Classificação do Conteúdo Textual dos GitHub Wikis de Repositórios *Open Source*

Gabriel Moreira Chaves[1], **Ian Bittencourt Andrade**[2]

[1] Pontifícia Universidade Católica de Minas Gerais (PUCMG)
Caixa Postal 1.686 – 30.535-901 – Belo Horizonte – MG – Brazil

[2]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUCMG) – Belo Horizonte, MG – Brazil

{gabriel.chaves.1200613,ian.andrade}@sga.pucminas.br

Abstract. *The GitHub Wiki is used for building and defining the documentation and is one of the first impressions. However, there was still no work that characterized the GitHub Wiki textual content in an open source repository. In this way, this study will identify a purpose and a descriptive objective, from the categorization of GitHub, with the intention of facilitating users and owners of third-party entities, the discovery of relevant information, through a descriptive research, carr ied out through a study of case, accompanied by a research, seeking to promote an overview of the structures that are being addressed in the constructions of GitHub Wikis, in order to capture the presence of consistencies and applied to the structures.*

Resumo. *O GitHub Wiki é utilizado para a construção e definição da documentação e é uma das primeiras impressões em um repositório de software na plataforma GitHub. Todavia, ainda não existia um trabalho que caracterizasse o conteúdo textual de um GitHub Wiki em repositórios open source. Dessa forma, este estudo tem por objetivo identificar a finalidade e a relevância dos GitHub Wikis, a partir da categorização, com intenção de facilitar aos usuários e proprietários de repositórios a descoberta de informações relevantes, por uma pesquisa descritiva, realizada através de um estudo de caso, acompanhado de um survey, buscando promover uma visão geral sobre as estruturas que estão sendo abordadas nas construções dos GitHub Wikis, de modo a captar a presença de consistências e padrões aplicados às estruturas.*

Bacharelado em Engenharia de Software - PUC Minas
Trabalho de Conclusão de Curso (TCC)

Orientador de conteúdo (TCC I): Marco Rodrigo Costa - mrcosta@pucminas.br
Orientador acadêmico (TCC I): Laerte Xavier - laertexavier@pucminas.br
Orientador do TCC II: (A ser definido no próximo semestre)

Belo Horizonte, 22 de maio de 2022.

1. Introdução

A Gestão do Conhecimento é uma disciplina que visa reunir o capital intelectual das organizações [Rus and Lindvall 2002]. Através dela, é viável cumprir os objetivos organizacionais por meio das melhores práticas do uso do conhecimento. Na Engenharia

de Software, a Gestão do Conhecimento atinge o crescimento mais rápido das inovações por meio da acelerada evolução do conhecimento e a geração de um ambiente propício à utilização da informação já obtida [Jain 2011]. Apoiado nisso, pode-se identificar a redução da força de trabalho em novas atividades. Assim, alguns projetos *open source* localizados na plataforma GitHub utilizam arquivos README e os GitHub Wikis como formas de documentação dessas iniciativas, para que outras pessoas possam conhecer e colaborar com os projetos.

Os arquivos README executam um papel essencial na construção da primeira impressão de um desenvolvedor ao visitar um repositório de software e na documentação do projeto de software que o repositório hospeda [Prana et al. 2019]. No entanto, apesar dos inúmeros ganhos que a Gestão do Conhecimento traz em Engenharia de Software, com a utilização de ferramentas de documentação como o GitHub Wiki em projetos *open source*, não existem estudos que destacam o uso e o conteúdo de um GitHub Wiki, ou seja, qual é a finalidade, a relevância para a comunidade e como são estruturadas essas documentações. Portanto, o problema tratado neste estudo é **a ausência de trabalhos que classificam o conteúdo textual de um GitHub Wiki em repositórios *open source***.

Estudos relacionados ao tema abordaram a importância da plataforma GitHub para a hospedagem de código fonte e suas documentações, onde são utilizados por pesquisadores e usuários acadêmicos. Assim, os pesquisadores podem precisar do código para apoiar uma parte de seu trabalho acadêmico. Dessa forma, precisam entender completamente o algoritmo e todas as condições por trás do código para atingir seus objetivos de pesquisa. Já os usuários acadêmicos também podem ter a necessidade de usar alguns métodos específicos em determinadas aulas para criar sua própria função em um repositório [Jing 2019]. Nesse sentido, este estudo relaciona a Gestão do Conhecimento nos repositórios *open source* com o conteúdo presente em seus GitHub Wikis.

