, como no exemplo a seguir:

1. Entrada: *'Maria gosta de biscoitos!'*
2. Frase após "tokenização": *['Maria','gosta','de','biscoitos']*

2.4.2 Remoção de Termos Comuns: *Stop words*

Algumas palavras são extremamente comuns na linguagem. Entretanto, essas palavras não adicionam valor quando se deseja procurar por informações nos documentos. Essas palavras são chamadas de *stop words* (MANNING; RAGHAVAN; SCHüTZE, 2008). Remover as *stop words* permite a identificação de termos mais relevantes para a análise do documento. Como mostra (LI et al., 2017), é importante estudar a coleção de documentos que será utilizada para identificar a lista de *stop words* necessária para remoção, uma vez que frases podem perder o sentido após sua remoção. Um exemplo de *stop words* pode ser visto na Figura 3.

a	ao	aos
aquela	aquelas	aqueles
aquilo	as	até

Figura 3 – Exemplo de *stop words* para a Língua Portuguesa

2.4.3 *Stemming*

Stemming é uma das técnicas utilizadas no pré-processamento de documentos. O processo consiste na redução das derivações e variações das palavras para que se obtenha apenas sua raiz. Essa redução é feita por meio da remoção do prefixo e do sufixo de um termo. Essa abordagem permite que informações mais relevantes sejam extraídas dos documentos (BALAKRISHNAN; LLOYD-YEMOH, 2010).

2.4.4 *Bag of Words* (BOW)

Uma das maneiras mais comuns de classificação de texto é chamada de *Bag of Words*. Essa técnica produz um dicionário com todos os termos da coleção e calcula se um termo está presente nesse dicionário, e qual a frequência desse termo. De acordo com (MANNING; RAGHAVAN; SCHüTZE, 2008), o *Bag of Words* preocupa-se somente com as ocorrências de cada termo, de forma a considerar a frase *Maria é mais rápida que João* similar em termos de conteúdo com *João é mais rápido que Maria*. A vantagem do *Bag of Words* é a criação automática de uma lista com as palavras-chave dos documentos (MAALEJ; NABIL, 2015). Um exemplo do *Bag of Words* para a fase: *"Adorei o Aplicativo, nota 10"*, pode ser visto na Figura 4.

Adorei	o	Aplicativo	nota	10
1	1	1	1	1

Figura 4 – *Bag of Words*

A técnica do *Bag of Words* considera todos os termos igualmente importantes. Isso significa que os termos possuem pouco, ou nenhum, poder de discriminação. Essa discriminação é um fator importante para determinar a relevância de uma palavra na coleção de documentos (MANNING; RAGHAVAN; SCHüTZE, 2008).

2.4.5 *Term Frequency - Inverse Document Frequency* (TF-IDF)

Diferente do *Bag of Words*, *Term Frequency - Inverse Document Frequency* (TF-IDF) utiliza pesos e medidas estatísticas para medir o quão importante é um termo para uma coleção de documentos. TF-IDF é calculado como o produto de dois termos,

obtidos da seguinte maneira: primeiro é calculada a Frequência dos Termos (TF), e depois, calculada o Inverso da Frequência nos Documentos (IDF), como representado na equação 2.1

$$TF - IDF(w, d) = TF(w, d) * IDF(W) \tag{2.1}$$

Frequência dos Termos é calculada somando o número de vezes que um termo aparece em um documento.

$$TF(w, d) = \frac{F}{W_d} \tag{2.2}$$

Onde F é o número de vezes que uma palavra W aparece no documento D e W_d é o número de palavras no documento D (LI et al., 2017).

Inverso da Frequência nos Documentos é calculado para determinar o peso de um termo dada uma coleção, uma vez que nem todos os termos fornecem relevância para a análise. O peso calculado pelo IDF é proporcional para sua frequência. Tem valor alto para termos que aparecem raramente na coleção, e valor baixo para termos mais frequentes. IDF é representado na equação 2.3:

$$IDF = log(\frac{N}{df_t}) \tag{2.3}$$

Onde N é o número de documentos na coleção, e df_t é o número de documentos que possuem o termo t.

2.4.6 *N-Gram*

Segundo (GUTHRIE et al., 2006), o uso do *N-Gram* proporciona maior entendimento do contexto das palavras nos documentos. É possivel utilizar esse fato para identificar documentos que têm conteúdos similares. O *N-Gram* representa o conjunto de uma sequência de palavras co-ocorrentes, dada uma distância N para um texto qualquer. Para a frase *'Um belo dia'* e $N = 2$, os *N-Gramas* correspondentes seriam:

1. *'Um belo'*
2. *'belo dia'*

Nesse trabalho, foram utilizadas as técnicas apresentadas anteriormente para pré-processar os dados que serão apresentados no Capítulo 5. Adicionalmente, serão utilizado algoritmos de Aprendizado de Máquina, os quais são apresentados na seção 2.5.

2.5 Aprendizado de Máquina

De acordo com (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012), Aprendizado de Máquina, *Machine Learning* (ML) no inglês, são métodos computacionais que utilizam a experiência para realizar predições com certo grau de precisão. No contexto do ML, experiências são informações coletadas anteriormente e que estão à disposição para análise. ML consiste em arquitetar, com eficiência e precisão, algoritmos de predição.

Algoritmos de ML já são implementados com sucesso para uma variedade de problemas, inclusive problemas de análise de documentos, que é o problema apresentado nesse trabalho. Algumas das principais classes de problemas de aprendizagem são acordadas a seguir:

Classificação: envolve a atribuição de uma categoria a um item. Por exemplo, pode-se classificar uma notícia em uma categoria como política, esportes ou negócios;

Regressão: envolve predizer qual o valor de um item. Exemplo de regressão clássica é determinar o valor de uma casa com base na sua localidade, tamanho e número de quartos, e

Clusterização: identifica grupos homogêneos com base nas suas características. É comum para analisar um número grande de dados e identificar comunidades.

Dois conceitos fundamentais do ML são o conjunto de treino e o conjunto de teste, apresentados a seguir:

Conjunto de Treino é o conjunto de dados, o quão já é conhecido o resultado verdadeiro que o classificador deveria retornar. Esse conjunto é utilizado para realizar a parametrização dos classificadores, e

Conjunto de Teste é o conjunto de dados, no qual já é conhecido o resultado da predição, resultados os quais o classificador nunca teve acesso. Esse conjunto será utilizado para validar estatísticamente os resultados após a classificação.

2.5.0.1 Cenários de Aprendizado

Existem vários cenários no contexto do *Machine Learning*. Esses cenários diferem-se de acordo com o tipo de dado disponível para treinamento do modelo. Alguns dos cenários mais comuns são descritos, brevemente, a seguir:

Aprendizado Supervisionado: o algoritmo recebe um conjunto de dados de treino, os quais já possuem identificadores. Esse conjunto de treino já marcado permite ao algoritmo realizar predições para dados que nunca foram vistos e que não possuem identificadores. É o cenário mais comum do ML, sendo, portanto, bem aceito para a

solução dos problemas de classificação e regressão apresentados anteriormente, e

Aprendizado Não-Supervisionado: o algoritmo recebe um conjunto de dados não-identificados e, então, realiza premissas para todos os pontos não-vistos. É mais complexo avaliar o resultado produzido por essa abordagem. *Clustering* é um exemplo de aprendizado não-supervisionado.

Este trabalho, como é apresentado no Capítulo 5, tem como foco o cenário do Aprendizado Supervisionado.

2.5.1 Algoritmos de Classificação

A seguir são descritos alguns dos algoritmos de classificação usados em problemas que envolvem documentos e processamento de linguagem natural.

2.5.1.1 Classificador *Naive Bayes*

Entre as técnicas para solucionar os problemas de classificação em ML, está o *Naive Bayes*. *Naive Bayes* é uma técnica comum de classificação de texto na literatura ((LEWIS, 2005), (MAALEJ; NABIL, 2015), (GUZMAN; EL-HALABY; BRUEGGE, 2015)), sendo derivada do teorema de Bayes, conforme pode ser observado na equação 2.4. *Naive Bayes* assume que todos os atributos de um preditor são independentes. Essa suposição é chamada de independência condicional.

Dado o teorema de Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.4}$$

Onde $P(A|B)$ é a probabilidade de um A acontecer, uma vez que B é verdade. $P(B|A)$ é a probabilidade de B acontecer, uma vez que A é verdade. $P(A)$ e $P(B)$ são probabilidades de A e B acontecerem, independentemente um do outro.

Agora, assume-se que todos os atributos são independentes dado o valor de A. Assim temos:

$$f_{nb}(B) = p(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} p(x_i|A) \tag{2.5}$$

Onde $f_{nb}(B)$ é o classificador de *Naive Bayes* (ZHANG, 2004).

2.5.1.2 *K-nearest Neighbor* (KNN)

K-nearest neighbor é um algoritmo que realiza uma operação de classificação ao utilizar o conjunto de treino disponível para achar exemplos com a menor distância do objeto que se deseja classificar (PEDREGOSA et al., 2017). O algoritmo de *K-nearest neighbor* realiza essa operação calculando a distância entre vetores (ALFONS; LIND, 2009) para K vizinhos, onde K é o número de vizinhos, com os quais é desejado realizar a comparação. Ressalta-se que esse é definido por experimentação. Os efeitos da seleção de uma contante K podem ser vistos na Figura 5.

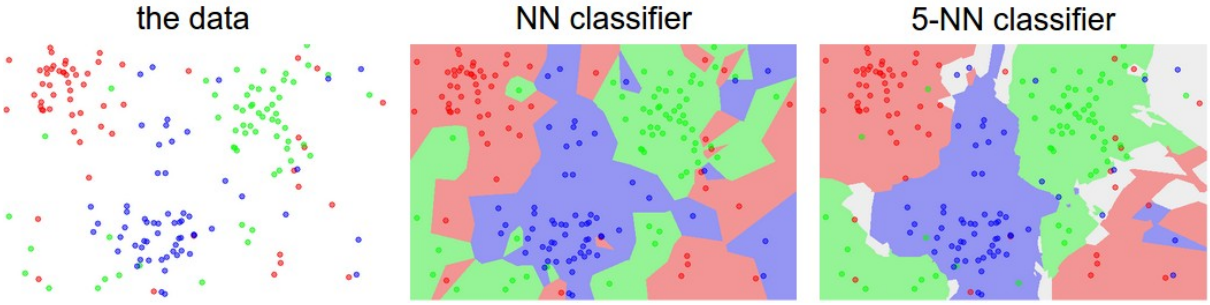

Figura 5 – Diferentes escolhas para K > 0 (CS231N. . . , 2017)

A distância entre os objetos pode ser calculada utilizando a distância euclidiana, mostrada na equação 2.6. Para classificação de documentos, uma das maneiras de calcular essa distância é realizar o cálculo da distância com o vetor resultante do processo de TF-IDF, mostrado na seção 2.4.5.

$$D_{euclidiana}(X, Y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \tag{2.6}$$

A métrica mais comum no processamento de linguagem natural para calcular a semelhança entre vetores é o uso do cosseno.

$$cos_{semelhanca}(v, w) = |\frac{v \cdot w}{|v||w|}| = cos(\theta) \tag{2.7}$$

onde:

$$|v| = \sqrt{\sum_{i=1}^{n} v_i^2} \tag{2.8}$$

Há ainda outros meios de calcular a distância entre vetores, e similaridade entre dados, como apresentados por (LESKOVEC; RAJARAMAN; ULLMAN, 2010).

2.5.2 Algoritmos de Clusterização

Dado um conjunto de pontos em um espaço multidimensional, o objetivo da clusterização é calcular a divisão da seções, *cluster*, para esses pontos, de forma que os pontos no mesmo *cluster* sejam mais similares que os pontos de outros *clusters* (AGARWAL; MUSTA, 2004). A seguir, são apresentados alguns dos algoritmos de clusterização.

2.5.2.1 Algoritmo *K-means*

Segundo (ARTHUR; VASSILVITSKII, 2007), *K-means* é um dos algoritmos de clusterização mais antigos, e importantes. É definido de acordo com o trabalho de (LLOYD, 1957).

K-means define K centroides que são utilizados para definir os *clusters*, onde K é o número de *clusters* desejados. Um ponto P é considerado parte do $Cluster_1$, quando a distância entre a centroide do $Cluster_1$, e P é menor que a distância entre P e os demais *clusters* $Cluster_2, ..., Cluster_n$ (PIECH, 2013). O algoritmo *K-means* é mostrado na Figura 6, e é descrito a seguir:

1. Inicialize as centroides dos *clusters* $\mu_1, \mu_2, ..., \mu_n$ aleatoriamente no plano R;
2. Calcule a distância D entre os pontos P, no plano R, com as centroides definidas anteriormente;
3. Para a menor distância D entre P_i e a centroide μ_k, atribua P_i ao *cluster* referente a μ_k;
4. Defina um novo valor para as centroides sendo a média entre as distâncias da centroide μ_k com os pontos P_k, e
5. Enquanto não convergir, volte ao passo 2.

2.5.3 Métricas de Avaliação do Aprendizado de Máquina

As duas métricas mais bem aceitas em sistemas de recuperação de informação são: *Precision* e *Recall* (SINGHAL, 2001). De acordo com (MANNING; RAGHAVAN; SCHüTZE, 2008), e dada a Tabela 1, *precision* e *recall* podem ser definidas da seguinte maneira:

***Precision* (*P*):** é a fração dos documentos que são relevantes.

$$Precision = \frac{Tp}{(Tp + Fp)} \tag{2.9}$$

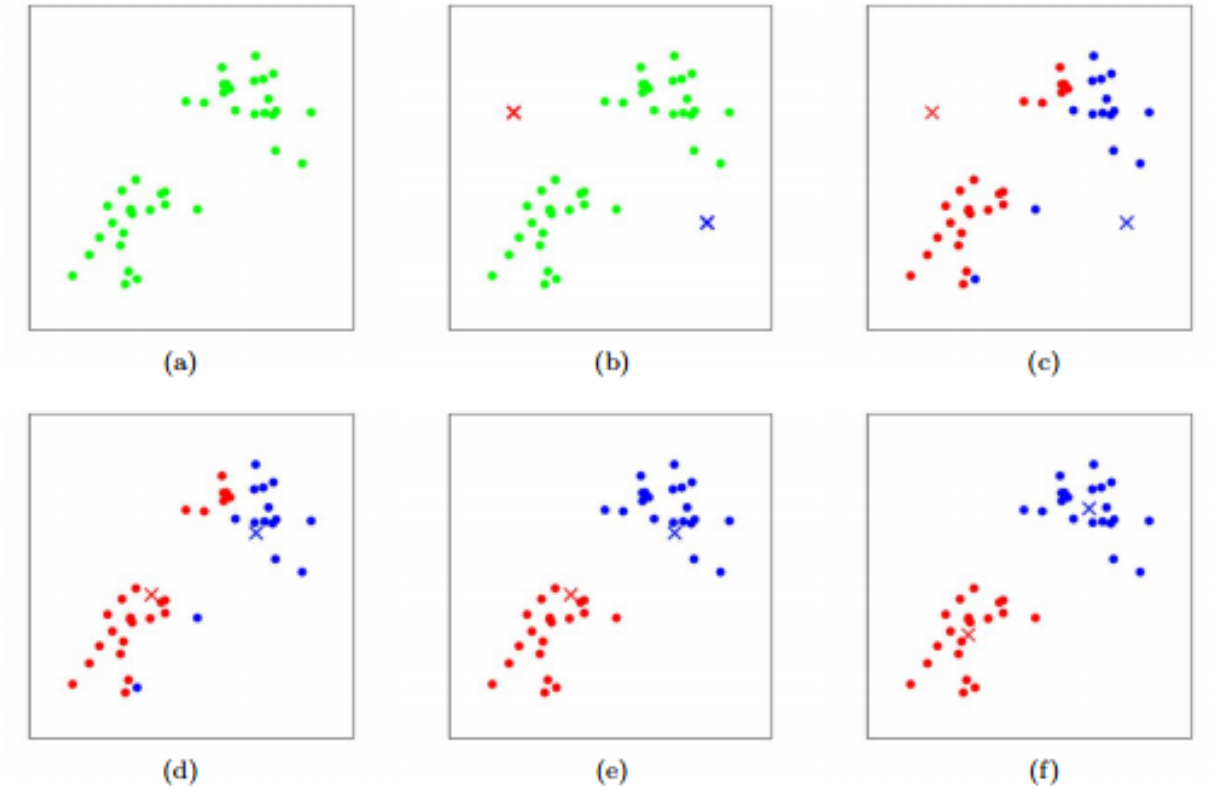

Figura 6 – Algoritmo *K-means*. Os pontos representam exemplos de treino e as cruzes representam as centroides dos *clusters*. (a) Conjunto de dados de teste. (b) Inicialização randômica das centroides. (c-f) Ilustração das interações do algoritmo. Em cada interação, os pontos são atribuídos à centroide mais próxima; então a centroide é movida para o local da média dos pontos. O processo se repete até as centroides convergirem. (NG, 2017)

***Recall* (*R*):** é a fração dos documentos relevantes que foram retornados.

$$Recall = \frac{Tp}{(Tp + Fn)} \tag{2.10}$$

Adicionalmente, para medir o sucesso dos algoritmos de classificação, pode-se utilizar o *F-measure*, descrito a seguir:

***F-measure*:** é a média harmônica ponderada entre *precision* e *recall* (MANNING; RAGHAVAN; SCHüTZE, 2008), e pode ser escrita de acordo com a Equação 2.11.

$$F - measure = \frac{2PR}{P + R} \tag{2.11}$$

Tabela 1 – Tabela de Contingência - (MANNING; RAGHAVAN; SCHüTZE, 2008)

	Relevante	Não-Relevante
Recuperado	Verdadeiro Positivo (Tp)	Falso Positivo (Fp)
Não-Recuperado	False Negativo (Fn)	Verdadeiro Negativo (Tn)

2.6 Resumo do Capítulo

Esse capítulo procurou abordar os principais referenciais teóricos, os quais apóiam e fundamentam o presente trabalho. Destaca-se a atividade de Elicitação de Requisitos, da Engenharia de Requisitos, permitindo estabelecer o principal foco de contribuição desse trabalho, sendo essa a atividade de atuação. São destacadas técnicas de elicitação tradicionais, como questionários e entrevistas, permitindo acordar ainda uma técnica mais emergente, chamada *CrowdRE*. Essa será a técnica utilizada nessa proposta. Visando lidar com o grande volume de informações intrínseco dessa técnica, foram também cobertos no capítulo aspectos relevantes de: (i) Processamento de Linguagem Natural, para elicitação de requisitos extraídos da multidão, uma vez que se torna impraticável a análise manual de milhares de documentos e dados, (ii) Algoritmos de Aprendizagem de Máquina, com a apresentação de dois classificadores - *Naive Bayes* e KNN, e um Algoritmo de Clusterização - *K-means* e (iii) métricas de avaliação dos classificadores, como *precision*, *recall* e *F-measure* que apoiam a tarefa de avaliar os algoritmos de ML.

