

# wiki-analysis

## Sistema de categorização de artigos da Wikipédia

Hugo Folloni Guarilha  
Álgebra Linear Algorítmica - 2022.1  
Universidade Federal do Rio de Janeiro

### Apresentação

O wiki-analysis é um sistema que, com uso de álgebra linear e vetores, tenta determinar a categoria (ou tema) de um artigo da Wikipédia, por meio de comparações entre palavras presentes no texto. O algoritmo, ainda, é capaz de aprender com as categorizações passadas para ser mais eficiente futuramente. O projeto foi desenvolvido por Hugo Folloni como trabalho final da disciplina de Álgebra Linear Algorítmica no período de 2022.1, na Universidade Federal do Rio de Janeiro.

Esta plataforma é composta por:

- Front-end, feito em TypeScript utilizando a biblioteca React
- Back-end, separado em:
  - ❖ geração do vetor, com Python
  - ❖ API, com TypeScript, junto do framework Express para Node.

As categorias definidas são:

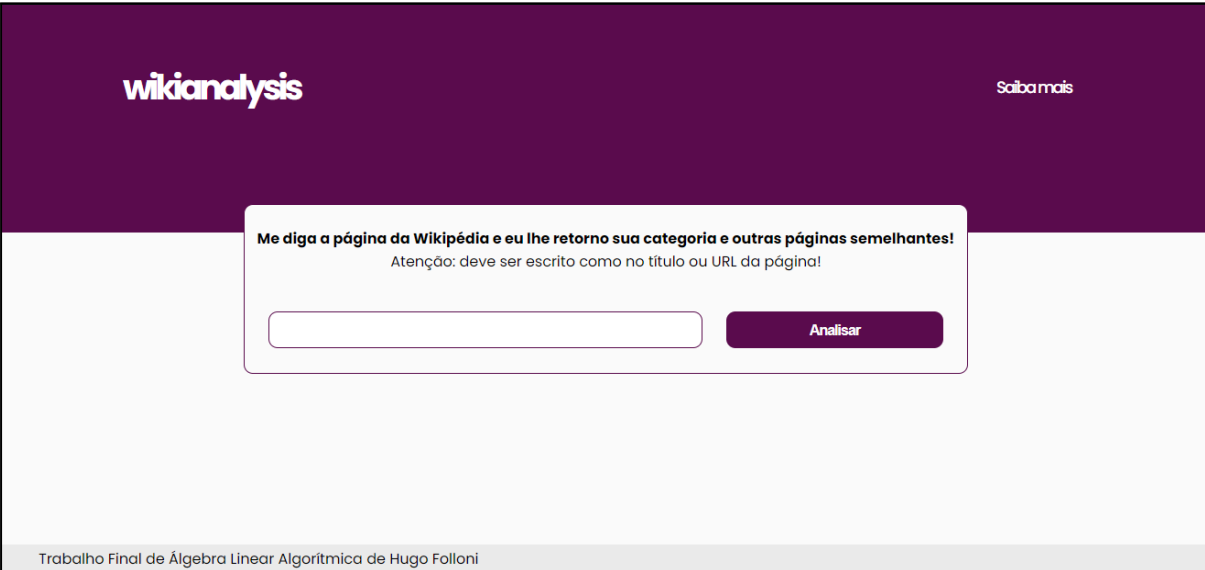
- Ciências
- Entretenimento
- Esportes
- Geografia
- História
- Música
- Sociedade
- Tecnologia

O código-fonte do projeto e sua versão live pode ser encontrada nos seguintes link abaixo:

- [Código-Fonte no Github](#)
- [Website](#)
- [API](#)
- [README](#)

## Front-end

Para a criação da parte da plataforma que roda no computador, foi utilizada a biblioteca React, junto da linguagem TypeScript. Ela consiste em uma página inicial, que possui instruções básicas para o correto uso, um link redirecionando para o Github, além de, claro, um input e um botão. O usuário deve escrever no input o título de sua página, ou como no fim de sua url (após “https://pt.wikipedia.org/wiki/...”). Ao clicar em “Analisar”, será iniciado o algoritmo de categorização.