O objetivo geral deste estudo é **avaliar e classificar o conteúdo textual presente no GitHub Wiki em repositórios *open source***. Os objetivos específicos são: 1) propor modelos de classificação de conteúdo de GitHub Wikis; 2) identificar quais são as categorias mais recorrentes no conteúdo dos GitHub Wikis; 3) identificar quais modelos de aprendizado de máquina utilizados melhor realiza as classificações do conteúdo dos GitHub Wikis.

Com este estudo, espera-se obter uma classificação do conteúdo textual de um GitHub Wiki em projetos *open source* na plataforma GitHub através de modelos de aprendizagem de máquina, gerando métricas de desempenho, associadas à precisão, a revocação e ao F1 *score*. Assim, com a classificação, será possível identificar a finalidade e a relevância dos GitHub Wikis com intenção de facilitar aos usuários e proprietários de repositórios a descoberta de informações relevantes, buscando promover uma visão geral sobre as estruturas que estão sendo abordadas nas construções dos GitHub Wikis, de modo a captar a presença de consistências e padrões aplicados às estruturas.

O presente trabalho segue a seguinte estrutura: a Seção 2 apresenta os conceitos, teorias, mecanismos e processos consolidados e dão respaldo teórico/conceitual ao trabalho. Em seguida, a Seção 3 apresenta trabalhos que tratam do mesmo objeto de estudo, que estão no mesmo domínio ou tentam resolver problemas correlatos ao que foi proposto. Já a Seção 4 apresenta a categoria de pesquisa e sua justificativa, o ambiente

de desenvolvimento, métodos utilizados e suas justificativas, bem como as métricas de avaliação.

2. Fundamentação Teórica

Nesta seção são abordados os cinco principais conceitos deste estudo: Gestão do Conhecimento, Wikis, GitHub Wiki, Aprendizagem de Máquina e Métricas de Desempenho.

2.1. Gestão do Conhecimento

A Gestão do Conhecimento é uma disciplina que visa reunir o capital intelectual das organizações. O seu conceito surgiu na metade da década de 1980, a partir da demanda de extrair conhecimento do "dilúvio de informações" sendo empregado principalmente como um termo do "mundo dos negócios". Na década de 1990, muitas indústrias adotaram o conceito com tecnologias de computadores comerciais, facilitadas pelo desenvolvimento em áreas como a Internet, sistemas de suporte a grupos, mecanismos de busca, portais, *data warehouses* de conhecimento e a aplicação de análises estatísticas e técnicas de inteligência artificial [Rus and Lindvall 2002].

Nessa perspectiva, o conhecimento está embutido nas pessoas, no produto, no processo e na estrutura de uma organização. Ele pode ser captado a partir de sistemas organizacionais, processos, produtos, regras e cultura [Shankar et al. 2003]. Assim, o conhecimento pode ser classificado como explícito e tácito. O conhecimento explícito é estruturado e pode ser facilmente verbalizado, transmitido via linguagem formal e sistemática. Esse conhecimento pode ser armazenado, transportado e compartilhado por documentos e sistemas. Já o conhecimento tácito é altamente pessoal e difícil de ser formalizado, pois, está enraizado nas ações, experiências, ideias, valores ou mesmo nas emoções do sujeito.

Para garantir o uso e a aplicação da Gestão do Conhecimento, faz-se necessário a capitalização do conhecimento. Dessa maneira, a capitalização do conhecimento se encaixa como uma parte do processo de criação do conhecimento organizacional, tendo como objetivo construir informações amplas e aperfeiçoá-las por meio da disseminação. Portanto, um processo que irá garantir a Gestão do Conhecimento utiliza, explora e reutiliza o conhecimento, além de incluir aspectos técnicos e gerenciais, que vão desde atividades relacionadas à identificação do conhecimento até sua avaliação [Rosenthal-Sabroux and Grundstein 2005].