3 Referencial Tecnológico

Nesse capítulo, tem-se a apresentação do suporte tecnológico, o qual tem sido utilizado para o desenvolvimento desse trabalho. O capítulo está dividido em Seções. A Seção 3.1 é dedicada à apresentação dos suportes utilizados no desenvolvimento do classificador proposto nesse trabalho. Na Seção 3.2, são apresentados os suportes utilizados para o gerenciamento da construção do software, objeto de estudo desse trabalho. Na Seção 3.3.1, têm-se os suportes dedicados à elaboração da monografia. Por fim, são apresentadas as considerações finais do capítulo.

3.1 Desenvolvimento do Classificador

Esta Seção é dedicada a expor as tecnologias que foram utilizadas para o desenvolvimento da aplicação. A Seção foi organizada em subseções que mostram todas as tecnologias utilizadas.

3.1.1 Python

Python é uma linguagem de programação de alto-nível, interpretada e orientada a objetos. Python possui estrutura de dados de alto-nível, com tipagem dinâmica, permitindo o desenvolvimento rápido de aplicações (PYTHON, 2018). A versão utilizada para o desenvolvimento da aplicação foi a versão 3.

3.1.2 *VirtualEnv*

VirtualEnv é uma ferramenta para criar ambientes virtuais para programação com a linguagem Python. Essa ferramenta permite que os programas necessários para o desenvolvimento sejam isolados do resto do sistema operacional. Permitindo, assim, que versões específicas possam ser utilizadas sem interferir em outros projetos, ou funcionalidades do sistema operacional (VIRTUALENV, 2018). A versão 4.8.3 foi utilizada para o desenvolvimento desse trabalho.

3.1.3 *VirtualEnvWrapper*

VirtualEnvWrapper é uma extensão do software *virtualenv*. Essa extensão permite que sejam criados, e excluídos, com facilidade, os ambientes virtuais. Tornando fácil a

configuração para uso em projetos (VIRTUALENVWRAPPER, 2018). A versão 4.8.3 foi utilizada pra o desenvolvimento desse trabalho.

3.1.4 Scikit-Learn

Scikit-Learn é um módulo Python que integra diversos algoritmos de aprendizado de máquina. Esse módulo foca em viabilizar o aprendizado de máquina para não-especialistas. Scikit-learn disponibiliza uma API de alto-nível, com poucas dependências, e foco na facilidade de aprendizado (BUITINCK et al., 2013). A versão 0.22 foi utilizada para a realização desse trabalho.

3.1.5 Pandas

Pandas é um módulo Python, *open-source*, que entrega alto desempenho e facilidade de uso para analisar dados (PANDAS, 2018). A versão 0.23 foi utilizada para o desenvolvimento desse trabalho.

3.1.6 Numpy

Numpy é um pacote para computação científica em Python. É um módulo que possui integração com a linguagem C para produzir resultados rápidos (NUMPY, 2018). Numpy é utilizado como base em diversos outros módulos, como o pandas e o scikit-learn. A versão utilizada para a realização desse trabalho foi a versão 0.19.1.

3.1.7 *Natural Language Toolkit*

Natural Language Toolkit, ou NLTK, é uma plataforma para construir programas para trabalhar com linguagem humana. Provê uma interface fácil de utilizar e possui diversas ferramentas para processar texto, como por exemplo: ferramenta de tokenização de texto e de classificação (NATURAL..., 2018). A versão utilizada nesse trabalho foi a 3.2.5.

3.1.8 Python-eve

Eve é um *framework* para construir *APIs* REST com facilidade (PYTHON-EVE, 2018). Eve utiliza outro framework como base, o *flask* (FLASK, 2018). Esse *framework* adota o minimalismo para entregar uma *API* funcional rapidamente, agilizando o desenvolvimento. A versão utilizada para esse trabalho foi a versão 0.8.

3.1.9 MongoDB

MongoDB é um banco de dados *no-sql*, *open-source*, baseado em documentos. MongoDB é flexivel e permite o armazenamento de documentos no estilo JSON (MONGODB, 2018). Esse banco foi escolhido, pois os dados analisados vieram em formato JSON e o *framework* Python-eve utiliza uma conexão com o MongoDB por padrão. Adicionalmente, existe a possibilidade de utilização de um servidor gratuito oferecido pelo próprio MongoDB, o MongoDB Atlas **MongoAtlas**, o qual permite a utilização de até 512 MB de espaço em nuvem. A versão do MongoDB utilizada para esse trabalho foi a versão 3.4.

3.2 Gerenciamento

Esta Seção é dedicada a expor as metodologias e ferramentas que foram utilizadas para o gerenciamento desse trabalho. A Seção foi organizada em subseções, apresentadas a seguir.

3.2.1 Kanban

O Kanban é uma estrutura usada para auxiliar o desenvolvimento ágil. Os itens, ou tarefas, são representadas visualmente em um quadro, permitindo a visibilidade do estado de cada tarefa. O quadro do Kanban, para esse trabalho, foi dividido em três partes: *To-Do*, *Doing* e *Done*. Esses quadros representam os estados, a fazer, fazendo e Feito, respectivamente.

3.2.2 Git

Git é um sistema de controle de versão gratuito e *open-source*. Permite o desenvolvimento de aplicações simples até as mais complexas. A versão utilizada foi a 2.17.

3.3 Elaboração da Monografia

Esta Seção é dedicada a expor as ferramentas que foram utilizadas para a escrita desse trabalho. A Seção foi organizada em subseções, apresentadas a seguir.

3.3.1 *OverLeaf*

OverLeaf é uma plataforma colaborativa *online* para escrever textos utilizando o LaTeX. (OVERLEAF, 2018). A versão v2 foi utilizada para o desenvolvimento desse trabalho.

3.4 Resumo do Capítulo

A tabela a seguir resume as tecnologias mencionadas anteriormente, descrevendo-as, brevemente, e citando suas versões.

Tabela 2 – Tecnologias utilizadas

Nome	Descrição	Versão
Python	Linguagem de Programação de alto nível	3
VirtualEnv	Ferramenta para criar ambientes virtuais	4.8.3
VirtualEnvWrapper	Extensão do *VirtualEnv*	4.8
Scikit-Learn	Framework com diversos algorítimos de aprendizado de máquina	0.22
Pandas	Módulo para manipulação e análise de dados	0.22
Numpy	Módulo Python para computação científica	0.19.1
NLTK	Módulo Python para tratar linguagem natural	3.2.5
Python-eve	Framework para construção de APIs RESTful	0.8
MongoDB	Banco de dados NoSQL	3.4
Kanban	Metodologia para organizar tarefas em estados	-
Git	Controle de versão	2.17
OverLeaf	Editor de textos LaTeX online	v2

4 Metodologia

Este capítulo aborda a metodologia adotada durante o desenvolvimento dessa monografia. O capítulo é dividido em seções, que são apresentadas a seguir. Na Seção 4.1, são explicadas como são a abordagem, a natureza, os objetivos e os procedimentos adotados para a realização desse trabalho. Na Seção 4.2, é apresentado o fluxo das atividades que foi adotado tanto para a realização desse trabalho, quanto para o desenvolvimento da aplicação, apresentada no Capítulo 5. Na Seção 4.3, é apresentado o processo de coleta e análise dos resultados iniciais. Na Seção 4.4, é apresentado o cronograma definido para esse trabalho. Por fim, são apresentadas as considerações finais do capítulo.

4.1 Classificação da Pesquisa

Segundo (GERHARDT; SILVEIRA, 2009), a pesquisa científica é o resultado de um inquérito ou exame minucioso com o objetivo de resolver um problema. Assim, pode-se identificar uma pesquisa quanto à sua abordagem, sua natureza, seus objetivos e seus procedimentos.

4.1.1 Abordagem

A pesquisa qualitativa têm como objetivo o aprofundamento da compreensão do assunto estudado. Pesquisadores que utilizam o método qualitativo não quantificam valores, pois os dados analisados são não-numéricos. Em contrapartida, a pesquisa quantitativa centra-se na objetividade. Os dados são coletados, preferencialmente, a partir de ferramentas e instrumentos padronizados. Há o uso da matemática para descrever o comportamento observado (GERHARDT; SILVEIRA, 2009). A pesquisa quantitativa foi adotada para a realização dessa monografia, uma vez que: há coleta controlada de dados; é enfatizada a objetividade na coleta, e análise dos dados, e por fim os dados numéricos produzidos são analisados através de procedimentos estatísticos.

4.1.2 Natureza

De acordo com (GERHARDT; SILVEIRA, 2009), existem dois tipos de pesquisa:

Pesquisa Básica tem como objetivo gerar conhecimentos novos, úteis para o avanço da ciência, sem aplicação prática prevista, e

Pesquisa Aplicada tem como objetivo gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos.

Esse trabalho tem como objetivo auxiliar gerentes de software na priorização de funcionalidades através do *feedback* dos usuários. Caracteriza-se uma pesquisa aplicada, pois há aplicação prática do estudo, e envolve interesses de um grupo específico.

4.1.3 Objetivos

(GIL, 2002) classifica pesquisas em três grupos, brevemente apresentados a seguir:

Pesquisa Exploratória têm como objetivo proporcionar maior familiaridade com o problema para torná-lo mais explícito. Essa pesquisa, geralmente, envolve as seguintes atividades: levantamento bibliográfico; entrevistas com pessoas que tiveram experiências com o problema pesquisado e análise de exemplos;

Pesquisa Descritiva procura descrever as características e os fenômenos de determinada realidade. Em uma pesquisa sobre a preferência de uma marca, é interessante saber características sobre o consumidor, tais como: idade, sexo, em qual cidade mora, e

Pesquisa Explicativa procura identificar fatores que determinam ou contribuem para a ocorrência dos fenômenos.

Esse trabalho é caracterizado como uma pesquisa exploratória e explicativa, pois há a realização de atividades como o levantamento bibliográfico e a identificação dos fatores que contribuem para a realização das atividades que serão apresentados no Capítulo 5.

4.1.4 Procedimentos

De acordo com (FONSECA, 2002), a pesquisa possibilita uma aproximação e um entendimento da realidade a ser investigada. A modalidade de pesquisa adotada para a realização dessa monografia é:

Pesquisa Bibliográfica é feita a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos. Qualquer trabalho científico inicia-se com essa pesquisa (FONSECA, 2002), e

Pesquisa-Ação pressupõe uma participação planejada do pesquisador na situação problemática a ser investigada. O processo de pesquisa recorre a uma metodologia sistemática, no sentido de transformar as realidades observadas, a partir da sua compreensão, conhecimento e compromisso para a ação dos elementos envolvidos na pesquisa.

4.2 Fluxo das Atividades

Essa Seção apresenta uma breve descrição das atividades que são adotadas para auxiliar o desenvolvimento desse trabalho. O processo como um todo pode ser visto na Figura 7.

Realizar Levantamento Bibliográfico (Realizada): envolve a pesquisa necessária para auxiliar o desenvolvimento desse trabalho. Ao realizar essa atividade, é possível compreender melhor o domínio das questões levantadas na Seção 1.2. Assim como é possível definir e refinar o escopo;

Definir Escopo (Realizada): baseado no contexto estudado durante a atividade de Levantamento Bibliográfico, foi possível definir os limites de ação desse trabalho, seus objetivos gerais, seus objetivos específicos e o escopo da aplicação, apresentada no Capítulo 5;

Desenvolver Referencial Teórico (Realizada): tem como objetivo apresentar os principais conceitos e fontes para apoiar esse trabalho;

Descrever Metodologia (Realizada): envolve determinar a metodologia para guiar a pesquisa sobre o assunto abordado e definir o fluxo de atividades que guiará o desenvolvimento deste trabalho e do software proposto;

Descrever Referencial Tecnológico (Realizada): apresenta, e descreve, as tecnologias definidas para o desenvolvimento do software apresentado no Capítulo 5;

Desenvolver Prova de Conceito (Realizada): essa macroatividade, descrita na Figura 8, envolve as seguintes atividades: mapeamento inicial de requisitos de software; realização da *sprint* para implementação dos requisitos priorizados, e a classificação manual de exemplos para a base de treino. Tem como objetivo provar a viabilidade do software desenvolvido e apresentado no Capítulo 5. A descrição das atividades pode ser vista na Seção 4.2.1;

Definir Escopo da Aplicação (Realizada): tem como objetivo documentar a aplicação capaz de solucionar as questões elicitadas no Objetivo Geral e nos Objetivos Específicos, apresentados anteriormente na Seção 1.4;

Refinar Monografia (Realizada): é realizada em paralelo às atividades descritas anteriormente. Tem como objetivo realizar correções na parte escrita do trabalho, ao longo do tempo;

Realizar Correções Sugeridas (Realizada): tem como objetivo realizar correções sugeridas ao término do Trabalho de Conclusão de Curso 1.

