Desenvolvimento de framework para aquisição, tratamento e análise de biosinais em python

Development of a framework for the acquisition, processing, and analysis of biosignals in Python

Pedro Henrique Germano Silva[1], Arthur Hauer[2], José Jair Alves Mendes Júnior[3]

RESUMO

O avanço das tecnologias de coleta e análise de dados tem impulsionado significativamente o campo da bioengenharia nos últimos anos, destacando a importância do processamento de biosinais, como ECG, EEG e EMG, na pesquisa biomédica. Para lidar com a crescente complexidade desses dados, o desenvolvimento de um framework eficiente é essencial, simplificando a transformação de dados complexos em informações úteis e acessíveis. Este artigo enfatiza a relevância crítica de tal framework e destaca a importância da documentação abrangente para tornar a ferramenta acessível a um público amplo, acelerando assim os avanços na bioengenharia. A metodologia apresenta o uso do framework Open_BCI_Framework, com ênfase na configuração do arquivo "configuration.json". A documentação automatizada desempenha um papel fundamental, combinando docstrings e a biblioteca Sphinx do Python para criar documentação de alta qualidade em vários formatos. Em resumo, a pesquisa destaca a necessidade de frameworks eficientes e documentação adequada para promover a análise de biosinais, proporcionando avanços na bioengenharia.
PALAVRAS-CHAVE: Biosinais, Documentação, Open_BCI_Framework.

ABSTRACT

The advancement of data collection and analysis technologies has significantly driven the field of bioengineering in recent years, highlighting the importance of processing biosignals such as ECG, EEG, and EMG in biomedical research. To address the growing complexity of this data, the development of an efficient framework is essential, simplifying the transformation of complex data into useful and accessible information. This article emphasizes the critical relevance of such a framework and underscores the importance of comprehensive documentation to make the tool accessible to a broad audience, thus accelerating advancements in bioengineering. The methodology presents the use of the Open_BCI_Framework, with a focus on configuring the "configuration.json" file. Automated documentation plays a pivotal role, combining docstrings and the Python Sphinx library to create high-quality documentation in various formats. In summary, the research highlights the need for efficient frameworks and proper documentation to advance biosignal analysis, contributing to progress in bioengineering.
KEYWORDS: Biosignals, Documentation, Open_BCI_Framework

INTRODUÇÃO

Nos últimos anos, o campo da bioengenharia está sendo impulsionado pelo avanço das tecnologias de coleta e análise de dados. Um dos setores que se beneficiam desse progresso é o campo dos biosinais, abrangendo vários sinais biológicos, como eletrocardiogramas (ECG), eletroencefalogramas (EEG) e eletromiogramas (MENDES

[1] Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, Brasil E-mail: pedrosilva.2001@alunos.utfpr.edu.br. ID Lattes: 8065700475984265.
[2] Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, Brasil E-mail: arthur.hauer@hotmail.com. ID Lattes: 4166177649192618.
[3] Docente no DAELN. Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, Brasil E-mail: jjjunior@utfpr.edu.br. ID Lattes: 1920188611669631.

JUNIOR et al., 2020). Esses biosinais fornecem informações sobre o funcionamento biológico e desempenham um papel fundamental no diagnóstico, tratamento e monitoramento de condições médicas.

À medida que a variedade de biosinais e a quantidade de dados aumentam, os desafios relacionados à interpretação se tornam mais complexos. Nesse contexto, desenvolver uma ferramenta (*framework*) eficiente para coletar, tratar e analisar biosinais é uma realização de grande importância. Este framework é particularmente valioso para pesquisadores de processamento de sinais, pois simplifica o processo de transformar dados complexos em informações relevantes, tornando-o mais rápido e acessível.

Nesse cenário, o principal objetivo é destacar a relevância do desenvolvimento de um *framework* destinado à coleta, tratamento e análise de biosinais. Além disso, enfatiza-se a importância do desenvolvimento de uma documentação abrangente desse *framework*, assegurando que seja acessível a um público mais amplo. Essa documentação não apenas torna a ferramenta utilizável por um número maior de pessoas, mas também simplifica o processo de análise de dados, tornando-o mais rápido e acessível para pesquisadores e profissionais da bioengenharia. Este trabalho faz parte das pesquisas desenvolvidas durante o mestrado do aluno Arthur Hauer, do Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI), sendo uma das etapas do desenvolvimento do seu trabalho de conclusão.