2.2. Wikis

Os wikis são coleções livremente expansível de páginas Web interligadas em um sistema de hipertexto para armazenar e modificar informações, onde cada página pode ser facilmente editada por qualquer usuário por um navegador Web [Leuf and Cunningham 2001]. São uma categoria de software de Gestão do Conhecimento que surgiu com o desenvolvimento da Web 2.0. Com essa categoria de software, pode-se realizar a colaboração através de comunicação e compartilhamento de conhecimento entre usuários [Chen et al. 2018].

Os wikis são geralmente uma aplicação executada em um ou mais servidores Web, podendo ser executado em um ambiente de computação distribuída e descentralizada. O conteúdo, incluindo revisões anteriores, é geralmente armazenado em um sistema de arquivos ou em um banco de dados. Fundamentalmente existem três tipos em relação ao seu

uso: 1) wikis públicos, destinados a serem lidos por qualquer pessoa, porém geralmente nem sempre podem ser editados por todos; 2) wikis organizacionais, destinados a serem utilizado em um ambiente privado, particularmente para a aprimorar o compartilhamento de conhecimento interno; 3) wikis pessoais, destinados a serem usados por uma única pessoa para gerenciar notas pessoais.

2.3. GitHub Wiki

O GitHub Wiki é uma opção para documentar um repositório no GitHub. As páginas no GitHub Wiki são armazenadas em repositórios Git. As informações no README podem ser expandidas adicionando páginas no GitHub Wiki ao repositório do projeto [Bleiel 2016]. Cada repositório no GitHub permite adicionar um GitHub Wiki e as categorias de uso dos GitHub Wikis podem ser qualquer um dos três citados na Subseção 2.2. A plataforma permite hospedá-los em espaços pessoais e organizacionais de forma pública ou privada, possibilitando que cada usuário decida o modo de uso adotado.

As páginas do GitHub Wiki pode ser editadas no GitHub ou localmente. Por padrão, apenas usuários com acesso de gravação ao repositório podem realizar alterações nessas páginas, entretanto há uma opção para poder permitir que todos os usuários que tenham acesso de leitura ao repositório consigam editar as páginas do GitHub Wiki. Cada colaborador pode criar *branchs* ao trabalhar em repositórios de GitHub Wiki, mas somente as alterações enviadas para a *branch* principal são disponibilizadas os leitores.

2.4. Aprendizagem de Máquina

O aprendizado de máquina (ML, do inglês *Machine Learning*) é um dos campos pertencentes à computação moderna. Diversos estudos foram realizados para tornar as máquinas inteligentes. Porém, a aprendizagem, um comportamento humano natural, tem se tornado um aspecto essencial das máquinas desde a década de 1950, quando ocorreu o domínio do aprendizado de máquina [Shinde and Shah 2018]. ML pode ser classificada em duas formas: 1) supervisionada; 2) não supervisionada. Uma aprendizagem é considerada supervisionada somente se os dados utilizados estão antecipadamente classificados, qualificados ou categorizados. De outro modo, a aprendizagem de máquina é tida como não supervisionada. No presente estudo, o conceito e a aplicação da aprendizagem de máquina será referente a aprendizagem supervisionada.

2.4.1. Algoritmos de Aprendizagem de Máquina

Neste estudo, são utilizados cinco algoritmos de ML supervisionada, sendo: 1) Máquina de vetores de suporte (SVM, do inglês *Support Vector Machines*); 2) Regressão logística (LR, do inglês *Logistic Regression*); 3) Floresta aleatória (RF, do inglês *Random Forest*); 4) k-ésimo vizinho mais próximo (k-NN, do inglês *k-Nearest Neighbors*); 5) Multinomial Naive Bayes (MNB, do inglês Multinomial Naive Bayes).

1. **SVM:** é um algoritmo de ML supervisionado que geralmente é utilizado para tarefas de classificação ou regressão. Seu objetivo principal é treinar e classificar, baseado em um *dataset*. No algoritmo, cada dados do item é plotado como um ponto no espaço n-dimensional (onde n é o número de recursos obtidos) com o valor de cada recurso sendo o valor de uma determinada coordenada. Em seguida,

a classificação é realizada, encontrando o hiperplano que diferencia muito bem às duas classes;