Desenvolver o Classificador (Realizada): foi realizado o desenvolvimento da aplicação, apresentada no Capítulo 5. Seu desenvolvimento segue o padrão adotado para

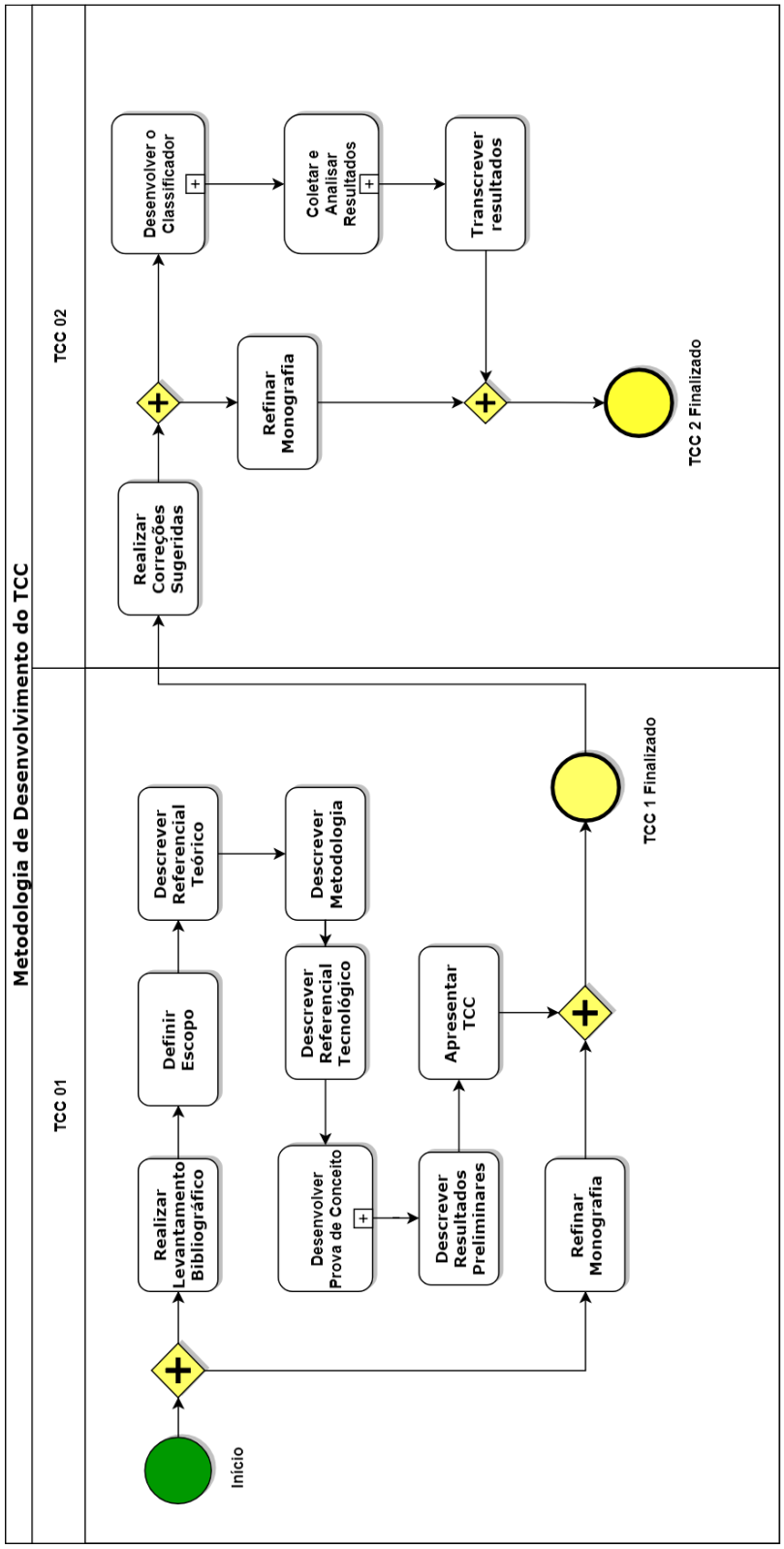

Figura 7 – Fluxo da Metodologia

a Prova de Conceito.

Coletar e Analisar Resultados (Realizada): durante essa macroatividade, foram realizadas a coleta e análise dos dados obtidos através do uso do Classificador proposto. O processo de coleta dos dados que foram utilizados de insumo para o Classificador é descritos na Seção 4.3.1.

Transcrever Resultados (Realizada): tem como objetivo documentar os resultados finais descritos no Capítulo 6.

4.2.1 Atividades do Desenvolvimento das Aplicações de Apoio e do Classificador

Todo o desenvolvimento dos produtos de software desse trabalho são guiados pelas atividades descritas por (SCHWABER; SUTHERLAND, 2017). As atividades envolvidas para alcançar esse objetivo são descritas a seguir:

Identificar Requisitos Iniciais: essa atividade tem como objetivo elicitar os requisitos iniciais e mais abstratos da aplicação;

Priorizar Requisitos: os requisitos identificados na atividade anterior são priorizados e, então, colocados no *backlog*;

Implementar Requisitos Priorizados: essa atividade é a realização de algumas práticas sugeridas da metodologia Scrum (SCHWABER; SUTHERLAND, 2017), entre as quais: definição do *backlog*, realização de *sprints* e entregas parciais;

Classificar Manualmente Exemplos para a Coleção de Treino: Como foi apresentado anteriormente, na Seção 2.5, algoritmos de Aprendizado de Máquina, que utilizam o cenário de aprendizado supervisionado, precisam de uma base de dados de treinamento. Uma vez que os dados coletados para a realização desse trabalho não são previamente classificados, houve a necessidade de classificá-los manualmente para a utilização dos algoritmos de aprendizado supervisionado.

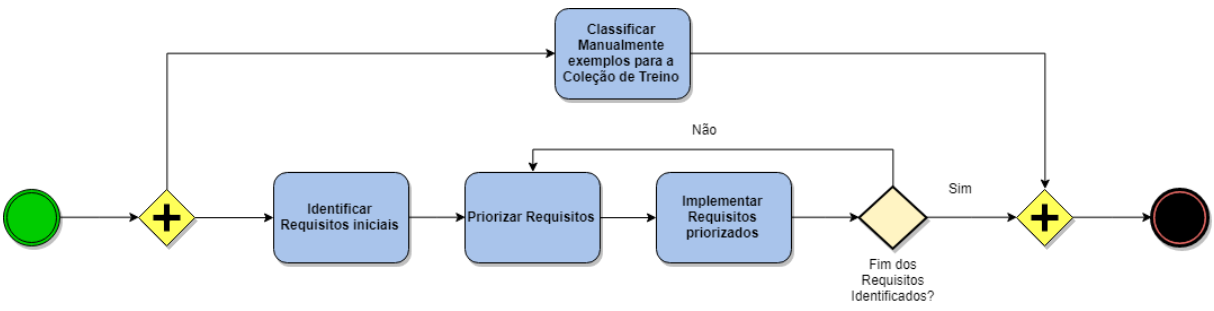

Figura 8 – Atividades do Desenvolvimento da Prova de Conceito

4.3 Coleta de Dados e Análise de Resultados

A coleta dos dados a serem estudados e sua análise são partes fundamentais de uma pesquisa. Fazem parte da constatação e devem ser iniciadas após o problema ter sido pesquisado. Os resultados da pesquisa não podem ser definidos sem que se tenha feito a constatação (GERHARDT; SILVEIRA, 2009).

4.3.1 Coleta de Dados

Segundo (GERHARDT; SILVEIRA, 2009), a coleta de dados deve levar em consideração três questões:

O que coletar? corresponde aos dados que serão coletados para testar as hipóteses. Para esse trabalho, serão coletados comentários de usuários em lojas de aplicativos de *smartphone*;

Com quem coletar? trata-se de recortar o campo das análises empíricas. Isto é, limitar o estudo à população total ou à uma amostra representativa, ou ilustrativa. Esse trabalho adota o uso de uma amostra representativa dos comentários de usuário em lojas de aplicativos, disponibilizada por (MCAULEY et al., 2015);

Como coletar? refere-se aos instrumentos de coleta dos dados, os quais devem ser testados antes da utilização real. Os dados coletados são provenientes de um conjunto de dados disponibilizados por (MCAULEY et al., 2015).

4.3.2 Análise de Resultados

O objetivo da pesquisa é responder à questão inicial. Segundo (GERHARDT; SILVEIRA, 2009), uma vez que os dados foram coletados, deve-se verificar se as informações correspondem às hipoteses. Adicionalmente, as informações podem fornecer interpretações sobre fatos não cogitados e refinar as hipóteses.

Neste trabalho, a análise dos dados foi feita com base nas métricas apresentadas na Seção 2.5.3, em conjunto com a observação dos dados após os procedimentos que serão apresentados no Capítulo 5. Uma visão geral das atividades pode ser vista na Figura 9, e seus detalhes são descritos a seguir:

Identificar Fatores de Influência: essa atividade foi realizada para identificar: (i) quais são os elementos que contribuíram para a resolução dos objetivos citados anteriormente; (ii) quais informações, nos comentários de usuário, são relevantes, e (iii) quais poderiam ser utilizados para alimentar a pesquisa.

Coletar Dados do Classificador: após identificar os fatores de influência, e utilizá-los como dados para o software proposto, foi realizada a coleta dos dados retornados pelo software.

Calcular Métricas: foi realizado o cálculo de *Precision* e *Recall*, apresentadas na Seção 2.5.3. Caso não fosse atingido um resultado satisfatório, uma nova iteração para identificar novos fatores de influência foi realizada, e

Documentar Dados: após atingir resultados satisfatórios nas etapas anteriores, os dados finais foram documentados e entendidos como resultado obtido pela pesquisa.

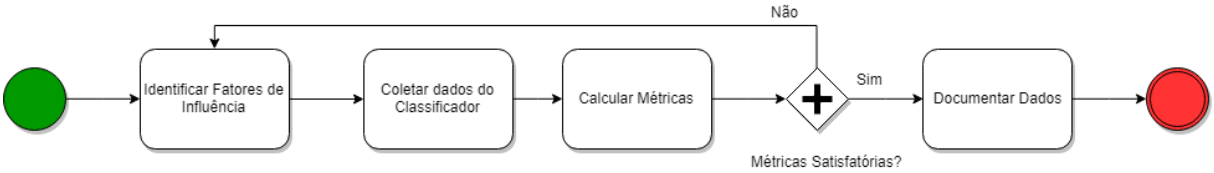

Figura 9 – Atividades da Análise de Resultados

4.4 Cronograma

As Tabelas 3 e 4, apresentadas a seguir, acordam os cronogramas que descrevem os prazos das atividades descritas anteriormente.

4.5 Resumo Do Capítulo

Esse capítulo procurou abordar a metodologia adotada para o desenvolvimento do presente trabalho. Foi apresentado o referencial teórico que classifica a pesquisa quanto a: (i) sua abordagem, sendo a pesquisa quantitativa; (ii) sua natureza, sendo a pesquisa aplicada; (iii) seus objetivos, sendo a pesquisa explicativa, com base nos fenômenos identificados; (iv) e seus procedimentos, sendo essa pesquisa caracterizada por ser uma pesquisa bibliográfica. Adicionalmente, foram apresentadas as atividades que compõem o fluxo de desenvolvimento do trabalho e seus respectivos prazos. Por fim foram descritas as metodologias para desenvolvimento da prova de conceito bem como do software proposto, e para análise de resultados.

Atividades	**Janeiro**	**Fevereiro**	**Março**	**Abril**	**Maio**	**Junho** **Julho**
Definir Tema	**X**					
Realizar Levantamento Bibliográfico	**X**	**X**				
Definir Escopo		**X**				
Descrever Referencial Teórico			**X**			
Descrever Metodologia				**X**		
Descrever Referencial Tecnológico				**X**		
Desenvolver Prova de Conceito				**X**		
Descrever Resultados Preliminares					**X**	
Apresentar Monografia						**X**
Refinar Monografia	**X**	**X**	**X**	**X**	**X**	**X**

Tabela 3 – Cronograma de Atividades do Trabalho de Conclusão de Curso 1

Atividades	**Julho**	**Agosto**	**Setembro**	**Outubro**	**Novembro**	**Dezembro**
Realizar Correções Sugeridas	**X**					
Desenvolver o Classificador	**X**	**X**	**X**			
Coletar e Analisar Resultados		**X**	**X**			
Transcrever Resultados				**X**		
Refinar Monografia	**X**	**X**	**X**	**X**	**X**	
Apresentar TCC 2						**X**

Tabela 4 – Cronograma de Atividades do Trabalho de Conclusão de Curso 2

5 Aplicação

Nesse capítulo, é abordada a aplicação que foi desenvolvida ao longo desse Trabalho de Conclusão de Curso. O capítulo encontra-se dividido em seções, que são apresentadas a seguir. Na Seção 5.1.1, é apresentada a aplicação fazendo o uso de uma abordagem caixa-preta, sendo essa especificada como um fluxograma na notação SADT (LAKHOUA; ANNABI, 2006). Na Seção 5.1.2, são apresentadas as atividades de classificação e clusterização, as quais acordam uma visão mais detalhada da aplicação. Por fim, é apresentado o resumo do capítulo.

5.1 A aplicação

A aplicação tem a intenção de auxiliar gerentes de software a priorizar as solicitações de usuários feitas em lojas de aplicativos. Para isso, foram classificados automaticamente os comentários de usuários em requisitos de software e, então, agrupados de acordo com o conteúdo. Os requisitos identificados são, então, classificados em: requisitos funcionais (i.e funcionalidades) ou requisitos não funcionais (i.e critérios de qualidade). Nas próximas seções desse capítulo, serão apresentadas as atividades envolvidas nas etapas de classificação bem como de clusterização.

5.1.1 Nível 1

Com base no modelo SADT (LAKHOUA; ANNABI, 2006), a Figura 10 ilustra a aplicação orientada a uma abordagem do tipo caixa preta, na qual são relevantes apenas as entradas, os controles, os mecanismos e as saídas. Nesse caso, sendo omitidos os detalhes mais internos da aplicação. Como entrada, seta à esquerda, têm-se os *Feedbacks* dos usuários nas lojas de aplicativos. Os *Feedbacks* foram obtidos através de (MCAULEY et al., 2015). Como controles, seta acima, serão utilizados conceitos, princípios e demais modelos conceituais acordados nas referências: (GROEN et al., 2017), (GROEN; DOERR; ADAM, 2015), (LEVY; HADAR; TE'ENI, 2015), (BRABHAM, 2008), (STOREY et al., 2014). Destacam-se, nesse contexto, conforme proposto pelos autores em:

- (BRABHAM, 2008) mostra que a multidão consegue resolver problemas, os quais alguns times especializados não conseguem. Sob certas circunstâncias, a multidão é mais inteligente que a pessoa mais inteligente dentro da multidão.

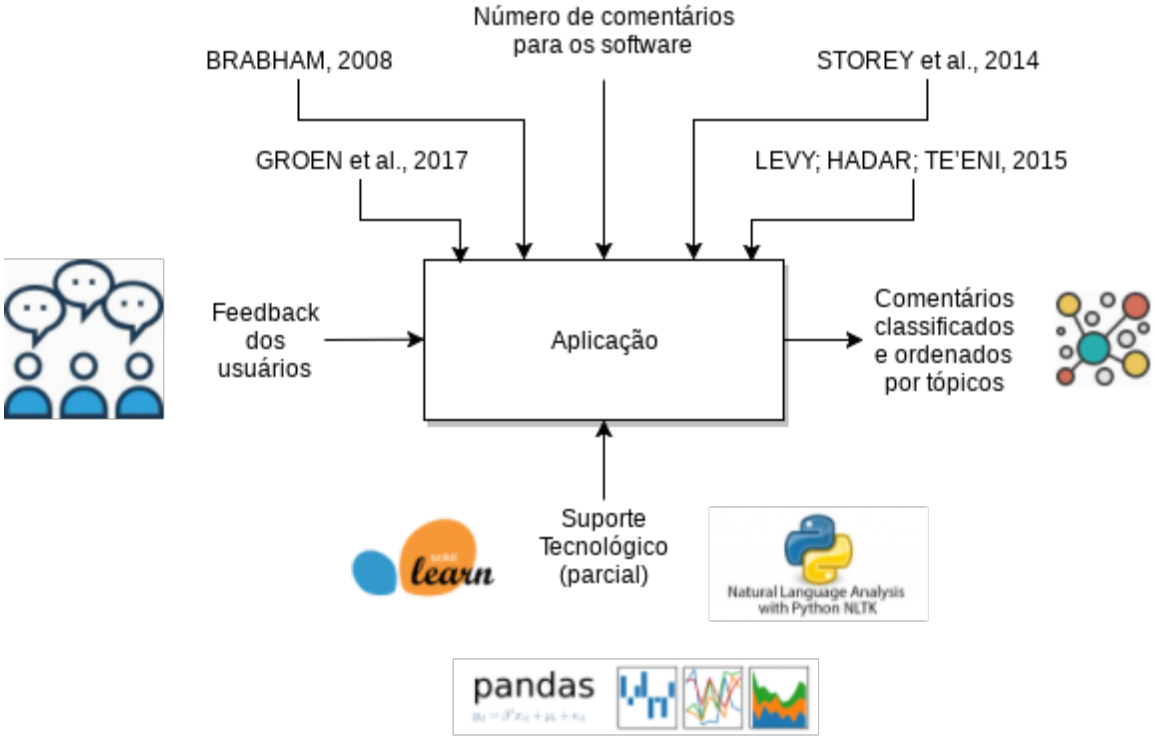

Figura 10 – Nível 1 - Visão Geral da Aplicação. O diagrama possui a seguinte estrutura: (i) a seta a esquerda representa os dados de entrada necessários à aplicação, (ii) a seta acima, as atividades de controle, (iii) a seta abaixo, os mecanismos que darão suporte às atividades, e (iv) a seta a direita, o resultado final do processo (LAKHOUA; ANNABI, 2006).

- (GROEN et al., 2017) apresenta a dificuldade de engajar um número considerável de *stakeholders*, e mostra a possibilidade de mobilizar os usuários do software através de canais de comunicação específicos, como lojas de aplicativos.
- (LEVY; HADAR; TE'ENI, 2015) cita quais são os fatores de motivação para um indivíduo participar no processo de *crowdsourcing*. Os fatores incluem: aprender algo novo, ganhar alguma compensação financeira ou algum tipo de benefício.
- (STOREY et al., 2014) apresenta o impacto das redes sociais no desenvolvimento de software e mostra como desenvolvedores utilizam esse meio de comunicação entre os usuários.