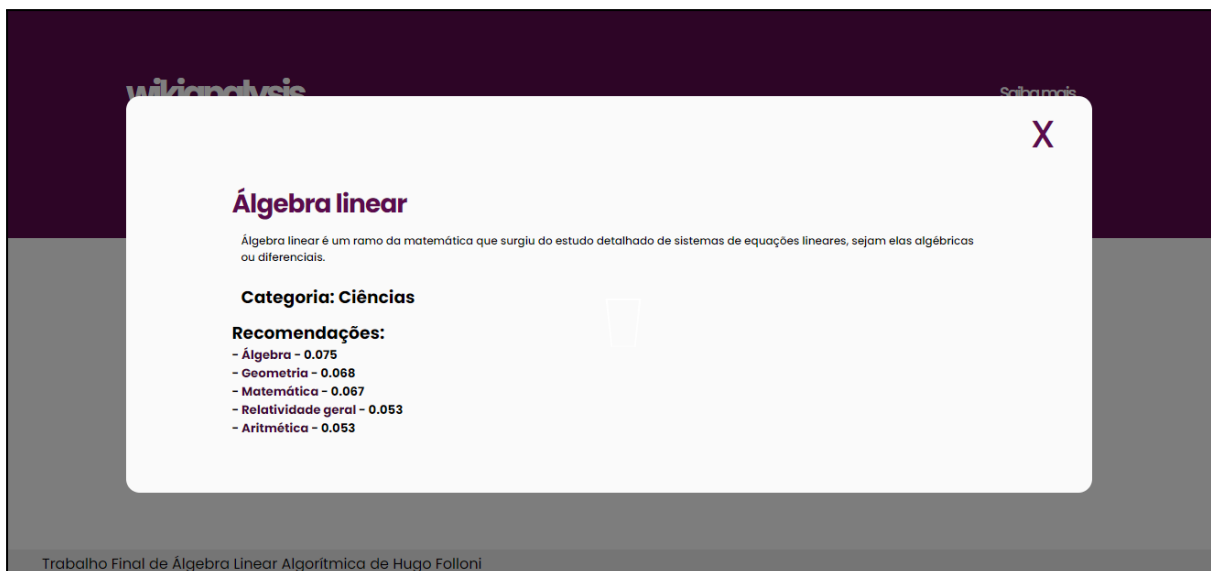
The screenshot shows the Wikianalysis web application. It has a dark purple header with the logo 'wikianalysis' on the left and a link 'Saiba mais' on the right. The main content area is white and features a central instruction box with a light purple border. The box contains the text: 'Me diga a página da Wikipédia e eu lhe retorno sua categoria e outras páginas semelhantes!' followed by a note: 'Atenção: deve ser escrito como no título ou URL da página!'. Below this text is a white input field and a dark purple button labeled 'Analisar'. At the bottom of the page, there is a light gray footer with the text: 'Trabalho Final de Álgebra Linear Algorítmica de Hugo Folloni'.

## Algoritmo de categorização

Este algoritmo tem como objetivo comparar a página passada pelo usuário com as presentes no banco de dados. Os passos de execução são:

- **Scrappy da página:**  
Neste momento, o algoritmo faz a leitura de todo o texto presente no artigo, buscando a frequência de cada palavra.
- **Comparação com as palavras que formam o vetor:**  
Existe um vetor em  $\mathbb{R}^{400}$  que, para cada dimensão, possui uma palavra considerada relevante para determinação da categoria. Neste momento, será feita a análise da frequência de cada uma dessas quatrocentas palavras no texto, gerando então um vetor.

- Comparação do vetor do artigo com os vetores no banco de dados:  
O algoritmo irá calcular o cosseno entre esse vetor gerado para a página e os vetores de todas as outras já existentes no banco de dados. Então, ele irá armazenar o cosseno, a url da página do banco de dados, e sua categoria em um objeto, e o adicionará em uma lista. A lista então é ordenada por cosseno, de forma decrescente.
- Categorização do artigo:  
Com base nos cinco maiores cossenos, o algoritmo irá classificar a página que o usuário requereu. Essa análise será da seguinte forma:
  - ❖ A categoria principal será obtida com base na categoria mais presente dentre os cinco maiores cossenos.
  - ❖ Caso haja outra categoria com mais de uma aparição, ela será considerada a categoria secundária.
- Resposta ao usuário:  
Agora, será gerada uma caixa de resposta para o usuário, mostrando a categoria que foi encontrada (podendo ser única, no caso de apenas a categoria principal, ou mostrando também a categoria secundária, caso haja). Além disso, também será disponibilizada uma lista das páginas mais semelhantes, com base nessa definição de vetor, tendo um link para ela, e o cosseno entre ela e este artigo. No console de desenvolvedor, também está presente a lista com todas as páginas, ordenadas por cosseno, na qual o usuário pode ver também a distância entre as páginas e outras próximas.



- Inserção no banco de dados  
Agora que a página já possui uma categoria, um vetor e uma url, podemos adicioná-la ao banco de dados. Sendo assim, para as próximas requisições, artigos também poderão ser comparados com ela, aperfeiçoando o algoritmo.

## Back-end

Para a criação do projeto, foi necessária a implementação de um back-end, para analisar os dados que irão gerar o vetor, e transmitir as informações do banco de dados para o front-end.