2. **LR:** é um algoritmo que pode ser usado para modelar a probabilidade de uma determinada classe ou evento. É usado quando os dados são linearmente separáveis e o resultado é de natureza binária ou dicotômica;
3. **RF:** é um algoritmo de ML utilizado em problemas de classificação e regressão. Ele constrói árvores de decisão em diferentes amostras e resulta a maioria das avaliações para classificação, em caso de regressão a média é resultante. Uma das características mais pertinentes do algoritmo RF é que ele pode trabalhar com um conjunto de dados contendo variáveis contínuas como no caso de regressão e variáveis categóricas como no caso de classificação;
4. **k-NN:** o algoritmo realiza previsões calculando a similaridade entre a amostra de entrada e cada instância de treinamento. No algoritmo k-NN, a saída é uma associação de classe. A priori, geralmente k é um número inteiro positivo. O algoritmo fundamentalmente se resume a formar um parecer principal entre as k instâncias como uma dada observação "invisível". A similaridade é definida de acordo com uma métrica de distância entre dois pontos de dados;
5. **MNB:** é um algoritmo baseado no Teorema de Bayes com uma suposição de independência entre os preditores. Simplificando, um classificador Naive Bayes assume que a presença de uma característica particular em uma classe não está relacionada à presença de qualquer outra característica. Por exemplo, uma fruta pode ser considerada uma banana se for amarela, côncava e com peso médio de 90 gramas. Mesmo que essas características pendam umas das outras ou da existência de diferentes características, todas essas particularidades contribuem independentemente para haver a probabilidade de que esta fruta seja uma banana por isso é conhecida como "Naive".

2.5. Métricas de Desempenho

As métricas adotadas pelo presente trabalho são utilizadas sobretudo para avaliar o desempenho dos modelos de ML utilizados para as classificações. Dessa forma, o desempenho é medido por equações matemáticas que equiparam os resultados. Portanto, as métricas avaliadas são: 1) precisão; 2) revocação (do inglês, *recall*); 3) F1 *score*. De modo a gerar uma melhor compreensão das métricas supracitadas, considera-se a Tabela 1 para demonstrar a assertividade dos modelos de ML. Assim, a definição de Verdadeiro Positivo, Falso Negativo, Falso Positivo e Verdadeiro Negativo são geradas a partir das classificações realizadas pelos modelos.

Tabela 1. Matriz de confusão

Atribuído			
Real		**Positivo**	**Negativo**
	Positivo	Verdadeiro Positivo (VP)	Falso Negativo (VN)
	Negativo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

2.5.1. Equação de Precisão

A pontuação de precisão do modelo mede a proporção de categorias positivamente previstas que estão corretas. A precisão é usada em conjunto com o *recall* para compensar falsos positivos e falsos negativos. Como demonstrado pela Equação 1, o cálculo para obter a precisão é a razão entre o número de Verdadeiros Positivos (VP) pelo número de Verdadeiros Positivos (VP) mais o número de Falsos Positivos (FP).

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{1}$$

2.5.2. Equação de Revocação

A métrica de revocação ou *recall* representa a capacidade de o modelo de prever corretamente as previsões positivas de todas as previsões positivas que poderiam ter sido feitas. Dessa forma, representado pela Equação 2, a revocação é o número de previsões positivas (VP) pela razão da soma entre Verdadeiros Positivos (VP) e Falsos Negativos (FN).

$$\text{Revocação} = \frac{VP}{VP + FN} \tag{2}$$

2.5.3. Equação de F1 *score*

O F1 *score* é uma métrica de desempenho do modelo de ML que dá peso igual tanto à precisão quanto à revocação. Assim, representada pela Equação 3, o F1 *score* é calculado a partir da multiplicação entre os valores de precisão e *recall*, vezes dois, dividido pela soma entre o valor de precisão e o valor de *recall*.

$$\text{F1 } \textit{score} = \frac{2 * (\text{Precisão} * \text{Revocação})}{\text{Precisão} + \text{Revocação}} \tag{3}$$

3. Trabalhos Relacionados

Os trabalhos relacionados discutidos nesta seção envolvem classificações de conteúdo de documentações, entre outros meios de compartilhamento de conhecimento. Desse modo, os trabalhos estão ordenados conforme o seu nível de aproximação com o que este trabalho se propõe a resolver.