Um fator importante para a realização de uma atividade envolvendo multidões, é o número de indivíduos na multidão. A literatura não especifica um número exato de participantes que uma multidão deve ter. Entretanto, deve-se atentar que um número pequeno de participantes contribui pouco para identificar as necessidades do software. (GROEN et al., 2017) cita que *CrowdRE* têm o objetivo de atingir o maior número de indivíduos possível.

Como mecanismos, seta abaixo, foram utilizados os Referenciais Tecnológicos apresentados nas seções 3.1.4, 3.1.5, 3.1.7. Por fim, como saídas, seta à direita, têm-se os comentários classificados pelas categorias de requisitos de software bem como ordenadas por assunto para cada categoria. Em termos de categorias, podem ser mencionadas as categorias apresentadas na Seção 5.1.3.1.

5.1.2 Nível 2

Ao expandir o módulo do Nível 1, e de acordo com a Figura 11, é possível ver os processos que compõem o Nível 2, também representados pelo modelo SADT (LAKHOUA; ANNABI, 2006). Esses processos são a Classificação e a Clusterização, respectivamente.

Para o processo de Classificação, têm-se: (i) como entrada, os comentários realizados pelos usuários em lojas de aplicativos; (ii) como mecanismos, as tecnologias apresentadas na Seção 3.1 e, (iii) como saída, os comentários classificados em requisitos de software, de acordo com a Seção 5.1.3.1.

Para o processo de Clusterização, têm-se: (i) como entrada, o resultado do processo anterior, os comentários classificados em requisitos de software; (ii) como mecanismos, as tecnologias apresentadas, anteriormente, na Seção 3.1, e como saída (iii) os comentários agrupados por assunto.

Para ambos os processos, foram preservados os controles definidos para o Nível 1, na Seção 5.1.1

Os detalhes desses processos são descritos nas próximas seções.

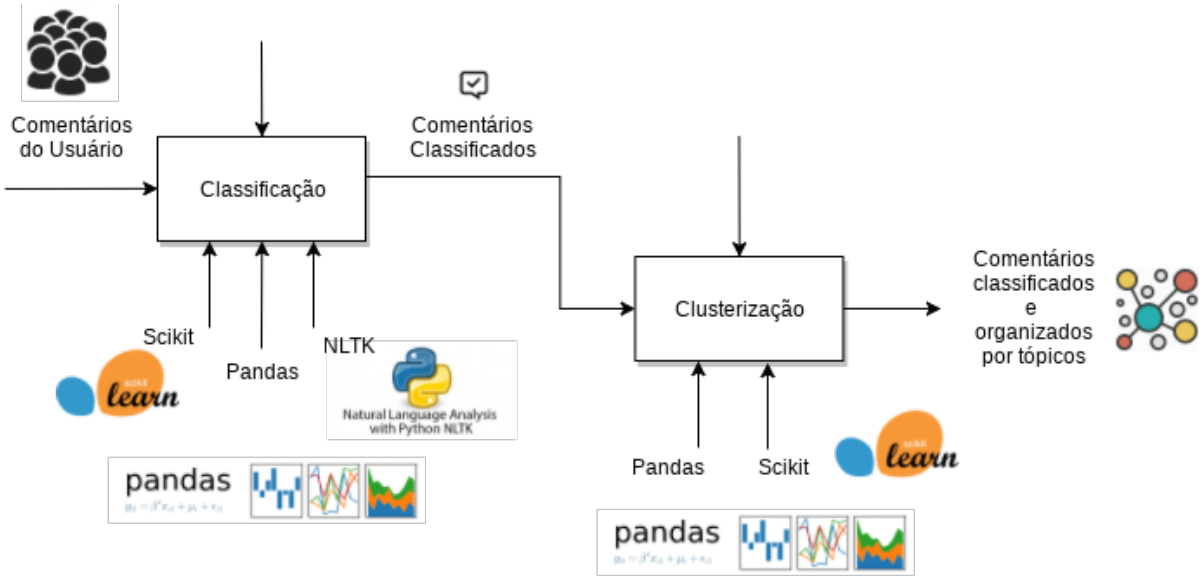

Figura 11 – Nível 2 - Visão Interna da Aplicação (LAKHOUA; ANNABI, 2006)

5.1.3 Classificação

A Figura 12 mostra o processo de uma classificação supervisionada, conceito apresentado no Referencial Teórico, na Seção 2.5.0.1. Esse processo possui duas fases: (i) a fase de treinamento, na qual um comentário passa pelo pré-processamento, discutido na Seção 5.1.3.2, a fim de extrair informações relevantes para treinar o algoritmo na identificação da categoria desse comentário, e (ii) a fase de predição, na qual um comentário, cuja categoria ainda é desconhecida, passa pelo mesmo pré-processamento para que suas informações forneçam insumos para o Classificador. Esse último é o responsável por predizer a categoria a qual o comentário pertence.

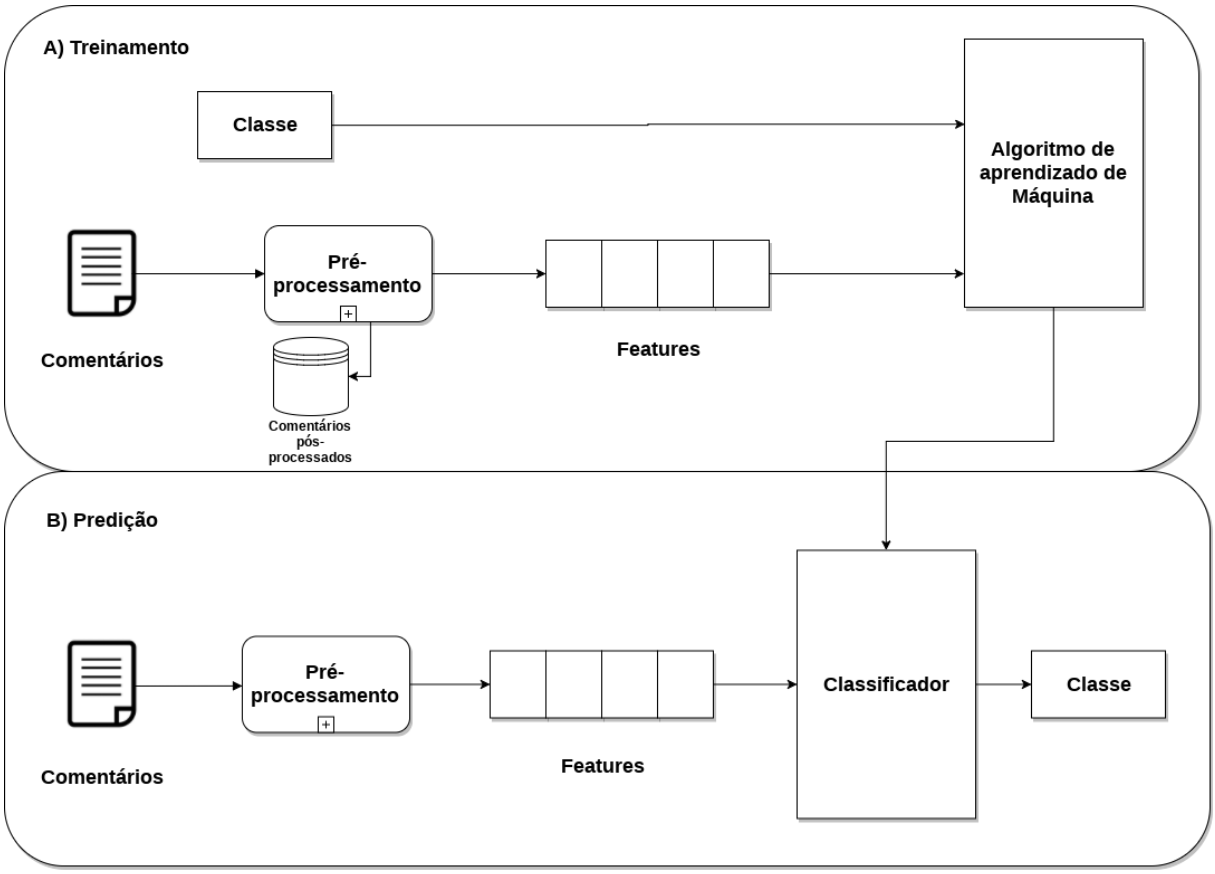

Figura 12 – Arquitetura do Classificador - (BIRD; KLEIN; LOPER, 2014) (Adaptado)

Nas seções 5.1.3.1 a 5.1.3.4, serão tratados detalhes quanto aos comentários; as atividades que compõem o pré-processamento, e os algoritmos de aprendizado de máquina que foram utilizados nesse trabalho.

5.1.3.1 Comentários

Os comentários foram obtidos através de uma coleção de *reviews* proposta em (MCAULEY et al., 2015). Essa coleção possui cerca de 753.000 *reviews* de usuários, na

língua inglesa. Os comentários estão, em sua maioria, dispostos como apresentados na Figura 13. Breve explicações sobre cada atributos são conferidas a seguir.

reviewerID: identificador do usuário,

asin: identificador do produto que recebeu o comentário,

reviewerName: nome do usuário que realizou o comentário,

helpful: utilidade do comentário para a comunidade. O quão útil esse comentário foi para outras pessoas,

reviewText: o texto do comentário em si. A opinião do usuário.

overall: avaliação numérica do usuário sobre o produto.

summary: título do comentário. Uma breve descrição introdutória.

unixReviewTime: data de realização do comentário em *Unix Timestamp*.

reviewTime: data de realização do comentário no formato *mês dia e ano*.


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano. He is having
      a wonderful time playing these old hymns. The music is at times hard to
     read because we think the book was published for singing from more than
     playing from. Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


Figura 13 – Estrutura de um *Review*

A solução apresentada nesse trabalho utiliza os atributos *reviewText* e *summary* para realizar a Classificação e a Clusterização, pois são nesses atributos que a informação sobre a opinião do usuário é expressa no comentário. Adicionalmente, esses atributos são utilizados para a realização das atividades de pré-processamento, apresentadas na Seção 5.1.3.2

As categorias escolhidas para a classificação desses comentários são:

- **Requisitos Funcionais**: comentários que se encontram na categoria Requisitos Funcionais são aqueles que correspondem à definição descrita na Seção 2.1;

- **Requisitos Não funcionais**: comentários que se encontram na categoria Requisitos Não Funcionais encaixam-se na descrição da Seção 2.1.2 ou dentro de uma das categorias apresentadas na Figura 2, e

- **Outros**: Comentários que não se encaixam em algum tipo de requisição, como por exemplo, descrições de como o aplicativo funciona para outros compradores, *spam* e agradecimentos.

5.1.3.2 Pré-processamento

O pré-processamento é o conjunto de atividades que resulta na extração da informação do texto. Para esse trabalho, são aplicados cinco processos, vistos na Figura 14, e descritos a seguir.

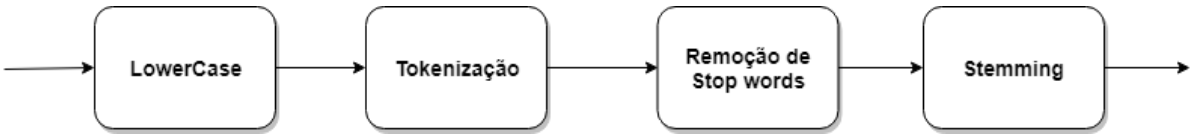

Figura 14 – Fluxo de pré-processamento

LowerCase: consiste em transformar os comentários para caixa baixa, evitando que palavras iguais sejam descartadas ao longo do processo por estarem escritas de maneiras diferentes. Por exemplo: *Aluno* e *aluno* são a mesma palavra, mas seriam consideradas diferentes, caso não houvesse essa etapa de tratamento.

Tokenização: como apresentado na Seção 2.4, tokenização é a subdivisão dos comentários em unidades menores, chamadas *tokens*. Isso facilitará a execução das próximas etapas.

Remoção de Stop Words: na Seção 2.4, também foi apresentado o conceito de *stop words*. Trata-se de uma etapa para remoção de algumas palavras que não são relevantes ao procurar por informações em documentos, ou seja, essas são chamadas de *stop words* (MANNING; RAGHAVAN; SCHüTZE, 2008). No idioma português, têm-se "aquele","ou", "até", e "ao"como exemplos de *stop words*. Sua remoção é fundamental para que as palavras mais relevantes sejam identificadas.

Stemming: durante a fase de pré-processamento, há ainda a realização da etapa de *stemming*. Essa etapa é necessária para reduzir palavras parecidas ao seu *lemma*. Dessa maneira, palavras como *Correr* e *Corremos*, por exemplo, têm a mesma função morfológica.

5.1.3.3 *Features*

Como apresentado anteriormente, nas seções 2.4.4 e 2.4.5, *Bag of Words* e TF-IDF são técnicas para extrair informação do texto considerando a frequência e a importância das palavras nos documentos.

Após a etapa de pré-processamento, onde os documentos são normalizados, foi utilizado o TF-IDF para identificar quais palavras são mais importantes para a classificação de cada documento. Lembrando que essa técnica considera, por exemplo, que a frase *Ana é mais divertida que José* similar, em termos de conteúdo, com *José é mais divertido que Ana*.

Ao final dessa atividade, têm-se uma matriz *NxM*, onde *N* representa os valores atribuídos do TF-IDF às palavras da coleção de documentos, e *M* os documentos. Ao final desse processo, têm-se as *features*, representadas na Figura 12.

5.1.3.4 Algoritmo de Aprendizado de Máquina

Para atingir o objetivo desse trabalho, foi realizada uma classificação supervisionada com o uso do algoritmo de *Naive Bayes*, apresentado, em detalhes, na Seção 2.5 dessa monografia. O algoritmo foi responsável por classificar os comentários dos usuários em requisitos de software. As categorias, as quais esses que esses comentários serão atribuídas, foram apresentadas na Seção 5.1.3.1.

Após a realização do treinamento, o algoritmo está pronto para realizar a predição, como é indicado na Figura 12, que resulta na atribuição de uma categoria ao comentário.

5.1.3.5 Classe

Como observado na Figura 12, durante a fase Treinamento, existe a necessidade de informar ao algoritmo de aprendizado de máquina, nos casos de aprendizado supervisionado, qual classe o comentário pertence. Entretanto, como apresentado na Seção 5.1.3.1, os comentários não possuem classificação prévia.

Para realizar o proposto nesse Trabalho de Conclusão de Curso, existe a necessidade de classificação prévia de uma quantidade de comentários para cada categoria apresentada na Seção 5.1.3.1. Então, foi realizada uma etapa de classificação manual, pelo autor desse trabalho, nesses exemplos.

5.1.4 Clusterização

Após a etapa de Classificação, foi realizada a atividade de Clusterização dos documentos. Essa etapa tem como objetivo identificar tópicos dos comentários que fazem parte da mesma categoria, e agrupá-los para mostrar a relevância das requisições e de onde surgem as necessidades dos usuários. As atividades envolvidas no processo de Clusterização são apresentadas a seguir.

5.1.4.1 *K-Means*

Assim como apresentado anteriormente na Seção 2.5.2.1, o algoritmo de *K-means* realiza uma clusterização ao buscar centroides que são utilizados para definir os *clusters*. Após o pré-processamento dos dados, passos apresentados na Figura 14. Dessa maneira, o dicionário de dados resultante do processo de *TF-IDF* pode ser inserido como parâmetro do algoritmo de *K-means*. Como resultado, têm-se as palavras mais influentes para a aglomeração proposta pelo algoritmo.

5.2 Resumo do Capítulo

Neste capítulo, apresentou-se a aplicação desenvolvida ao longo desse Trabalho de Conclusão de Curso. Procurou-se detalhar em diferentes níveis. Inicialmente, tem-se a aplicação como uma caixa preta; depois é apresentada uma visão mais detalhada, composta de duas etapas: (i) **Classificação**, que comporta duas outras atividades internas, sendo Treinamento e Predição. Para cada atividade, foram apresentadas suas atividades, entre elas: pré-processamento, os comentários que serão classificados, a extração das *features*, e a arquitetura do algoritmo de Classificação; e (ii) **Clusterização**, que mostra como o algoritmo de *k-means* realizará a identificação do assunto tratado nos comentários.

6 Resultados

Neste capítulo, são apresentados os resultados obtidos durante a realização desse trabalho. O capítulo encontra-se dividido em Seções. Na Seção 6.1, são apresentados os produtos de software que deram apoio à classificação manual realizada no trabalho. Na Seção 6.2, são apresentados os resultados do classificador através de iterações do ciclo de pesquisa-ação realizados para esse trabalho. Na Seção 6.3, são apresentados os resultados para o processo de clusterização. E por fim, têm-se as considerações finais do capítulo.