### Geração do vetor

Para a escolha das palavras que irão constituir o vetor, ou seja, ser determinantes de cada uma das dimensões, é executado um algoritmo em python, presente na pasta /server/src/setup do código-fonte. Para o correto funcionamento, o administrador do projeto (nesse caso, eu) deve determinar algumas páginas que serão utilizadas (recomendo no mínimo 50 para cada categoria).

O algoritmo lê as páginas presentes em cada arquivo /categories/pages/ que foram definidos pelo usuário (as categorias utilizadas são definidas no nome destes arquivos). O seu funcionamento é dividido em scrappy, análise das palavras e população do banco de dados.

- Scrappy (scraper.py e words\_frequency.py):

O algoritmo lê todas as linhas do arquivo de páginas, fazendo scrappy e adicionando a uma lista um objeto, que contém a palavra e sua quantidade de aparições. Ao fim, teremos uma lista de palavras e ocorrências para cada uma das categorias, que será escrita em um arquivo .txt na pasta /categories/words/.

- Análise de palavras (vector.py):

Aqui, os arquivos .txt serão lidos por ordem decrescente de ocorrências, sendo então escrito em um arquivo .txt no diretório /categories/vector/ quais são as palavras presentes em quantidade relevante apenas naquela categoria. Assim então, teremos uma lista de algumas palavras que só “aparecem” na categoria requerida.

Após isso, cabe ao administrador fazer uma seleção de quais palavras são realmente relevantes para o algoritmo. Como escolhi 50 palavras por categoria, levei em conta como abranger todos os conteúdos daquele tema, para o melhor funcionamento da plataforma.

- População do banco de dados (population.py):

Nessa fase do algoritmo, já com as palavras relevantes em mãos, será criado um banco de dados com todas as páginas passadas em /categories/pages/. A forma de seleção de categoria aqui será um pouco diferente daquela conhecida no front-end. Aqui, a categoria será determinada pela quantidade de aparições das palavras de cada tema. Como o vetor é dividido em 50 palavras por categoria, a categoria que tiver mais ocorrências será a que o vetor possuirá.

Ao fim desse processo, teremos um banco de dados populado com algumas páginas, que serão comparadas com as requisitadas pelo front-end. Além disso, será executado o arquivo reliability.py, que retornará qual a porcentagem de acerto dessa população (um erro é obtido quando a categoria do arquivo é diferente da categoria da população).

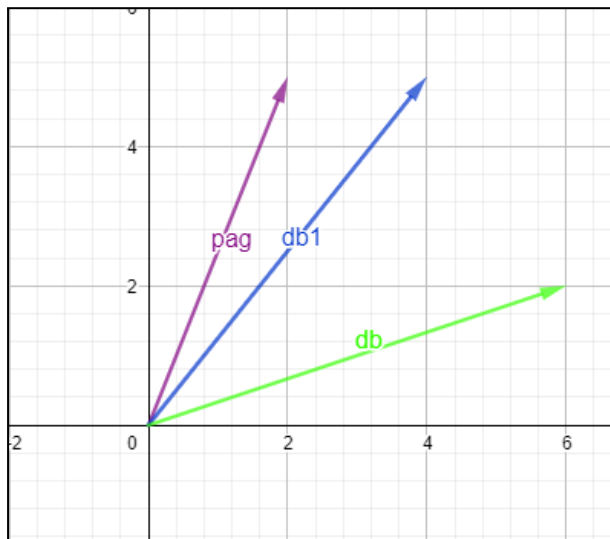
## Requisições do banco de dados

As requisições serão feitas por meio de uma API. Ela será construída com TypeScript e Express, com o objetivo de ler o nosso banco de dados SQLite, retornando um JSON na url <https://wiki-analysis-ala.herokuapp.com/api>. Aqui, teremos então o nome dos artigos, sua url e seu vetor, para posterior análise pelo front-end.

Um ponto interessante é que, por meio do endpoint /admin no nosso front-end, podemos excluir algumas páginas do banco de dados, passando seu id consumido na API, excluindo também, então, a análise daquela página (caso haja algum erro).

## Implementação e Algoritmo

O algoritmo construirá, para cada artigo da Wikipédia, um vetor, levando em conta as palavras formadoras de cada dimensão. Então, ele fará a comparação da página com base nos cossenos entre ela e os outros vetores.



Para calcular o cosseno entre vetores  $x$  e  $y$ ,

utilizamos a fórmula  $\frac{x^t y}{||x|| ||y||}$ , que

utiliza o produto interno entre dois vetores e sua norma (o seu tamanho).