METODOLOGIA

Nesta seção, descreve-se primeiro as etapas a serem seguidas para a utilização do framework Open_BCI_Framework, desenvolvido por Arthur Hauer (HAUER, 2023). Em seguida, são apresentadas as metodologias e ferramentas utilizadas para a criação automática da documentação.

UTILIZAÇÃO DO FRAMEWORK

Para a utilização do framework Open_BCI_Framework, é necessário entender o arquivo "configuration.json" que fica na pasta "config" do projeto. É nesse arquivo que é definida toda a sequência de processamento dos dados desejados, seja coleta, tratamento ou análise dos biosinais. Essa sequência de processamento dos dados é totalmente personalizável, ou seja, se já se possui os dados e é necessário apenas analisá-los, basta apenas configurar o arquivo "configuration.json"; ou ainda, se é desejado apenas coletar e salvar os dados, também é possível fazê-lo.

Este arquivo de configuração organiza os elementos do *framework* em uma estrutura de nós, em que cada nó representa uma manipulação/processamento que é desejada ser feita com os dados. Por exemplo, um *dataset* de EEG em csv precisa ser segmentado em partes menores, de modo a permitir classificar cada parte menor em um tipo de movimento motor realizado pelo indivíduo cujos dados de EEG foram coletados. Para fazer isso com o Open_BCI_Framework, o arquivo de configuração terá quatro nós, sendo o primeiro um nó que irá ler o arquivo csv, o segundo que irá segmentar os dados lidos pelo nó anterior, o terceiro que irá classificar os dados segmentados e um quarto que irá salvar o resultado da segmentação. Então para utilizar o *framework* é necessário saber quais são as etapas necessárias para o processamento dos dados, e então cada etapa será representada por um nó que será adicionado ao arquivo "configuration.json".

Com relação a estruturação do arquivo "configuration.json", é necessário compreender a estrutura básica de um JSON (Notação de Objetos JavaScript, do inglês JavaScript Object Notation), que consiste de uma coleção de pares nome/valor, dentro de chaves. O primeiro par nome/valor do JSON é o "nodes", que irá conter como valor um objeto, e ele representa justamente os nós de processamento explicados anteriormente separados em dois grandes grupos: o "root", o qual representa todas as entradas de dados, seja a leitura de um arquivo csv, seja um coleta em tempo real de um biossinal; o "common", que representa todos os processamentos que será feito com os dados, bem como a parte de persistência (salvamento) dos dados. A Figura 1 mostra a estrutura básica descrita anteriormente.

Figura 1 – Estrutura básica do "configuration.json"

Fonte: Autoria própria

O par nome/valor "root" é um objeto composto de vários nós de entrada de dados. É possível existir mais de um nó dentro deste objeto, sendo cada nó uma fonte de dados diferente, sejam eles relacionados ou não. Por exemplo, em um arquivo csv tem-se os coletados de biossinais de EMG, e em outro arquivos tem os relativos eventos (trials) dos sinais coletados (piscar, levantar a mão etc); então, para utilizar esses dados no *framework* é necessário ter dois nós de entrada de dados.

O par nome/valor "common" é um objeto composto por vários nós de processamento e persistência de dados. Caso seja necessário várias etapas de processamento, pode-se adicionar vários nós dentro da estrutura "common", na qual é possível configurar que o resultado do processamento de um nó será o dado de entrada de um outro nó de processamento ou armazenamento.

Por fim, dentro de "root" e "common", tem-se de fato os nós de processamento. Cada nó é um objeto com o nome qualquer definido pelo usuário, que possui um tipo que é representado por modelos (*models*). O modelo é um tipo de processamento a ser realizado com os dados em um determinado nó. Cada nó possui suas configurações específicas e podem ser encontradas na documentação. Então, para adicionar uma etapa de processamento basta alocar um nó (objeto) dentro de "root" ou "common" de um modelo que seja adequado para aquilo que se deseja processar. Algumas configurações que são comuns para todos os nós são: "module", que é o módulo o qual se encontra o modelo; "type" (tipo), o módulo em si; "outputs", que é um objeto com arrays dentro, em que cada item do array aponta para onde vão os resultados do processamento desse nó (os resultados podem ir para mais de um nó). Ainda sobre "outputs", cada array dentro de outputs representa um tipo de saída do nó, e cada item do array representa o nó que receberá os dados. Cada item do array é um objeto com os seguintes parâmetros: "node",

o nome do nó para onde os dados serão mandados; "input", que não influencia no funcionamento do framework.