6.1 Classificação Manual de *Reviews*

Com o propósito de criar o classificador, apresentado no Capítulo 5, foi realizado o desenvolvimento de produtos de software intermediários e auxiliares. Uma vez que o classificador usa algoritmos de aprendizado supervisionado, cuja arquitetura é apresentada na Seção 5.1.3, existe a necessidade de uma quantidade de dados previamente classificada para uso no aprendizado durante a fase de treinamento. Dessa forma, optou-se pela construção de um aplicativo de *smartphone* para tal fim.

Para a construção desse aplicativo, foi utilizada a metodologia de desenvolvimento apresentada no Capítulo 4, sendo esta, uma adaptação do *Scrum* (SCHWABER; SUTHERLAND, 2017). O *backlog* desse produto, bem como os critérios de aceitação, podem ser vistos na Tabela 5

A primeira etapa do trabalho consistiu em classificar uma quantidade de *reviews* manualmente. Para essa classificação, foram definidas, conforme a Seção 5.1.3.1, as classes de requisitos não funcionais, funcionais e outros. Os requisitos funcionais e não funcionais foram classificados de acordo com (SOMMERVILLE, 2011). A última versão do aplicativo desenvolvido pode ser vista na Figura 15

História de Usuário	Critério de Aceitação
Eu, como usuário, desejo classificar os comentários de usuário entre as categorias de requisitos possíveis para aumentar o banco de dados de treino do algoritmo de *machine learning*.	As categorias de requisitos são Funcionais, Não funcionais e outros.
	Os requisitos não funcionais devem atender às categorias estabelecidas por (SOMMERVILLE, 2011).
Eu, como usuário, quero visualizar somente os requisitos classificados como Requisito Funcional para saber quantos já foram classificados.	Devem ser mostrados somente os comentários que já foram classificados.
	Não deve ser possível mudar a classificação desses comentários.
Eu, como usuário, quero visualizar somente os requisitos classificados como Requisito Não Funcional para saber quantos já foram classificados.	Devem ser mostrados somente os comentários que já foram classificados.
	Não deve ser possível mudar a classificação desses comentários.
Eu, como usuário, quero que o aplicativo mostre comentários, os quais o classificador julgou ser da categoria Requisito Funcional para que eu possa classificá-los corretamente.	O aplicativo deve retornar somente comentários que o algoritmo de aprendizado de máquina julgou ser da categoria Requisito Funcional.
	Deve ser permitida a edição de categorias desses comentários pelo aplicativo.
Eu, como usuário, quero que o aplicativo mostre comentários, os quais o classificador julgou ser da categoria Requisito Não Funcional para que eu possa classificá-los corretamente.	O aplicativo deve retornar somente comentários que o algoritmo de aprendizado de máquina julgou ser da categoria Requisito Não Funcional.
	Deve ser permitida a edição de categorias desses comentários pelo aplicativo.
Eu, como usuário, quero que o classificador realize, automaticamente, a identificação de um comentário em requisito de software.	As categorias de Requisitos que o classificador pode predizer são: Funcional, Requisito Não funcional e outros, como definidos na Seção 5.1.3.1.

Tabela 5 – Histórias de Usuário e seus critérios de aceitação

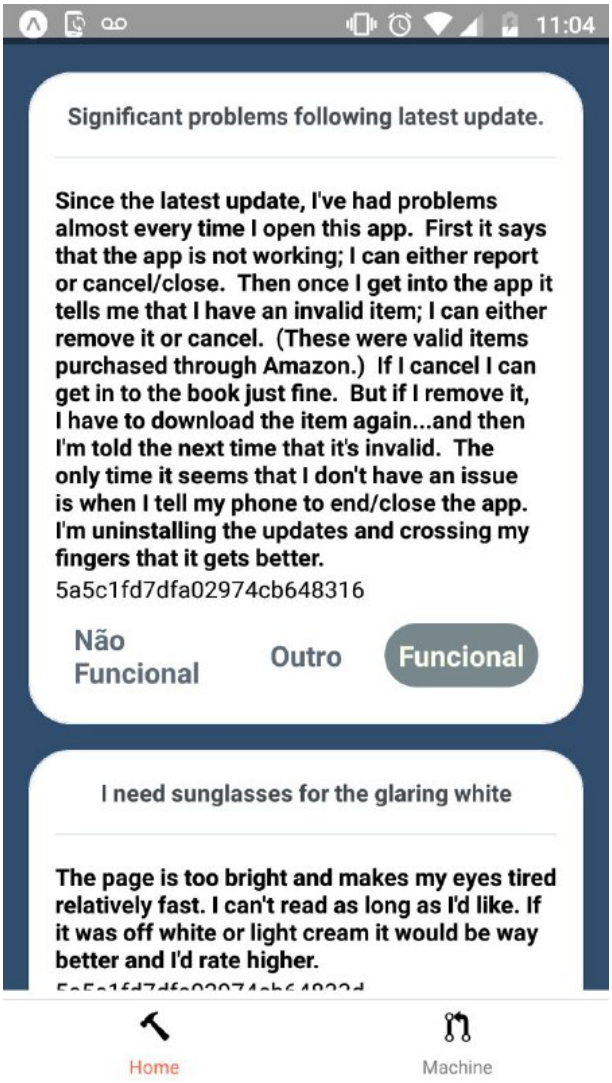

Figura 15 – Aplicativo desenvolvido para classificar manualmente os *reviews* dos usuários. Os comentários são dispostos na tela com a opção de serem classificados dentro de uma das categorias: (i) Requisito Funcional, (ii) Requisito Não Funcional e (iii) Outros.

Para realizar essa tarefa, o aplicativo de *smartphone* citado anteriormente foi utilizado. Esse aplicativo tem a função de facilitar a busca e a classificação dos comentários. O fluxo de utilização é:

- **Buscar novos comentários**: Comentários não classificados são buscados na base para que sejam classificados entre as categorias disponíveis;
- **Dispor comentários na tela**: Os comentários aparecem na tela em forma de lista, como mostra a Figura 15;
- **Classificar**: Os comentários apresentados são classificados entre Não funcional, Outro ou Funcional, e

- **Ir para o próximo comentário**: Após a classificação de um comentário, parte-se para outro comentário, repetindo o processo a partir do passo 2 "Dispor comentários na tela".

Uma vez que a lista fica sem comentários, o aplicativo fica responsável por requisitar mais comentários ao servidor.

6.1.1 Arquitetura Cliente-Servidor

A Figura 16 mostra a arquitetura da solução de software para o aplicativo. A arquitetura conta com dois componentes: (i) o cliente, construído em *React Native* (REACT..., 2018), o qual é responsável por apresentar os comentários e permitir que o usuário classifique-os nas categorias apresentadas na Seção 5.1.3.1, e (ii) o servidor, o qual é responsável por (a) guardar a informação no banco de dados, (b) processar a informação, (c) classificar os comentários nas categorias desejadas, e (d) disponibilizar informações para o cliente, quando requisitado, na forma HTTP. A arquitetura foi construída utilizando-se as tecnologias descritas no Capítulo 3. A comunicação entre cliente e servidor acontece via internet, obedecendo ao protocolo HTTP, e seguindo o padrão *RESTFUL*.

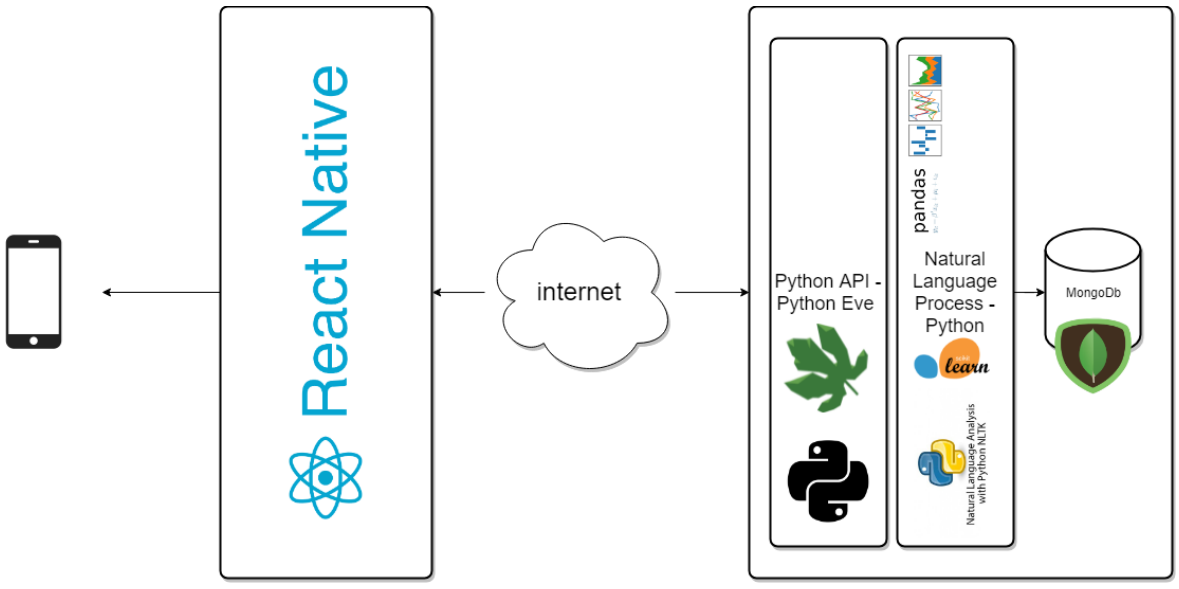

Figura 16 – Arquitetura da Aplicação Auxiliar

6.2 Ciclos de Pesquisa-Ação

Nessa Seção, são apresentados os cenários de uso, os quais foram especificados visando analisar o classificador desenvolvido para a aplicação, em especial, considerando

a conformidade dos resultados obtidos via classificador, comparando-os aos resultados coletados manualmente. Tem-se ainda o levantamento de várias estatísticas médias do classificador, tais como: *precision*, *recall*, *f1-score*, *support*, *macro avg*, *micro-avg* e *weighted avg*. Nessa etapa do projeto, orientou-se pela metodologia de análise de resultados, acordada no Capítulo 4. Os cenários de uso são utilizados nas descrições das iterações de pesquisa-ação. Eles descrevem as principais particularidades de cada iteração, tais como quais critérios são foco em cada iteração, dentro outros detalhes. A cada iteração, mudanças no classificador e no *dataset* foram feitas a fim de melhorar a conformidade dos resultados coletados com o classificador e os dados coletados manualmente, bem como melhorar as estatísticas médias. Percebe-se, nos cenários de uso, o apoio das tecnologias, as quais foram acordadas no Capítulo 3. O algoritmo *Multinomial Naive Bayes* foi utilizado, pois é indicado para a classificação de texto (BUITINCK et al., 2013).

6.2.1 Cenário de Uso 1

Realizada a atividade de classificar manualmente os comentários, a fim de obter uma quantidade de observações para treinamento do algoritmo de aprendizado de máquina, foi conduzido o Cenário de Uso 1. Este cenário é descrito a seguir.

- **Objetivo**: Construir um classificador capaz de identificar três classes distintas. Nesse caso, portanto, um classificador multinomial para as três categorias de comentários apresentadas na Seção 5.1.3.1;

- **Algoritmos Utilizados**: Os algoritmos utilizados para a construção do Cenário de Uso 1 foram *MultinomialNB* e *TfidfVectorizer*. Esses algoritmos estão disponíveis através da biblioteca, apresentada na Seção 3, Scikit-Learn (BUITINCK et al., 2013);

- **Parâmetros dos Algoritmo**: Nenhum parâmetro dos algoritmo foi modificado e/ou adaptado nesse Cenário de Uso 1. Sendo assim, os parâmetros foram os padrões de cada algoritmo da biblioteca Scikit-Learn (BUITINCK et al., 2013);

- ***Reviews* Utilizados**: Foram utilizados 4784 comentários classificados, em *Funcional requirement*, *Non-functional requirement* e *Other*. Com base na classificação manual, têm-se 3347 da categoria *Other*, 928 da categoria *Functional requirement* e 509 da categoria *Non-functional requirement*, e

- **Tamanho do Subconjunto de Treino e Teste**: Para treinar o algoritmo, foi utilizada uma relação de 70/30. Isso significa que o subconjunto de Treino corresponde a 70% dos comentários e o subconjunto de Teste a 30% dos comentários.

.

Obteve-se os seguintes resultados para esse Cenário:

	precision	**recall**	**f1-score**	**support**
Functional requirement	0	0	0	177
Non-functional requirement	0	0	0	120
Other	0,7	1	0,82	694
Micro avg	0,7	0,7	0,7	991
Macro avg	0,23	0,33	0,27	991
Weighted avg	0,49	0,7	0,58	991

Tabela 6 – Resultados do Classificador para as Três Categorias

Pode-se observar que os resultados do classificador foram baixos. O classificador errou todas as predições entre as categorias *Functional requirement* e *Non-functional requirement*. Entretanto, acertou 70% na categoria *Other*.

Esse efeito deve-se ao fato de que os consumidores não são pessoas treinadas, ou especializadas, em realizar comentários técnicos e informativos. Então, é natural que muitos dos *reviews* escritos não sejam algum tipo de requisição. Logo, pode-se esperar que o classificador tenha um viés para a categoria mais abundante, *Other*, uma vez que o *dataset* é desbalanceado.

De acordo com (YAP et al., 2014), há meios de lidar com *datasets* desbalanceados. As duas técnicas utilizadas são:

- ***Random Oversampling*** (ROS): consiste em duplicar exemplos aleatórios da classe minoritária, e
- ***Random UnderSampling*** (RUS): consiste em descartar exemplos aleatórios da classe maioritária.

Cabe ressaltar que *Oversampling* pode levar ao *Overfitting*, enquanto *Undersampling* pode remover exemplos importantes do *dataset*.

6.2.2 Cenário de Uso 2

Foi adotada a estratégia de (CASAMAYOR; GODOY; CAMPO, 2009) para a classificação das categorias de *Functional requirement* e *Non-functional requirement*. A estratégia consiste em elaborar um classificador binário para cada categoria desejada. Isto é, um classificador para identificar se um comentário é da categoria “Funcional” ou

não e “Não funcional” ou não. Optou-se por não criar um classificador para a categoria *Other*, pois, comentários dessa categoria não são foco primário para o estudo.

Dessa maneira, foram criados classificadores individuais para cada categoria. Para que isso fosse possível, uma mudança em como o conjunto de dados, *dataset*, é visto foi realizada. O *dataset* para a classificação entre todas as categorias, Cenário de Uso 1, possuía os valores alvo, *requirement type*. Para os classificadores individuais, dois novos *datasets* foram criados. Para cada categoria, seu *dataset* corresponde a uma mudança na variável alvo, *requirement type*, de maneira que a palavra “sim” corresponde aos comentários, cuja categoria é a desejada e “não” quaisquer outras categoria. Por exemplo, para a categoria “funcionais”, onde, no *dataset* original, os comentários estavam classificados como *Functional*, foram reclassificados para ‘sim’, e quaisquer outras categoria, *Non-functional* e *Other*, reclassificados para "nao".

- **Objetivo**: Construir classificadores binários para as categorias *Functional requirement* e *Non-functional Requirement*. Isto é, dois classificadores que identifiquem somente uma categoria de requisito, *Funcional* e *Non-functional*, cada um;

- **Algoritmos Utilizados**: Os algoritmos utilizados para a construção do Cenário de Uso 2 foram *MultinomialNB* e *TfidfVectorizer*. Esses algoritmos estão disponíveis através da biblioteca, apresentada no Capítulo 3, Scikit-Learn (BUITINCK et al., 2013);

- **Parâmetros dos Algoritmos**: Nenhum parâmetro dos algoritmo foi modificado e/ou adaptado nesse Cenário de Uso 2. Sendo assim, os parâmetros foram os padrões de cada algoritmo da biblioteca Scikit-Learn (BUITINCK et al., 2013);

- ***Reviews* Utilizados**: Foram utilizados 4784 comentários classificados entre *Funcional requirement*, *Non-functional requirement* e *Other*. Com base na classificação manual, têm-se 3347 da categoria *Other*, 928 da categoria *Functional requirement* e 509 da categoria *Non-functional requirement*, e

- **Tamanho do Subconjunto de Treino e Teste**: Para treinar o algoritmo, foi utilizada uma relação de 70/30. Isso significa que o subconjunto de Treino corresponde a 70% dos comentários e o subconjunto de Teste a 30% dos comentários.

Os passos de pré-processamento realizados anteriormente foram aplicados na construção dos classificadores individuais. Os resultados dos classificadores individuais podem ser vistos nas Tabelas 7 e 8.