O cosseno varia entre  $[-1, 1]$  e o quanto maior ele for, mais próximos dois vetores estão. Logo, o cosseno entre dois vetores pode indicar o quão semelhantes eles são. Essa foi a maneira escolhida para determinar a correspondência entre duas páginas. O nosso algoritmo irá comparar o vetor da nossa página com todos os outros vetores do nosso banco de dados.

Após isso, ao ordenar nossa lista de comparações por cosseno, teremos uma forma de ver quais os artigos mais próximos do requisitado. Analisando as categorias destes artigos, imagina-se que consigamos encontrar com sucesso a categoria para o nosso.

Inicialmente, as páginas com as quais a requisitada será comparada são aquelas enviadas pelo administrador para a população inicial do banco de dados. Porém, a cada requisição, o algoritmo salva o novo vetor no banco de dados, e, com isso, aprende e aumenta o número de comparações que ele pode fazer, o que aumenta a eficiência.

Porém, caso as palavras que definem o vetor de quatrocentas dimensões não sejam abrangentes o suficiente, ou seja, existam tópicos dentro de certa categoria que não são correspondidos por essas palavras, poderemos ter erros.

## Erros conhecidos

Infelizmente, nenhum software é isento de erros. No caso do wiki-analysis, foram definidos diversos conjuntos de páginas, tentando conseguir o menor erro possível. Por meio do script `reliability.py`, foi obtida uma porcentagem de 91.2% de acerto para as 575 páginas do conjunto final, definidas no diretório `/server/src/setup/categories/pages`.

Além disso, é conhecida uma falha na tentativa de determinar a categoria de páginas ligadas à biologia, especificamente a espécies animais e vegetais. Ao requisitar a categorização de um artigo de um animal ou planta, são obtidas as categorias “sociedade” ou “geografia”, na maioria dos casos. Isso ocorre por uma provável não-presença de palavras relevantes para esses tópicos dentre aquelas que definem o vetor.

Por fim, um erro intrínseco e indireto do projeto é que a escolha de páginas depende do administrador, por conta de não existir uma forma de conseguir artigos aleatórios dentro de um portal da Wikipédia. Isso introduz um viés na escolha das palavras, diminuindo, talvez, sua abrangência.

## Rodando localmente

Segue passo-a-passo de como rodar o projeto localmente, do zero.

```
# Clone o repositório
$ git clone https://github.com/hugofolloni/wiki-analysis

# Acesse a pasta
$ cd wiki-analysis

# Instale os pacotes necessários
$ yarn
$ cd server
$ yarn
$ cd ..

# Rode o servidor, para criar o banco de dados
$ yarn server

# Crie a pasta /pages dentro de /server/src/setup/categories para definir as páginas que o algoritmo utilizará para aprender

# Acesse a pasta de setup e rode o arquivo vector.py
## Este arquivo roda todas as funções necessárias para definição do vetor
$ cd server/src/setup
$ python vector.py

# Volte ao root e rode o front-end
$ cd ../../../../
$ yarn start
```

Além disso, o projeto pode ser acessado nos links nas páginas 1 ou 7. Nessa versão, ele já está com banco de dados populado e passou por algumas requisições, o que aumentou sua eficiência.

## Considerações Finais

Com esse trabalho, foi possível colocar em prática diversos conceitos aprendidos em sala de aula, como modelagem de problemas, cálculo da “diferença” entre vetores por cosseno e distância, entre outros. Gostaria de agradecer ao professor João Paixão pelos ensinamentos e conselhos.

Além disso, gostaria de agradecer aos amigos que me auxiliaram com esse projeto, me ajudando com a escolha das melhores páginas para cada categoria, com os testes e com os feedbacks. Também agradeço aos professores e orientadores, que me auxiliaram com artigos, dicas e ideias para a melhor realização deste trabalho.

E por fim, agradecer a Wikipédia, por disponibilizar uma plataforma gratuita para escrita e leitura de artigos em diversas categorias, objeto de estudo deste trabalho.

## Referências e Links

### Bibliotecas:

- <https://www.npmjs.com/package/vectorious>
- <https://beautiful-soup-4.readthedocs.io/en/latest/>
- <https://numpy.org/doc/stable/>

### Artigos:

- [https://en.wikipedia.org/wiki/Bag-of-words\\_model](https://en.wikipedia.org/wiki/Bag-of-words_model)
- [https://en.wikipedia.org/wiki/Document-term\\_matrix](https://en.wikipedia.org/wiki/Document-term_matrix)
- <https://machinelearningmastery.com/gentle-introduction-bag-words-model/>

### Trabalho:

- <https://github.com/hugofolloni/wiki-analysis>
- <https://wiki-analysis.netlify.app>
- <https://wiki-analysis-ala.herokuapp.com/api>
- <https://github.com/hugofolloni/wiki-analysis/blob/main/README.md>