Resumindo a utilização do framework, basta definir a sequência de processamento desejada, selecionar os modelos que realizam os processamentos definidos, inserir os nós de entrada dentro de "root" e os demais nós em "common", configurar cada nó conforme as configurações de cada modelo que representa o nó, e definir as saídas (*outputs*) de cada nó conforme o que foi estipulado na sequência de processamento.

DOCUMENTAÇÃO

Ao criar um *framework*, a documentação é especialmente importante. Os *frameworks* são estruturas complexas que fornecem funcionalidades extensas e, muitas vezes, exigem uma curva de aprendizado. Sem documentação adequada, os desenvolvedores podem demorar para a compreensão do funcionamento do framework, o que prejudica sua utilidade e adoção. Neste contexto, foi decidido a utilização de ferramentas para a documentação automática, na qual durante o próprio processo de programação do framework, através de comentários, é possível gerar uma documentação. Essas ferramentas são as docstrings e a biblioteca Sphinx (Desenvolvedores Sphinx, 2023), ambas ferramentas do ecossistema Python.

Uma docstring é uma string de documentação que aparece no início de uma função, método, classe ou módulo Python. Ela é usada para explicar o que a função faz, quais são seus parâmetros e o que ela retorna. As docstrings são cercadas por três aspas, simples ou duplas, e são uma forma eficaz de fornecer informações claras e concisas sobre o código.

Para automatizar a geração de documentação a partir de docstrings, a biblioteca Sphinx é uma ferramenta do ecossistema Python. O Sphinx permite criar documentações de alta qualidade em vários formatos. Para este projeto decidiu-se criar um site estático HTML, uma vez que permite que a documentação seja aberta em um navegador genérico pelo usuário e sua disponibilização pode ser *online*.

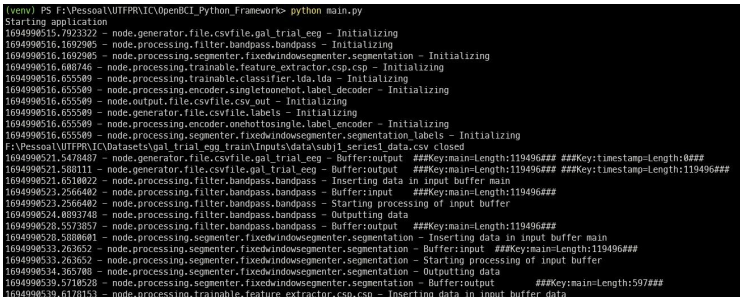
A utilização desta biblioteca se fez da seguinte forma:

1. Instalação do Sphinx usando o pip (instalador de bibliotecas Python), com o seguinte comando: pip install sphinx
2. Criação de um projeto Sphinx, com o seguinte comando: sphinx-quickstart docs. Isso irá criar um diretório "docs" com todas as configurações para gerar a documentação corretamente.
3. Criado o projeto Sphinx, no arquivo de configuração "conf.py", no qual foram definidas as configurações da documentação conforme desejado.
4. Escrever as docstrings. Neste projeto foram utilizadas as recomendações estilísticas do próprio Sphinx (VLADIMIROV, 2023).
5. Por fim, foi gerada a documentação utilizando os seguintes comandos no terminal:
    a. sphinx-apidoc -o docs .
    b. make html

RESULTADOS E DISCUSSÃO

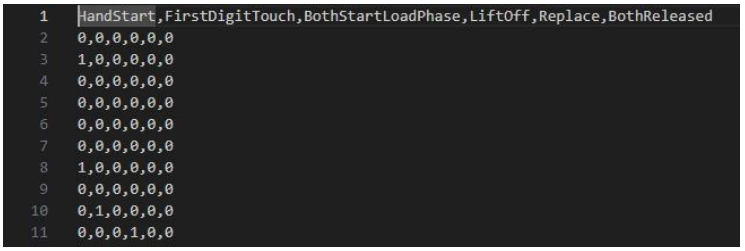
Seguindo aquilo que foi descrito sobre a utilização do framework na metodologia, e executando a aplicação no PowerShell do Windows, pode-se observar na Figura 2 os logs do framework em execução e na Figura 3 o resultado de uma classificação de biosinais de EEG. A Figura 4 mostra como ficou o resultado gerado utilizando a metodologia descrita anteriormente.