Vista a importância dos arquivos README para a construção e definição da documentação e da primeira impressão em um repositório de software na plataforma GitHub, ainda não existia um entendimento sistemático do conteúdo de um arquivo README [Prana et al. 2019]. Para isto, foi realizado um estudo qualitativo envolvendo seções de arquivos README de 393 repositórios. Além disso, foram projetados e avaliados um classificador e um conjunto de recursos para categorizar essas seções automaticamente. Neste classificador, são previstas oito categorias diferentes, atingindo um valor de F1 *score* de 0,746. Para avaliar a utilidade da classificação, as classes foram apresentadas para profissionais validarem. Como resultado, a maioria dos participantes notou

a classificação automatizada. Diante disto, o presente trabalho se baseia na metodologia de classificação para a definição do conteúdo, adotando às oito categorias propostas nas classificações e realização da validação a partir do F1 *score*, entretanto utilizando a aplicação desses conceitos em GitHub Wikis.

As notas de lançamento podem ser documentadas de diferentes maneiras com base em vários projetos, pois não há padronização aceita para documentar notas de lançamento [Bi et al. 2020]. Assim, é proposta uma compreensão das características de notas de lançamentos e como uma nota pode se adequar a cada público. Para realizar a pesquisa, foram selecionados projetos *open source* do GitHub. Antes de realizar o estudo, foram enviados *emails* para vários desenvolvedores de diferentes projetos do GitHub para discutir o processo de lançamento e produção de notas de lançamento. Logo depois, através da metodologia Meta/Pergunta/Métrica (GQM, do inglês *Goal/Question/Metric*) foram analisadas e caracterizadas as notas de lançamento. Portanto, o estudo supracitado se torna um ponto de referência para este presente trabalho, pois o processo de ML usado para prever os problemas que devem ser listados nas notas de versão são considerados quando executado o treinamento dos modelos de ML usados para a categorização do conteúdo textual de GitHub Wikis.

O conhecimento está dividido em duas formas, descritos como externalização ou combinação, pelo modelo SECI. Plataformas como o GitHub revolucionaram como os desenvolvedores trabalham, introduzindo novos canais para compartilhar conhecimento na forma de *pull requests*, *issues* e *wikis*. Não está claro como esses canais capturam e compartilham conhecimento [Tantisuwankul et al. 2019]. Dessa forma, o objetivo está em analisar esses canais de comunicação no GitHub. Primeiro, usando o modelo SECI. Em seguida, uma análise topológica, extraindo *insights* dos diferentes canais de comunicação do GitHub. Usando duas questões de pesquisa, foram exploradas a evolução dos canais e adoção de canais por projetos populares e impopulares. Por fim, os resultados mostram que 1) projetos atuais tendem a adotar múltiplos canais de comunicação; 2) os canais de comunicação mudam temporalmente; 3) os canais de comunicação são usados para capturar novos conhecimentos e atualizar o conhecimento existente. Portanto, o processo de classificação utilizado no trabalho relacionado traz uma base para a presente pesquisa, de modo a colaborar na identificação de quais modelos de aprendizado de máquina utilizados melhor realiza as classificações do conteúdo dos GitHub Wikis.

A maioria dos desenvolvimentos de aplicativos ocorre no contexto de Interface de Programação de Aplicações (APIs, do inglês *Application Programming Interfaces*) complexas [Kumar and Devanbu 2016]. Assim, a documentação de referência para APIs cresceu expressivamente em variedade, complexidade e volume, o que pode dificultar a navegação e pesquisa. Há uma necessidade crescente de desenvolver formas bem definidas de acesso ao conhecimento nas documentações. Seguindo esta linha, vários esforços de pesquisa lidam com a organização (ontologia) do conhecimento relacionado a APIs. O trabalho de engenharia do conhecimento, apoiado por uma análise qualitativa, por Maalej, W. and Robillard, M. P. (2013) identificou uma taxonomia útil do conhecimento de API. Com base nessa taxonomia, foi apresentada uma técnica domínio para extrair as categorias de conhecimento da documentação de referência de APIs. À vista disso, o sistema OntoCat apresenta nove recursos, combinações semânticas e estatísticas que classificam as diferentes categorias de conhecimento. A partir do que foi apresentado acima, o pre-

sente trabalho classifica o conteúdo dos GitHub Wikis baseando-se nas abordagens de aprendizado de máquina supervisionado.