	Precision	recall	f1-score	support
nao	0,82	1	0,9	814
sim	0	0	0	177
micro avg	0,82	0,82	0,82	991
macro avg	0,41	0,5	0,45	991
weighted avg	0,67	0,82	0,74	991

Tabela 7 – Resultado do Classificador Individual para a Categoria *Functional*

	Precision	recall	f1-score	support
nao	0,88	1	0,94	871
sim	0	0	0	120
micro avg	0,88	0,88	0,88	991
macro avg	0,44	0,5	0,47	991
weighted avg	0,77	0,88	0,82	991

Tabela 8 – Resultado do Classificador Individual para a Categoria *Non-functional*

Mais uma vez, observou-se que os classificadores não obtiveram resultados satisfatórios, errando na classificação das categorias alvo por completo, enquanto conseguem predizer, com alta confiabilidade, a categoria "não". Novamente, como já mencionado anteriormente na Seção 6.2.1, tais não conformidades devem-se ao fato do *dataset* ser desbalanceado.

6.2.3 Cenário de Uso 3

Para o Cenário de Uso 3, foi aplicada a técnica de *Undersampling*, descrita na Seção 6.2.1. Com auxílio da biblioteca Pandas, apresentada no Capítulo 3, foram retirados, aleatoriamente, comentários que não pertencia à categoria desejada, de maneira que o *dataset* ficasse balanceado.

Os comentários retirados fazem parte das outras duas categorias que não são o requisito alvo. Por exemplo, para o classificador balanceado da categoria *Functional*, os comentários retirados faziam parte dos comentários das categorias *Non-functional* e *Other*. Para o classificador balanceado de *Non-functional*, os comentários das categorias *Functional* e *Other* foram retirados.

- **Objetivo**: Construir classificadores binários balanceados, para as categorias *Functional requirement* e *Non-functional Requirement*, utilizando a técnica de *Random Undersampling*, apresentada na Seção 6.2.1;
- **Algoritmos Utilizados**: Os algoritmos utilizados para a construção do Cenário de Uso 3 foram *MultinomialNB* e *TfidfVectorizer*. Esses algoritmos estão disponíveis

através da biblioteca, apresentada no Capítulo 3, Scikit-Learn (BUITINCK et al., 2013) e correspondem ao, algoritmo de *Naive Bayes* e a transformação dos documentos na matriz Tf-idf, apresentados na Seções 2.5.1;

- **Parâmetros dos Algoritmos**: Nenhum parâmetro dos algoritmos foi modificado e/ou adaptado para o Cenário de Uso 3. Sendo assim, os parâmetros foram os padrões de cada algoritmo da biblioteca Scikit-Learn (BUITINCK et al., 2013);
- ***Reviews* Utilizados**: Foram utilizados 2763 comentários para o classificador da categoria *Functional requirement* e 2510 comentários para o classificador da categoria *Non-functional requirement*. Com base na classificação manual, têm-se para o classificador da categoria *Functional requirement*, 1835 da categoria 'nao' e 958 da categoria 'sim'. E para o classificador da categoria *Non-functional requirement*, têm-se, 2001 da categoria 'nao' e 509 da categoria 'sim', e
- **Tamanho do Subconjunto de Treino e Teste**: Para treinar o algoritmo, foi utilizada uma relação de 70/30. Isso significa que o subconjunto de Treino corresponde a 70% dos comentários e o subconjunto de Teste a 30% dos comentários.

.

As Tabelas 9 e 10 mostram, respectivamente, os resultados de desempenho para o Classificador de Requisito Funcional e o Classificador de Requisito Não Funcional com uso de *datasets* balanceados.

	Precision	**recall**	**f1-score**	**support**
nao	0,7	0,99	0,82	392
sim	0,77	0,06	0,11	173
micro avg	0,71	0,71	0,71	565
macro avg	0,74	0,53	0,47	565
weighted avg	0,72	0,71	0,6	565

Tabela 9 – Resultados do Classificador Balanceado para a Categoria *Functional*

	Precision	**recall**	**f1-score**	**support**
nao	0,75	1	0,86	397
sim	1	0,01	0,02	130
micro avg	0,76	0,76	0,76	527
macro avg	0,88	0,5	0,44	527
weighted avg	0,82	0,76	0,65	527

Tabela 10 – Resultados do Classificador Balanceado para a Categoria *Non-functional*

Novamente, observou-se que os classificadores não obtiveram resultados satisfatórios. Apesar do resultado alto da métrica *precision*, o valor de *recall* é consideravelmente

baixo. Com o *recall* baixo, significa que o classificador falhou em identificar muitos comentários das categorias alvo.

6.2.4 Cenário de Uso 4

Para o Cenário de Uso 4, foi mantido o uso de *datasets* balanceados, assim como no Cenário de Uso 3. O foco desse Cenário foi a utilização de parâmetros nos algoritmos para verificar as mudanças nos classificadores.

Os parâmetros foram modificados através de iterações. Cada iteração, mudando seu valor a fim de obter o melhor resultado das métricas estabelecidas na Seção 2.5.3. A descrição desse Cenário é acordada a seguir.

- **Objetivo**: construir classificadores binários balanceados, para as categorias *Functional requirement* e *Non-functional Requirement*, utilizando a técnica de *Random Undersampling*, apresentada na Seção 6.2.1. Adicionalmente, realizar mudanças nos parâmetros dos algoritmos utilizados a fim de obter resultados mais satisfatórios;

- **Algoritmos Utilizados**: Os algoritmos utilizados para a construção do Cenário de Uso 4 foram *MultinomialNB* e *TfidfVectorizer*. Esses algoritmos estão disponíveis através da biblioteca, apresentada no Capítulo 3, Scikit-Learn (BUITINCK et al., 2013), e correspondem ao algoritmo de *Naive Bayes*, e a transformação dos documentos na matriz Tf-idf, apresentados na Seções 2.5.1;

- **Parâmetros dos Algoritmos**: Os parâmetros modificados para esse cenário de uso foram *ngram_range* e *max_features*, para o algoritmo *TfidfVectorizer*, e *alpha*, para o algoritmo, *MultinomialNB* (BUITINCK et al., 2013);

- ***Reviews* Utilizados**: Foram utilizados 2763 comentários para o classificador da categoria *Functional requirement* e 2510 comentários para o classificador da categoria *Non-functional requirement*. Com base na classificação manual, têm-se para o classificador da categoria *Functional requirement*, 1835 da categoria 'nao' e 958 da categoria 'sim'. E para o classificador da categoria *Non-functional requirement*, têm-se, 2001 da categoria 'nao' e 509 da categoria 'sim', e

- **Tamanho do Subconjunto de Treino e Teste**: para treinar o algoritmo, foi utilizada uma relação de 70/30. Isso significa que o subconjunto de Treino corresponde a 70% dos comentários e o subconjunto de Teste a 30% dos comentários.

6.2.4.1 Determinando Valor de *alpha* para o Classificador de Requisitos Funcionais

De acordo com (SKLEARN. . . , 2018b), o algoritmo *MultinomialNB* possui três parâmetros de função, sendo um deles o parâmetro *alpha*. O parâmetro *alpha* é a suavização da maior semelhança. Isso significa que *alpha* aumenta a probabilidade de acontecimento de um evento. No caso de classificação de texto, entende-se como a probabilidade de uma palavra fazer parte de uma determinada categoria.

A Figura 17 mostra os resultados do algoritmo para as métricas *Precision* e *Recall* com mudanças no parâmetro *alpha* do algoritmo *MultinomialNB* (SKLEARN. . . , 2018b). Com auxílio da Figura 17, pode-se observar que para valores onde *alpha* é próximo de zero, há um aumento do valor do *Recall*, cuja métrica era a mais baixa nos Cenários de Uso anteriores.

O aumento na métrica *Recall* é interessante, pois significa que o algoritmo possui maior capacidade de identificar documentos da categoria desejada. Por essa análise, pode-se adotar $alpha = 0.01$ como parâmetro para o algoritmo *MultinomialNB*.

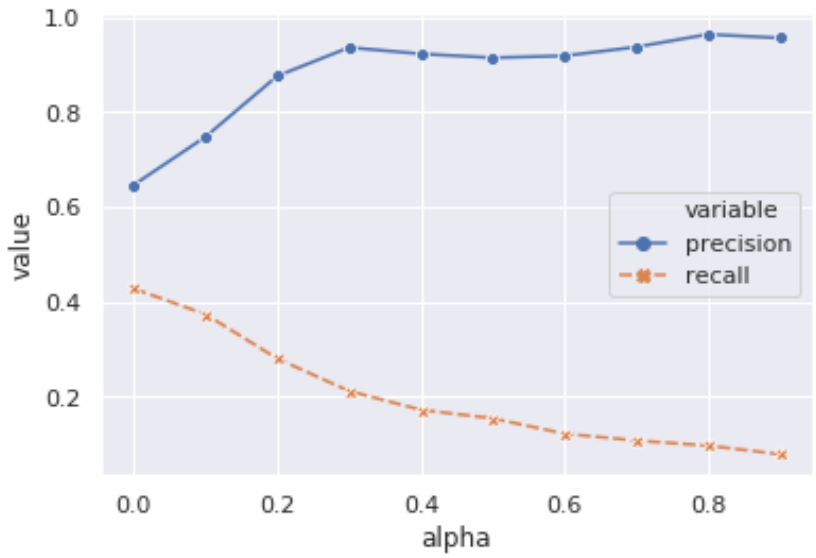

Figura 17 – Resultados das métricas *Precision* e *Recall* com a mudança numérica do parâmetro *alpha* para o Classificador de Requisitos Funcionais

6.2.4.2 Determinando Valor de *max_features* para o Classificador de Requisitos Funcionais

Adicionalmente, durante a fase de transformação dos documentos na matriz *TD-IDF*, é possível, de acordo com (SKLEARN. . . , 2018c), limitar o número máximo de palavras na matriz. Essas palavras são ordenadas por frequência, e as N primeiras continuam na matriz, enquanto o restante é desconsiderado.

A Figura 18 mostra os resultados das métricas *Precision* e *Recall* para a mudança numérica do parâmetro *max_features* do algoritmo *TfidfVectorizer* (SKLEARN..., 2018c).

Novamente, pode-se observar o aumento da métrica *Recall* quando o valor de *max_features* é 3500. Por essa análise, pode-se adotar *max_features = 3500* como parâmetro para o algoritmo *TfidfVectorizer*.

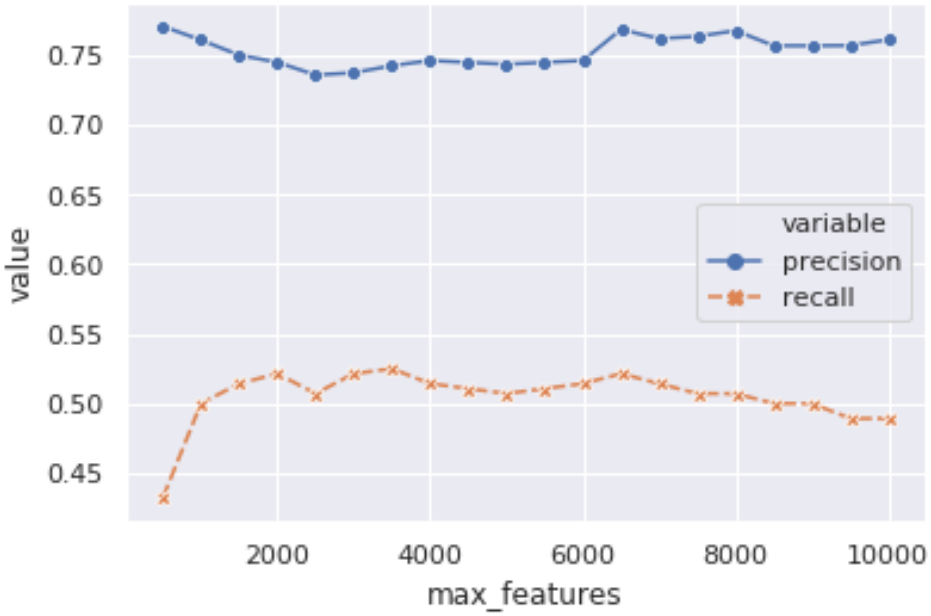

Figura 18 – Resultados das métricas *Precision* e *Recall* com a mudança numérica do parâmetro *max_features* para o Classificador de Requisitos Funcionais

6.2.4.3 Determinando Valor de *alpha* para o Classificador de Requisitos Não Funcionais

Uma vez que esse Cenário de Uso também faz uso de dois classificadores binários, é necessário realizar as atividades anteriores para o Classificador de Requisitos Não funcionais. As atividades seguiram o mesmo rigor das realizadas para o classificador de Requisitos Funcionais. A Figura 19 mostra os resultados das métricas *Precision* e *Recall* para o parâmetro *alpha* do algorimo *MultinomialNB* (SKLEARN..., 2018b).

Pode-se observar o aumento da métrica *Recall*, quando o valor de *alpha* é próximo de zero. Por essa análise, pode-se adotar *alpha = 0.01*, assim como no Classificador de Requisitos Funcionais, como parâmetro para o algoritmo *MultinomialNB*.

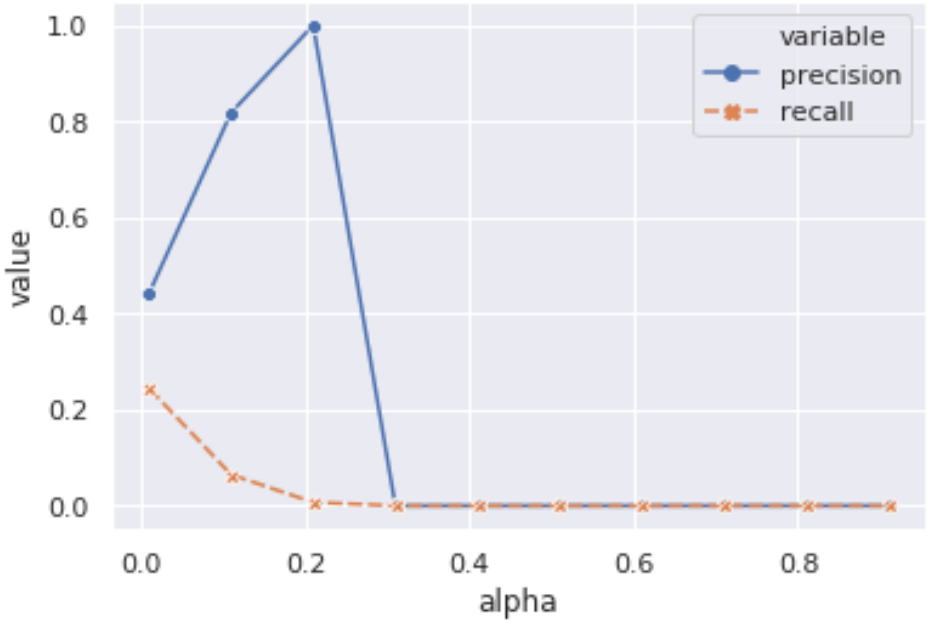

Figura 19 – Resultados das métricas *Precision* e *Recall* com a mudança numérica do parâmetro *alpha* para o Classificador de Requisitos Não Funcionais

6.2.4.4 Determinando Valor de *max_features* para o Classificador de Requisitos Não Funcionais

Também foi realizada a atividade de identificação do valor numérico para o parâmetro *max_features* do algoritmo *TfidfVectorizer* (SKLEARN..., 2018c) para o Classificador de Requisitos Não Funcionais. A Figura 20 mostra o resultado das métricas *Precision* e *Recall* para o parâmetro *max_features*.

Foi identificado o valor de *max_features*. Por essa análise, pode-se adotar *max_features = 2500* como parâmetro para o algoritmo *TfidfVectorizer*.

As Tabelas 11 e 12 mostram os resultados para o Classificador de Requisito Funcional e Classificador de Requisito Não Funcional com uso de *datasets* balanceados e mudança de parâmetros dos algoritmos.