Figura 2 – Documentação do Open_BCI_Framework



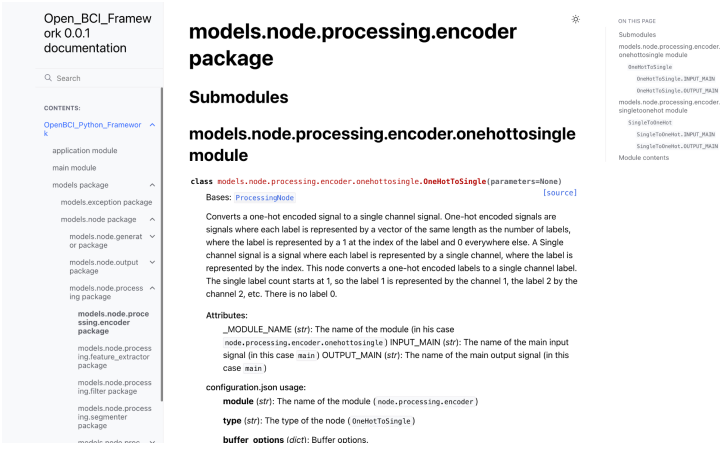
Fonte: Autoria própria

Figura 3 – Documentação do Open_BCI_Framework



Fonte: Autoria própria

Figura 4 – Documentação do Open_BCI_Framework



Fonte: Autoria própria

A partir da documentação e dos resultados obtidos, pode-se desenvolver em conjunto com um trabalho de mestrado uma ferramenta para processamento de sinais biomédicos. Dessa forma, trabalhos futuros focam-se na aplicação desta ferramenta para

diversos sinais e verificar o comportamento do framework no processamento de sinais de EEG, EMG e ECG.

A utilização do Open_BCI_Framework, como delineado na metodologia, destacou a versatilidade e o potencial deste framework para processamento de biosinais. A capacidade de personalização da sequência de processamento de dados por meio do arquivo "configuration.json" oferece uma solução adaptável para coleta e análise de biosinais, proporcionando uma ferramenta de grande valor.

A documentação desempenha um papel crucial em projetos complexos, como o Open_BCI_Framework. A utilização de ferramentas como docstrings e a biblioteca Sphinx do ecossistema Python permite uma documentação automatizada, melhorando a compreensão do framework e, consequentemente, sua utilidade e adesão. A capacidade de gerar documentação em vários formatos, como um site HTML estático, torna as informações acessíveis aos desenvolvedores e usuários, contribuindo para o sucesso do projeto e sua disponibilidade online.

Agradecimentos

O presente trabalho foi realizado com o apoio da Universidade Tecnológica Federal do Paraná/Brasil, e do orientador José Jair Alves Mendes Júnior.

Disponibilidade de código

Os códigos desenvolvidos utilizados estão disponíveis na plataforma GitHub https://github.com/arthurhauer/OpenBCI_Python_Framework.

Conflito de interesse

Os autores declaram não haver conflito de interesse.

REFERÊNCIAS

Desenvolvedores Sphinx. Boas-vindas. Disponível em: https://www.sphinx-doc.org/pt_BR/master/. Acesso em: 3 set. 2023.

HAUER, A. **OpenBCI_Python_Framework**. Disponível em: https://github.com/arthurhauer/OpenBCI_Python_Framework. Acesso em: 3 set. 2023.

MENDES JUNIOR, J.J.A., AGOSTINI, E., OKIDA, S., STEVAN JR, S.L. **Circuit Design for Surface Electromyography Data Acquisition IEEE Latin America Transactions**. 2015;13:3193-3200.

VLADIMIROV, L. **Writing docstrings**. Disponível em: https://sphinx-rtd-tutorial.readthedocs.io/en/latest/docstrings.html. Acesso em: 3 set. 2023.