Os desenvolvedores de software necessitam de acesso a diferentes categorias de informações que estão geralmente dispersas entre diferentes formas de documentação, como documentação de APIs ou no Stack Overflow [Treude and Robillard 2016]. A partir disso, o trabalho relacionado apresenta uma abordagem para aumentar automaticamente a documentação das APIs por *insights* do Stack Overflow, ou seja, identificando categorias. Com base em um conjunto de frases, foi comparado o desempenho de duas técnicas de classificação, bem como uma abordagem baseada em padrões para extração de frases. Em seguida, o SISE, abordagem baseada em aprendizado de máquina que usa como característica a análise das frases foi apresentada. Com base nisso, este presente trabalho irá basear-se nos mecanismos utilizados na construção do sistema SISE para construir um processo que realizará a classificação do conteúdo textual dos GitHub Wikis selecionados.

4. Materiais e Métodos

O presente trabalho se trata de um **estudo de caso** acompanhado de um ***survey***, caracterizando-se como uma **pesquisa descritiva**. O estudo de caso é realizado com a intenção de coletar **dados qualitativos** a partir do conteúdo dos GitHub Wikis. Após a coleta, pesquisadores e colaboradores classificam repositórios de GitHub Wiki a partir de oito categorias predeterminadas, apontadas na Tabela 2. Como resultado, se a maioria dos participantes classificarem o conteúdo como um conteúdo que se enquadra em determinada categoria, este conteúdo é classificado nessa categoria. As classificaçõe coletadas pelo *survey* e as classificações realizadas pelos pesquisadores são utilizadas para compor o conjunto de dados de treinamento e teste dos modelos de ML.

O uso da estratégia de *survey*, consiste no envio de mensagens por email para cada colaborador que realizou contribuições nos repositórios de GitHub Wikis utilizados na pesquisa apresentado na Figura 1, contendo um formulário para ser classificado o conteúdo apresentado na Figura 2. Para a ratificação das categorias, utiliza-se a análise estatística da classificação binária por meio da métrica F1 *score*, baseando-se na classificação multi-rótulo, apresentada pelos autores [Prana et al. 2019].

Tabela 2. Categorias de classificação do conteúdo dos GitHub Wikis

#	Categorias	Exemplos
1	*What*	Introdução, contexto do projeto
2	*Why*	Vantagens do projeto, comparação com trabalhos relacionados
3	*How*	Introdução ou início rápido, como executar, instalação, como atualizar, configuração, configuração, requisitos, dependências, idiomas, plataformas, demonstração, downloads, erros e bugs
4	*When*	Status do projeto, versões, planos de projeto, roteiro
5	*Who*	Equipe do projeto, comunidade, proprietário, lista de discussão, contato, reconhecimento, licença, código de conduta
6	*References*	Documentação da API, obtenção de suporte, feedback, mais informações, traduções, projetos relacionados
7	*Contribution*	Diretrizes de contribuição
8	*Other*	

Figura 1. Email enviado para os colaboradores

I figured out that you contributed to the GitHub Wiki from [repository name].

I kindly ask you to contribute to the classification of the content of this Wiki, answering the form: [form link].

Figura 2. Formulário enviado para os colaboradores

Classification of GitHub Wiki from [repository name]

Based on what you see in the textual content on the GitHub Wiki, how do you classify it.

Each question has a category and description with examples of text snippets related to certain categories.

For each question, if you consider the content on the GitHub Wiki to have a certain category, please respond with snippets of text that you classified as a certain category.

For each snippets of text, write on a separate line. If you feel that a certain category is not present, leave the answer blank.

What: Introduction, project context.
Why: Project advantages, comparison with related works.
How: Getting started or Quick start, how to run, installation, how to update, setup, configuration, requirements, dependencies, languages, platforms, demo, downloads, errors and bugs.
When: Project status, versions, project plans, roadmap.
Who: Project team, community, owner, mailing list, contact, acknowledgment, license, code of conduct.
References: API documentation, getting support, feedback, more information, translations, related projects.
Contribution: Contribution Guidelines.
Other:

4.1. Coleta de Dados

Coleta-se uma amostra contendo 500 GitHub Wikis dos repositórios mais populares da plataforma Github (considerando como mais populares os repositórios com o maior número de estrelas), por meio da GitHub API GraphQL v4. Essa API permite realizar a busca pelos repositórios mais populares da plataforma e retornar como resultado apenas as informações necessárias para o estudo. Dessa forma, é permitido verificar se o repositório possui o GitHub Wiki habilitado. Apenas repositórios com o campo "hasWikiEnabled" com o valor positivo são utilizados.