Classificador Requisitos Funcionais	**precision**	**recall**	**f1-score**	**support**
nao	0,79	0,91	0,84	551
sim	0,74	0,52	0,61	280
micro avg	0,78	0,78	0,78	831
macro avg	0,76	0,71	0,73	831
weighted avg	0,77	0,78	0,76	831

Tabela 11 – Resultados do Classificador Balanceado com Parâmetros *alpha = 0.01* e *max_features = 3500* para a Categoria *functional*

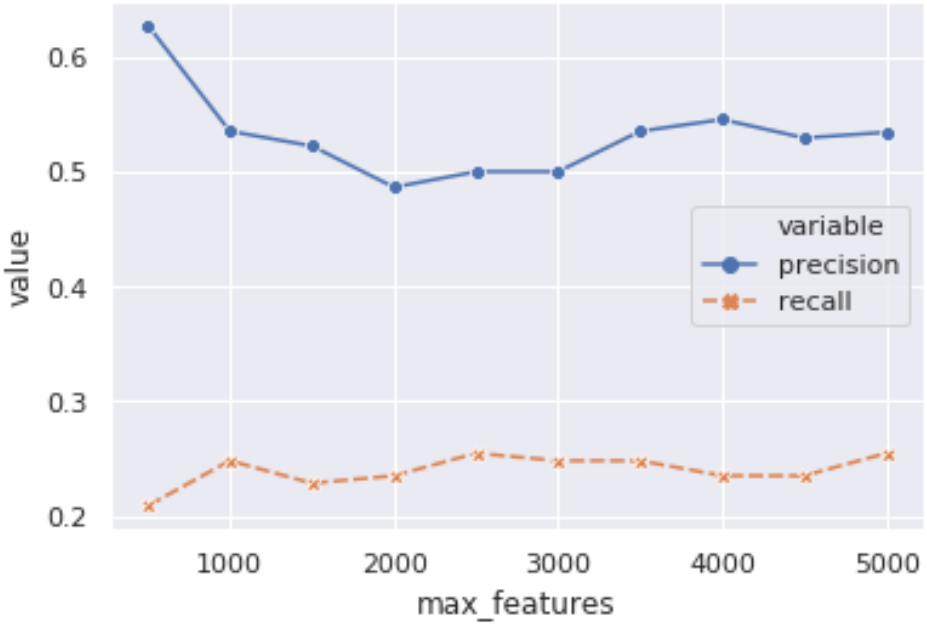

Figura 20 – Resultados das métricas *Precision* e *Recall* com a mudança numérica do parâmetro *max_features* para o Classificador de Requisitos Não Funcionais

Classificador Requisitos Não Funcionais	**Precision**	**recall**	**f1-score**	**support**
nao	0,83	0,94	0,88	601
sim	0,5	0,25	0,34	153
micro avg	0,8	0,8	0,8	754
macro avg	0,67	0,6	0,61	754
weighted avg	0,76	0,8	0,77	754

Tabela 12 – Resultados do Classificador Balanceado com Parâmetros *alpha = 0.01* e *max_features = 2500* para a Categoria *Non-functional*

6.3 Resultados Obtidos na Clusterização

Nessa Seção, são apresentados os resultados obtidos para a clusterização dos comentários classificados manualmente. A clusterização foi realizada com o algoritmo de *Kmeans*, apresentado na Seção 2.5.2. O algoritmo de *Kmeans* é um algoritmo de aprendizado não supervisionado, por isso não requer exemplos previamente classificados.

6.3.1 Cenário de Uso 1

Para o Cenário de Uso 1, foram utilizados os comentários dos aplicativos que haviam sido classificados entre as categorias apresentadas anteriormente. Apesar de não ser necessária a classificação neste caso, pois o algoritmo está na categoria dos não supervisionados. O foco desse Cenário foi a identificação de palavras-chave, e tópicos, para cada

aplicativo desse *dataset*.

- **Objetivo**: Identificar, através do uso do algoritmo de *Kmeans*, palavras-chave para cada aplicativo entre as categorias de Requisitos Funcionais e Requisitos Não Funcionais. Sendo essas palavras-chave subdivididas em quatro *clusters*, caracterizando quatro possíveis tópicos distintos;
- **Algoritmos Utilizados**: Os algoritmos utilizados para a construção do Cenário 1 foram *Kmeans* (SKLEARN..., 2018a) e o *TfidfVectorizer* (SKLEARN..., 2018c). O *TfidfVectorizer* foi utilizado, pois há a necessidade de transformar palavras para uma matriz numérica, a fim de ser utilizada na clusterização;
- **Parâmetros dos Algoritmos**: Os parâmetros modificados para esse cenário de uso foram *ngram_range* e *max_features*, para o algoritmo *TfidfVectorizer*. Esses parâmetros são os mesmos adotados na fase de classificação, na Seção 6.2.4. Para o algoritmo *Kmeans*, foi adotado o parâmetro $n\_clusters = 3$. Esse parâmetro define o número de clusters utilizados, e
- ***Reviews* Utilizados**: foram utilizados 928 comentários da categoria *Funcional requirement* e 509 comentários da categoria *Non-functional requirement*.

.

6.3.1.1 Palavras-chave Identificadas para Requisitos Funcionais

A Tabela 13 mostra o resultado da clusterização para identificação de palavras-chave sobre os Requisitos Funcionais de cinco aplicativos. Optou-se por apresentar os aplicativos pelo seu código uma vez que o *dataset*, provido originalmente por (MCAULEY et al., 2015), não os identifica por nome.

6.3.1.2 Palavras-chave Identificadas para Requisitos Não Funcionais

A Tabela 14 mostra o resultado da clusterização para identificação de palavras-chave sobre os Requisitos Não Funcionais de cinco aplicativos. Optou-se por apresentar os aplicativos pelo seu código uma vez que o *dataset*, provido originalmente por (MCAULEY et al., 2015), não os identifica por nome.

Código do App	cluster 0	cluster 1	cluster 2	cluster 3
B004A9SDD8	'monkeys', 'wish', 'just', 'picture', 't', 'graphics don', 'graphics', 'basicly', 'waste money', 'basicly just'	'app', 't', 'm', 'force close', 'close', 'force', 'application error', 'paid', 'enjoy', 'application'	'love', 'love love', 'fine', 'objects', 'song', 'son', 'different', 'tablet', 'fine force', 'cute'	' app', 'daughter', 'better', '34', '3', 'app 3', 'old daughter', 'old', 's', 'option'
B004AFQAUA	'like', 'playlist', 'pandora', 'music', 'download', 'app', 'monthly fee', 'monthly', 'like pandora', 'fee'	't', 'don', 'don t', 'music', 'computer', 'song', 'sign', 'streaming', 'play', 'wifi'	music', 'rhapsody', 'love', 'listen', 'play', 've', 'years', 'songs', 'time', 'used'	'kindle', 'app', 'service', 'work', 't', 'rhapsody', 'great', 'access', 'able', 'account'
B004AHBBPW	'love', 'daily verses', 'wish', 'wish versions', 'background', 'daily', 'love daily', 'verses', 'versions', 'love app'	'daily', 'like', 'bible', 'daily bible', 'daily verse', 't like', 'verse', 'line', 'reading', 't'	app', 'bible', 'bible app', 'read', 'kjv', 'wifi', 'connected', 'internet', 'use', 'listen'	'problem', 'fixed', 'error', 'update', 't', 'app', 'chapter', 's', 'works', 'blank'
B004ALVL6W	'buy', 'horrible', 'shot', 'recharge', 'tower', 'absolutely useless', 'absolutely', 'game horrible', 'base shot', 'useless'	'game', 'play', 'tower', 'uninstalled', 'played', 't', 'star', 'tower defense', 'different', 'defense game'	'towers', 'shoot', 't', 'don', 'don t', 'time', 'works', 'stuped', 'waste time', 'money'	'like', 'defense', 'music', 'game', 'worth', 'volume', 'pop', 'graphics', 'app', 'easy'
B004ANC00Q	't', 'app', 'apos', 'apos t', 'work', 'location', 'doesn', 'doesn apos', 'right', 'use'	'version', 'bluetooth', 'plugin', 'market', 'automatically', 'bluetooth plugin', 'android market', 'android', 'market version', 'plugin automatically'	'app', 'car', 'said', 'need', 'flip', 'said car', 'nice', 'away', 'use app', 'time'	'great', 'app', 'good', 'like', 'exit', 'button', 'card', 'sd card', 'turn', 'sd'

Tabela 13 – Resultados da Clusterização para os Requisitos Funcionais de cinco aplicativos.

Código do App	cluster 0	cluster 1	cluster 2	cluster 3
B004A9SDD8	'kindle', 'quot', 'downloaded kindle', 'loves easy', 'grandson play', 'grandson', 'play loves', 'playand', 'playand educational', 'educational'	'kids', 'old', 'month old', 'month', 'seven month', 'sit', 'sit play', 'son loves', 'old son', 'recommend kids'	'app', 'screen', 'does', 'works', 'daughter loves', 'update', 'size', 'size screen', 'tablet', 'glad'	'little', 'play', 'really', 'instead', 'stupid', 'stupid animation', 'just', 'just stupid', 'thought', 'thought book'
B004AFQAUA	'kindle', 'love', 'rhapsody', 'install', 'glad', 'phone', 'love love', 'use', 'songs', 'download'	'don t', 'don', 't', 'great', 'product', 'use', 'subscription', 'month', 'songs', 'money'	'just', 'app', 'better', 's', 'pandora', 'user', 'song', 'love', 'user friendly', 'friendly'	'app', 'music', 'rhapsody', 'library', 'service', 'artists', 'android', 'use', 'download', 'easy'
B004AHBBPW	'easy', 'use', 'easy use', 'love', 'app easy', 'app', 'day', 'use day', 'devotions', '34'	'able', 'good', 'save', 'verse', 'phone', 'particular verse', 'hand corner', 'verse save', 'corner screen', 'hand'	'app', 'great', 'bible', 'read', 'hard', 'great app', 'daily', 'god', 'follow', 'like'	'iphone', 'know', 'available', 'app', 'time length', 'awhile know', 'awhile', 'does know', 'app does', 's awhile'
B004ALVL6W	'play', 'responsive', 'clear', 'tutorial', 'instructions', 'easy', 'just', 'frustrating', 'fix', 'extra star'	'game', 's', 'towers', 'tower', 'power', 'station', 'place', 'pretty', 'bad', 'power station'	't', 'don', 'don t', 'game', 'fun', 'towers', 'time', 'couldn', 'couldn t', 'just'	'app', 'game', 'games', 'good', 'like', 'just', 's', 'really', 't', 'people'
B004ANC00Q	'use', 'battery', 'drained battery', 'drained', 'opened', 'use future', 'downloaded use', 'future date', 'battery removed', 'opened drained'	'apos', 'apos t', 't', 'permissions', 'apos s', 'install', 's', 'app', 'just', 'contacts'	'car', 'battery', 'walking', 'phone', 'directions', '3', 'app', 'year', 'really', 'like'	'info', 'need', 'personal', 'unnecessary', 'app', 'access', 'permissions', 'does need', 'send receive', 'receive'

Tabela 14 – Resultados da Clusterização para os Requisitos Não Funcionais de cinco aplicativos.

Observa-se que os *clusters* mostram palavras-chave dos comentários classificados manualmente das categorias *Functional* e *Non-functional*. Para algumas aplicações, é possível ver como os usuários se expressam quando comentam sobre aspectos de qualidade e sobre funcionalidades. Há *clusters* que indicam palavras-chave sobre funcionalidades como *version*, *plugin*, *bluetooth*, *sd card*. E *clusters* que apresentam aspectos de qualidade por meio de palavras-chave como: *size screen*, *frustrating*, *permissions*, *battery*.

6.4 Resumo do Capítulo

Este capítulo apresentou-se os resultados alcançados com o término desse trabalho. Os resultados refletem à questão de pesquisa, aos objetivos específicos e geral identificados na Seção 1.2. Apresentou-se como foi realizada a classificação dos comentários nas categorias *Functional*, *Non-functional* e *Other*. Adicionalmente, foram apresentados o aplicativo de *smartphone* que proveu suporte à classificação, bem como a arquitetura do servidor para salvar a informação.

Também foram apresentados os Ciclos de Pesquisa-Ação realizados para o desenvolvimento dos classificadores, mostrando que o desenvolvimento de dois classificadores individuais, um para a categoria *Functional* e outro para a categoria *Non-functional*. Ambos os classificadores obtiveram resultados satisfatórios após a mudança de alguns parâmetros nos algoritmos. E por fim, foi apresentado o resultado da clusterização desses comentários. Os comentários foram divididos em quatro *clusters* e as palavras-chave apresentadas na Seção 6.3.

7 Considerações Finais

O presente trabalho acorda a dificuldade de gerentes de software no que se refere à comunicação com os usuários finais dos produtos de software desenvolvidos. Adicionalmente, e com base em autores da área, argumenta-se que os métodos de elicitação tradicionais perdem a oportunidade de envolver um número grande de usuários. Visando lidar com esse *gap* tecnológico, o trabalho sugere o uso de *Crowd-Base Requirements Engineering (CrowdRE)*.

Nesse capítulo, serão retomadas algumas preocupações, colocações e contribuições desse trabalho, bem como serão apresentadas sugestões para trabalhos futuros. Dessa forma, tem-se: na Seção 7.1, a retomada de alguns aspectos chave desse projeto; na Seção 7.2, são acordadas as principais contribuições do mesmo, evidenciando os objetivos alcançados bem como a questão de pesquisa, e na Seção 7.3, são sugeridas ideias para trabalhos futuros, evoluindo o presente projeto.

7.1 Aspectos Chave

Técnicas de Elicitação tradicionais, em sua maioria, costumam ser atividades focadas na investigação de alguns usuários chave. Mesmo quando realizadas para um público alvo, como questionários, demandam recursos humanos e financeiros razoáveis para serem aplicadas e avaliadas considerando um grande grupo de pessoas.

Por outro lado, hoje têm-se que muitos aspectos podem ser elicitados diretamente via os próprios usuários dos produtos, através da investigação de redes sociais e comentários providos por esses usuários sobre o uso daqueles produtos. Esses aspectos podem ser muito úteis no processo de elicitação de requisitos, sejam esses funcionalidades ou critérios de qualidade. Entretanto, usar técnicas tradicionais tanto para avaliar um grande número de usuários, quanto para elicitar algo em linguagem natural, não é um processo viável, segundo (GROEN et al., 2017).

Diante do exposto, tem-se a necessidade de pesquisar recursos para lidar com vários aspectos chave, dentre eles:

- número grande de usuários (GROEN et al., 2017);
- informações relevantes disponíveis tanto em grande número, quanto, na maioria dos casos, em linguagem natural (GROEN et al., 2017);
- rápido atendimento às demandas de mercado (CARREñO; WINBLADH, 2013), e

- evolução constante dos produtos de software, dadas as exigências e impressões desses produtos quando implantados no mercado (BASOLE; KARLA, 2011).

O presente trabalho procurou contribuir nessa linha de pesquisa, propondo uma solução computacional orientada à *Crowd-Based Requirements Engineering (CrowdRE)*. Na próxima Seção, são apresentadas as principais contribuições alcançadas até o momento.

7.2 Contribuições da Pesquisa

O classificador desenvolvido nesse trabalho foi proposto com o objetivo de auxiliar a priorização de funcionalidades e de melhorias. A questão, apresentada na Seção 1.2, que buscou-se responder foi: *Há a possibilidade de auxiliar gerentes de software na priorização de funcionalidades e de melhorias utilizando grande quantidade de informações?*

Para solucionar essa questão, foi construído um classificador de requisitos de software, utilizando processamento de linguagem natural e aprendizado de máquina. Para a construção desse classificador fez-se uso da metodologia, apresentada no Capítulo 4.

Uma vez que o método de aprendizagem adotado no trabalho é o supervisionado, uma quantidade de comentários teve que ser classificada manualmente, sendo esta a maior dificuldade do trabalho. Entretanto, apesar de ter sido a maior dificuldade, não levou ao atraso, nem ao impedimento, de nenhuma outra atividade proposta no trabalho.

O resultado das atividades realizadas durante este trabalho foi o desenvolvimento de dois classificadores para identificar requisitos a partir de comentários nas lojas de aplicativos. Um classificador para identificar Requisitos Funcionais e outro para identificar Requisitos Não Funcionais, resultados expostos no Capítulo 5.

Através de ciclos de pesquisa-ação, foi possível realizar diferentes Cenários de Uso para os classificadores. Adicionalmente, foi realizada a identificação de palavras-chave dos aplicativos referentes aos comentários classificados manualmente. Foram separados em três *clusters* de palavras-chave para comentários identificados como Requisitos Funcionais, e três *clusters* de palavras-chave para comentários identificados como Requisitos Não Funcionais.

Por meio da análise dos Cenários de Uso, apresentados no Capítulo 5, a questão de pesquisa, acordada na Seção 1.2, e conforme a Tabela 15, é possível concluir que o projeto, de modo geral, obteve sucesso.

A partir do desenvolvimento desse trabalho, foi possível aplicar conceitos e conhecimentos estudados durante o curso de Engenharia de Software. Entre as disciplinas, destacam-se: Engenharia de Requisitos e Gerência de Configuração de Software. O trabalho é centrado na Engenharia de Requisitos, o que permitiu que fossem identificadas novas

Objetivos	Tipo	Completude
Há a possibilidade de auxiliar gerentes de software na priorização de funcionalidades e de melhorias utilizando grande quantidade de informações?	Questão de Pesquisa	100%
Criar um classificador automático de requisições de usuário, expressas em linguagem natural.	Geral	100%
Identificar regras para pré-processar os dados coletados	Específico	100%
Propor um processo ou um fluxo de atividades para pré-processamento de dados	Específico	100%
Identificar um algorítmo de aprendizado de máquina	Específico	100%
Classificar algumas avaliações para cada classe de requisitos selecionada para o trabalho. O número de avaliações será definido ao longo da pesquisa;	Específico	100%
Desenvolvimento da aplicação proposta	Específico	100%

Tabela 15 – Objetivos Gerais e Específicos e suas Determinadas Completudes

maneiras de elicitação de requisitos, e também novas maneiras de comunicação entre os desenvolvedores e os usuários do software.