Além disso, é verificado o idioma do GitHub Wiki e apenas Wikis em inglês serão utilizados. Outrossim, somente repositórios que contém como linguagem principal uma linguagem de programação são utilizados. Por fim, possuir o GitHub Wiki habilitado não garante que o repositório do GitHub Wiki exista e possua alguma página. Sendo assim, a possibilidade de clonar o repositório é verificada e se este possui algum conteúdo para ser classificado.

De modo a identificar se o idioma do GitHub Wiki é escrito em inglês, utiliza-se uma biblioteca em Python com uma base dados treinados para detectar idiomas a partir de um algoritmo de Naive Bayes com um N-grama de caracteres. Já para identificar se a linguagem principal retornada pela GitHub API GraphQL v4 é uma linguagem de programação, utiliza-se o arquivo "languages.yml" do repositório "github/linguist" na versão v7.20.0. O arquivo possui uma estrutura de mapa, onde suas chaves são os nomes das linguagens retornadas pela API e cada valor desse mapa possui o campo "type" que permite identificar se é uma linguagem de programação se o seu valor for "programming".

4.2. Ferramentas

A linguagem de programação Python, em sua versão 3.7 foi escolhida. Essa escolha se dá por Python ser uma linguagem de programação amplamente utilizada no meio científico, visto que contribui para a realização de tarefas que necessitam resolver problemas como processamento e manipulação de dados e aprendizado de máquina.

Foram utilizadas as seguintes bibliotecas para a linguagem de programação Python:

1. **Langdetect:** uma biblioteca de identificação de idiomas para a linguagem de programação Python. Utilizada para identificar se o idioma do conteúdo das páginas do GitHub Wiki foram escritos em inglês;
2. **PyYAML:** uma biblioteca de leitura e escrita de arquivos YAML para a linguagem de programação Python. Utilizada para ler o conteúdo do arquivo que contem as linguagens identificadas pelo GitHub;
3. **Asyncio:** uma biblioteca para escrever código concorrente na linguagem de programação Python. O tempo da coleta de dados a partir de requisições Web é determinado pelo tempo gasto aguardando a conclusão dessas operações. Devido a isso, a coleta dos dados realizada nesse trabalho ocorre concorrentemente;
4. **Aiohttp:** uma biblioteca de cliente HTTP assíncrona para a linguagem de programação Python que utiliza soquetes não bloqueantes, projetada para interoperar com a biblioteca Asyncio. Utilizada para realizar as requisições para a GitHub GraphQL API v4;
5. **Git:** utilizado para realizar a clonagem dos repositórios e extração dos emails dos contribuidores;
6. **NumPy:** uma biblioteca de computação científica para a linguagem de programação Python orientada para trabalhar eficientemente com *arrays*. Utilizada para tratar os dados coletados;
7. **Pandas:** uma biblioteca de manipulação e análise de dados para a linguagem de programação Python que permite a análise e manipulação de dados tabulares em *Dataframes*;
8. **Scikit-learn:** uma biblioteca de aprendizado de máquina para a linguagem de programação Python projetada para interoperar com a biblioteca NumPy. Os algoritmos de classificação assim como os relatórios das métricas utilizadas são procedentes dessa biblioteca.

A ferramenta Google Colab foi utilizada, pois a mesma permite escrever e executar Python no navegador com nenhuma configuração necessária, acesso gratuito a GPUs e compartilhamento fácil. Utiliza-se um ambiente de execução hospedado na Web, ofertado gratuitamente.

4.3. Treinamento e avaliação dos modelos

Os algoritmos de aprendizado de máquina utilizados são: SVM; LR; RF; k-NN; MNB. Exemplos de categorias de classificação do conteúdo dos GitHub Wikis são utilizados para realizar o treinamento dos modelos de aprendizado de máquina. Dessa forma, para ser possível realizar a avaliação dos modelos, métricas de *recall* e precisão são coletadas para a realização do cálculo do F1 *score*. Com o treinamento, são resultantes modelos que conseguem identificar quais categorias o GitHub Wiki avaliado se enquadra. Portanto, pode ser possível utilizar estes modelos para classificar qualquer GitHub Wiki escrito no idioma inglês.