Como o tema aqui investigado é abrangente, tem-se que esse projeto poderia ser evoluído em várias frentes de pesquisa. Algumas ideias de trabalhos futuros podem ser conhecidas na próxima Seção.

7.3 Trabalhos Futuros

Ressalta-se que ainda há melhorias que podem ser realizadas no trabalho de responder à questão de pesquisa. Algumas sugestões para trabalhos futuros são:

- **Comparação com outros Algoritmos de Aprendizado de Máquina**: esse trabalho utilizou o algoritmo de *Naive-Bayes* para construir dois classificadores. O teste com outros algoritmos tais como Redes Neurais, *Support Vector Machines*,

Nearest Neighbors, é uma oportunidade para comparar, e possivelmente melhorar, os resultados obtidos na classificação.

- **Identificação de Requisitos de Software em outros Canais de Comunicação**: esse trabalho buscou identificar requisitos de software sobre comentários realizados em lojas de aplicativos, tais como Apple Store®, Google Play® e Amazon Appstore®. Entretanto, há outros diversos meios de comunicação entre usuários e desenvolvedores, como por exemplo: Facebook® e Twitter®. (WILLIAMS; MAHMOUD, 2017), (GUZMAN; IBRAHIM; GLINZ, 2017) e (GUZMAN; ALKADHI; SEYFF, 2016) mostram estudos sobre a comunicação do usuário na plataforma Twitter®. Esses estudos procuraram identificar informações relevantes, disponibilizadas através de *Tweets* para Engenheiros de software, e responder a perguntas como: *Qual o tipo de conteúdo está presente em Tweets relacionados aos produtos de software?*, *O conteúdo dos Tweets é relevante aos stakeholders?*, e *Até que ponto é possível classificar Tweets como informação relevante para produtos de software?*. Pode-se aplicar o mesmo estudo do presente trabalho para *Tweets*, buscando identificar Requisitos Funcionais e Não Funcionais.

Referências

AGARWAL, P. K.; MUSTA, N. H. k-means projective clustering. In: *PODS '04 Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. [s.n.], 2004. p. 155–165. Disponível em: <https://dl.acm.org/citation.cfm?id=1055581>. Citado na página 43.

ALFONS, C.; LIND, E. F. P. Nearest neighbor classifiers. 2009. Disponível em: <https://pdfs.semanticscholar.org/35fb/29020ad9eb5459877048541fc329bf25ed65.pdf>. Citado na página 42.

ARTHUR, D.; VASSILVITSKII, S. k-means++: The advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics*. [s.n.], 2007. Disponível em: <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>. Citado na página 43.

BALAKRISHNAN, V.; LLOYD-YEMOH, E. The impact of social media on software engineering practices and tools. 2010. Citado na página 38.

BASOLE, R.; KARLA, J. Value transformation in the mobile service ecosystem: A study of app store emergence and growth. 2011. Disponível em: <https://pubsonline.informs.org/doi/abs/10.1287/serv.1120.0004>. Citado 2 vezes nas páginas 26 e 85.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural Language Processing with Python*. [S.l.], 2014. Disponível em: <https://www.nltk.org/book/ch06.html>. Citado 2 vezes nas páginas 15 e 61.

BONI, V.; QUARESMA, S. J. Aprendendo a entrevistar: como fazer entrevistas em ciências sociais. 2005. Disponível em: <https://periodicos.ufsc.br/index.php/emtese/article/viewFile/%2018027/16976>. Citado na página 33.

BRABHAM, D. C. Crowdsourcing as a model for problem solving: An introduction and cases. In: *Convergence 14 · February 2008*. [s.n.], 2008. p. 75–90. Disponível em: <https://www.researchgate.net/publication/239442893_Crowdsourcing_as_a_Model_for_Problem_Solving_An_Introduction_andCases>. Citado 2 vezes nas páginas 34 e 58.

BUITINCK, L. et al. API design for machine learning software: experiences from the scikit-learn project. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. [S.l.: s.n.], 2013. p. 108–122. Citado 5 vezes nas páginas 47, 70, 72, 74 e 75.

BURGESS, T. F. *A general introduction to the design of questionnaires for survey research*. [S.l.], 2001. Citado na página 33.

CARREñO, L. V. G.; WINBLADH, K. Analysis of user comments: An approach for software requirements evolution. In: *2013 35th International Conference on Software Engineering (ICSE)*. [s.n.], 2013. Disponível em: <http://ieeexplore.ieee.org/document/6606604/>. Citado 2 vezes nas páginas 26 e 84.

CASAMAYOR, A.; GODOY, D.; CAMPO, M. Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. In: *Information and Software Technology*. [s.n.], 2009. v. 52, p. 436–445. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0950584909001918>. Citado na página 71.

CLELAND-HUANG, J. et al. The detection and classification of non-functional requirements with application to early aspects. In: *14th IEEE International Requirements Engineering Conference (RE'06)*. [s.n.], 2006. Disponível em: <http://ieeexplore.ieee.org/document/1704047/>. Citado na página 27.

CS231N: Convolutional Neural Networks for Visual Recognition. 2017. Disponível em: <http://cs231n.github.io/classification/#nn>. Citado 2 vezes nas páginas 15 e 42.

FLASK. 2018. Disponível em: <http://flask.pocoo.org/>. Citado na página 47.

FONSECA, J. J. S. *Metodologia da pesquisa científica*. [S.l.], 2002. v. 4. Disponível em: <http://leg.ufpi.br/subsiteFiles/lapnex/arquivos/files/Apostila_-_METODOLOGIA_DA_PESQUISA%281%29.pdf>. Citado na página 51.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de Pesquisa*. [S.l.], 2009. v. 1. Citado 2 vezes nas páginas 50 e 55.

GIL, A. C. *Como Elaborar Projetos de Pesquisa*. [S.l.], 2002. v. 4. Citado na página 51.

GOGUEN, J.; LINDE, C. Techniques for requirements elicitation. In: *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*. [s.n.], 1993. Disponível em: <http://ieeexplore.ieee.org/document/324822/>. Citado 3 vezes nas páginas 26, 33 e 34.

GROEN, E. C.; DOERR, J.; ADAM, S. Towards crowd-based requirements engineering a research preview. In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*. [s.n.], 2015. Disponível em: <https://link.springer.com/chapter/10.1007%2F978-3-319-16101-3_16>. Citado 2 vezes nas páginas 26 e 58.

GROEN, E. C. et al. The crowd in requirements engineering: The landscape and challenges. In: *IEEE Software (Volume: 34, Issue: 2, Mar.-Apr. 2017)*. [s.n.], 2017. Disponível em: <http://ieeexplore.ieee.org/document/7888433/>. Citado 6 vezes nas páginas 26, 35, 36, 58, 59 e 84.

GUERSES, S. et al. Eliciting confidentiality requirements in practice. 2005. Disponível em: <https://dl.acm.org/citation.cfm?id=1105642>. Citado na página 25.

GUNDA, S. G. Requirements engineering: Elicitation techniques. 2008. Disponível em: <http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A215169&dswid=-782>. Citado 3 vezes nas páginas 26, 33 e 34.

GUTHRIE, D. et al. A closer look at skip-gram modelling. 2006. Disponível em: <https://www.researchgate.net/publication/266863668_A_Closer_Look_at_Skip-gram_Modelling>. Citado na página 39.

GUZMAN, E.; ALKADHI, R.; SEYFF, N. A needle in a haystack: What do twitter users say about software? In: *2016 IEEE 24th International Requirements Engineering Conference (RE)*. [s.n.], 2016. Disponível em: <http://ieeexplore.ieee.org/document/7765515/>. Citado na página 87.

GUZMAN, E.; EL-HALABY, M.; BRUEGGE, B. Ensemble methods for app review classification: An approach for software evolution. In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering*. [s.n.], 2015. Disponível em: <http://ieeexplore.ieee.org/document/7372065/>. Citado na página 41.

GUZMAN, E.; IBRAHIM, M.; GLINZ, M. A little bird told me: Mining tweets for requirements and software evolution. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. [s.n.], 2017. Disponível em: <http://ieeexplore.ieee.org/document/8048886/>. Citado na página 87.

GUZMAN, E.; MAALEJ., W. How do users like this feature? a fine grained sentiment analysis of app reviews. In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. [s.n.], 2014. Disponível em: <http://ieeexplore.ieee.org/document/6912257/>. Citado na página 27.

HOSSEINI, M. et al. Towards crowdsourcing for requirements engineering. 2014. Disponível em: <http://eprints.bournemouth.ac.uk/21904/>. Citado 4 vezes nas páginas 26, 35, 36 e 37.

KEIL, M.; CARMEL, E. Customer-developer links in software development. 1995. Disponível em: <https://dl.acm.org/citation.cfm?id=203363>. Citado na página 25.

KUJALA, S. User involvement: A review of the benefits and challenges. In: *Behaviour & Information Technology*. [s.n.], 2003. Disponível em: <https://www.researchgate.net/profile/Sari_Kujala/publication/220208710_User_involvement_A_review_of_the_benefits_and_challenges/links/00b49514ab5808a143000000/User-involvement-A-review-of-the-benefits-and-challenges.pdf>. Citado na página 25.

KUJALA, S. et al. The role of user involvement in requirements quality and project success. In: *13th IEEE International Conference on Requirements Engineering (RE'05)*. [s.n.], 2005. Disponível em: <http://ieeexplore.ieee.org/document/1704047/>. Citado na página 25.

LAKHOUA, M.; ANNABI, M. Approach of analysis of methods sa, sadt and sart. In: *Information and Communication Technologies, 2006. ICTTA '06. 2nd*. [s.n.], 2006. Disponível em: <https://ieeexplore.ieee.org/abstract/document/1684416/>. Citado 4 vezes nas páginas 15, 58, 59 e 60.

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. *Mining of Massive Datasets*. [S.l.], 2010. Disponível em: <http://infolab.stanford.edu/~ullman/mmds/book.pdf>. Citado na página 42.

LEVY, M.; HADAR, I.; TE'ENI, D. A gradual approach to crowd-based requirements engineering: The case of conference online social networks. In: *2015 IEEE 1st International Workshop on Crowd-Based Requirements Engineering (CrowdRE)*. [s.n.], 2015. Disponível em: <http://ieeexplore.ieee.org/document/7367585/>. Citado 3 vezes nas páginas 35, 58 e 59.

LEWIS, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. 2005. Disponível em: <https://link.springer.com/chapter/10.1007/BFb0026666>. Citado na página 41.

LI, C. et al. Automatically classifying user requests in crowdsourcing requirements engineering. In: *The Journal of Systems and Software*. [s.n.], 2017. v. 138. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121217303096>. Citado 4 vezes nas páginas 27, 28, 37 e 39.

LLOYD, S. P. Least squares quantization in pcm. In: *IEEE TRANSACTIONS ON INFORMATION THEORY*. [s.n.], 1957. p. 155–165. Disponível em: <https://dl.acm.org/citation.cfm?id=1055581>. Citado na página 43.

MAALEJ, W.; NABIL, H. Bug report, feature request, or simply praise? on automatically classifying app reviews. In: *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*. [s.n.], 2015. Disponível em: <http://ieeexplore.ieee.org/document/7320414/>. Citado 2 vezes nas páginas 38 e 41.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. [S.l.], 2008. Citado 7 vezes nas páginas 17, 37, 38, 43, 44, 45 e 63.

MCAULEY, J. et al. Image-based recommendations on styles and substitutes. 2015. Disponível em: <http://cseweb.ucsd.edu/~jmcauley/pdfs/sigir15.pdf>. Citado 4 vezes nas páginas 55, 58, 61 e 80.

MCCLENDON, C. M.; REGOT, L.; AKERS, G. *What is Prototyping?* 2012. Disponível em: <http://www.umsl.edu/~sauterv/analysis/prototyping/proto.html>. Citado na página 34.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning*. [S.l.], 2012. v. 9. Citado na página 40.

MONGODB. 2018. Disponível em: <https://www.mongodb.com/>. Citado na página 48.

MORALES-RAMIREZ, I.; PERINI, A.; GUIZZARDI, R. S. An ontology of online user feedback in software engineering. 2015. Disponível em: <https://www.researchgate.net/publication/289675883_An_ontology_of_online_user_feedback_in_software_engineering>. Citado 2 vezes nas páginas 26 e 35.

NATURAL Language Toolkit. 2018. Disponível em: <https://www.nltk.org/>. Citado na página 47.

NG, A. *CS229 Lecture notes*: The k-means clustering algorithm. 2017. Disponível em: <http://cs229.stanford.edu/notes/cs229-notes7a.pdf>. Citado 2 vezes nas páginas 15 e 44.

NUMPY. 2018. Disponível em: <http://www.numpy.org/>. Citado na página 47.

NUSEIBEH, B.; EASTERBROOK, S. requirements engineering: A roadmap. In: *ICSE '00 Proceedings of the Conference on The Future of Software Engineering*. [s.n.], 2000. p. 35–46. Disponível em: <https://dl.acm.org/citation.cfm?id=336523>. Citado na página 33.

OVERLEAF. 2018. Disponível em: <https://www.overleaf.com/>. Citado na página 49.

PAGANO, D.; BRUEGGE, B. User involvement in software evolution practice: A case study. In: *2013 35th International Conference on Software Engineering (ICSE)*. [s.n.], 2013. Disponível em: <http://ieeexplore.ieee.org/document/6606645/>. Citado na página 27.

PAGANO, D.; MAALEJ, W. User feedback in the appstore: An empirical study. In: *2013 21st IEEE International Requirements Engineering Conference (RE)*. [s.n.], 2013. Disponível em: <http://ieeexplore.ieee.org/document/6636712/>. Citado 2 vezes nas páginas 27 e 36.

PANDAS. 2018. Disponível em: <https://pandas.pydata.org/>. Citado na página 47.

PEDREGOSA, F. et al. *nearest-neighbors*. 2017. Disponível em: <http://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors>. Citado na página 42.

PIECH, C. *K Means*. 2013. Disponível em: <http://stanford.edu/~cpiech/cs221/handouts/kmeans.html>. Citado na página 43.

POHL, K.; RUPP, C. *Requirements Engineering Fundamentals*. [S.l.], 2011. v. 1. Citado na página 27.

PYTHON: Executive summary. 2018. Disponível em: <https://www.python.org/doc/essays/blurb/>. Citado na página 46.

PYTHON-EVE. 2018. Disponível em: <http://python-eve.org/>. Citado na página 47.

REACT Native. 2018. Disponível em: <https://facebook.github.io/react-native/>. Citado na página 69.

SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide*: The definitive guide to scrum: The rules of the game. [S.l.], 2017. Citado 3 vezes nas páginas 27, 54 e 66.

SINGHAL, A. Modern information retrieval: A brief overview. 2001. Disponível em: <https://periodicos.ufsc.br/index.php/emtese/article/viewFile/%2018027/16976>. Citado na página 43.

SKLEARN Kmeans. 2018. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>. Citado na página 80.

SKLEARN MultinomialNB. 2018. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html>. Citado 2 vezes nas páginas 76 e 77.

SKLEARN TfidfVectorizer. 2018. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>. Citado 4 vezes nas páginas 76, 77, 78 e 80.

SOMMERVILLE. *Software Engineering*. [S.l.], 2011. v. 9. Citado 7 vezes nas páginas 15, 25, 30, 31, 32, 66 e 67.

STOREY, M. et al. Stemming and lemmatization: A comparison of retrieval performances. 2014. Disponível em: <http://eprints.um.edu.my/13423/1/rp030_I3007.pdf>. Citado 2 vezes nas páginas 58 e 59.

THE STANDISH GROUP REPORT. Chaos report. 2015. Disponível em: <https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>. Citado na página 27.

VIRTUALENV. 2018. Disponível em: <https://pypi.org/project/virtualenv/>. Citado na página 46.

VIRTUALENVWRAPPER. 2018. Disponível em: <https://virtualenvwrapper.readthedocs.io/en/latest/>. Citado na página 47.

WELLS, D. *Extreme Programming*. 1996. Disponível em: <http://bu.ufsc.br/framerefer.html>. Citado na página 27.

WILLIAMS, G.; MAHMOUD, A. Mining twitter feeds for software user requirements. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. [s.n.], 2017. Disponível em: <http://ieeexplore.ieee.org/document/8048885/>. Citado 2 vezes nas páginas 26 e 87.

YAP, B. W. et al. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In: *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*. [s.n.], 2014. p. 13–22. Disponível em: <https://link.springer.com/chapter/10.1007/978-981-4585-18-7_2>. Citado na página 71.

ZHANG, H. The optimality of naive bayes. 2004. Disponível em: <http://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf>. Citado na página 41.