Os modelos são avaliados por meio da métrica F1 *score*. Cada categoria é avaliada individualmente, ou seja, para cada categoria é avaliado se o modelo realizou corretamente a classificação do conteúdo do GitHub Wiki. A classificação é considerada correta se ocorrer um "verdadeiro positivo" ou um "verdadeiro negativo". Após gerar os oito valores de F1 *score*, uma média é realizada de modo a identificar quais modelos de aprendizado de máquina utilizados melhor realiza as classificações do conteúdo dos GitHub Wikis.

4.4. Cronograma

O desenvolvimento do trabalho é planejado seguindo uma abordagem de tarefas realizadas advindas de um cronograma delimitado por períodos quinzenais como é apresentado na Tabela 3.

Tabela 3. Quinzenas para desenvolvimento do trabalho. G: Tarefas atribuídas ao Gabriel; I: Tarefas atribuídas ao Ian

	2022						
	agosto		**setembro**		**outubro**		**novembro**
Etapas	**1ª Q**	**2ª Q**	**1ª Q**	**2ª Q**	**1ª Q**	**2ª Q**	**1° Q**
Extração dos dados da API do GitHub	G						
Extração dos dados dos repositórios		G					
Algoritmo do envio da *survey*	I						
Algoritmo da coleta da *survey*		I					
Classificação do conteúdo dos GitHub Wikis			G / I				
Análise e manipulação dos resultados das classificações				I			
Treinamento dos modelos de aprendizado de máquina				G	G / I	G / I	
Discussão e avaliação dos resultados							G / I

Referências

Bi, T., Xia, X., Lo, D., Grundy, J., and Zimmermann, T. (2020). An empirical study of release note production and usage in practice. *IEEE Transactions on Software Engineering*.

Bleiel, N. (2016). Collaborating in github. In *2016 IEEE International Professional Communication Conference (IPCC)*, pages 1–3.

Chen, P., Song, J., Zhao, M., and Song, L. (2018). Building discipline knowledge repository based on wiki technology. In *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pages 407–410.

Jain, R. (2011). Improvement in software development process and software product through knowledge management. *International Journal of Computer Technology and Applications*, 2(5):1557–1562.

Jing, W. (2019). Open science: Github site publication of hotspot algorithms. *University of Minnesota Digital Conservancy*.

Kumar, N. and Devanbu, P. (2016). Ontocat: Automatically categorizing knowledge in api documentation. *arXiv preprint arXiv:1607.07602*.

Leuf, B. and Cunningham, W. (2001). *The wiki way: Quick collaboration on the Web*. Addison-Wesley Professional, Boston.

Maalej, W. and Robillard, M. P. (2013). Patterns of knowledge in api reference documentation. *IEEE Transactions on Software Engineering*, 39(9):1264–1282.

Prana, G. A. A., Treude, C., Thung, F., Atapattu, T., and Lo, D. (2019). Categorizing the content of github readme files. *Empirical Software Engineering*, 24(3):1296–1327.

Rosenthal-Sabroux, C. and Grundstein, M. (2005). A process modeling approach to identify and locate potential crucial knowledge: The gameth framework. *IRMA05*.

Rus, I. and Lindvall, M. (2002). Knowledge management in software engineering. *IEEE Software*, 19(3):26–38.

Shankar, R., Gupta, A., Narain, R., et al. (2003). Strategic planning for knowledge management implementation in engineering firms. *Work study*.

Shinde, P. P. and Shah, S. (2018). A review of machine learning and deep learning applications. In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pages 1–6. IEEE.

Tantisuwankul, J., Nugroho, Y. S., Kula, R. G., Hata, H., Rungsawang, A., Leelaprute, P., and Matsumoto, K. (2019). A topological analysis of communication channels for knowledge sharing in contemporary github projects. *Journal of Systems and Software*, 158:110416.

Treude, C. and Robillard, M. P. (2016). Augmenting api documentation with insights from stack overflow. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 392–403.