Biscoitos Tia Gertrudes

github.com/ImLusca/Biscoitos-Tia-Gertrudes

Tia Gertrudes é uma senhora muito doce que possuí o dom da culinária. Porém, nos tempos modernos, seus quitutes vendidos em sua bela lojinha vêm perdendo espaço para as vendas do uaiComida, aplicativo de celular que permite que as pessoas comprem açúcares e gorduras saturadas sem sair de casa. E como essa nova geração é preguiçosa e acredita que um bom doce feito à mão com muito amor e carinho não vale uma caminhada até sua lojinha, tia Gertrudes vem vendendo cada vez menos.

Então, para superar esse problema, tia Gertrudes pede para que seu muito prestativo sobrinho Lucas crie um desses sites online para que sua lojinha também possa fazer parte dessa nova era. Mas tia Gertrudes não conseguia navegar no próprio site. E para consertar isso, pediu para que Lucas fizesse com que ela pudesse conversar com o computador para ele levá-la para as páginas corretas. E assim começamos essa epopeia:

Do que se trata?

Este projeto tem como permitir busca semântica para encontrar recursos dentro de um site.

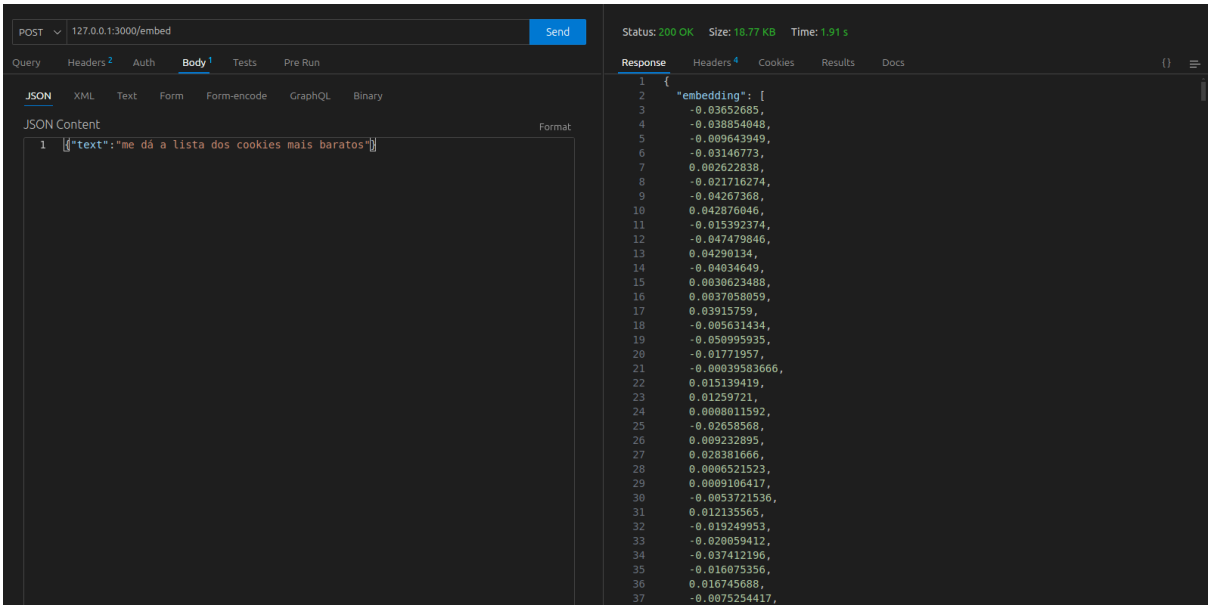
Como é feito?

Para isso, geramos um Json com todos os recursos disponíveis no site. Geramos palavras-chave que podem ser relacionadas a esses recursos e criamos embeddings delas. Depois disso, quando o usuário faz uma solicitação, pegamos o seu request, e comparamos o embedding da sua solicitação com os embeddings disponíveis, e assim, geramos o contexto que alimentará um promp para um modelo de geração de texto que irá construir a response correta baseada no contexto que lhe foi fornecido.

Mas oque são Embeddings?

Embeddings são uma técnica fundamental em processamento de linguagem natural (PLN) e aprendizado de máquina que transforma dados categóricos, como palavras, frases ou até mesmo imagens, em vetores numéricos de dimensão fixa. Esses vetores capturam informações semânticas e relacionamentos entre os dados, permitindo que algoritmos de machine learning trabalhem com eles de forma eficiente.

Exemplo:



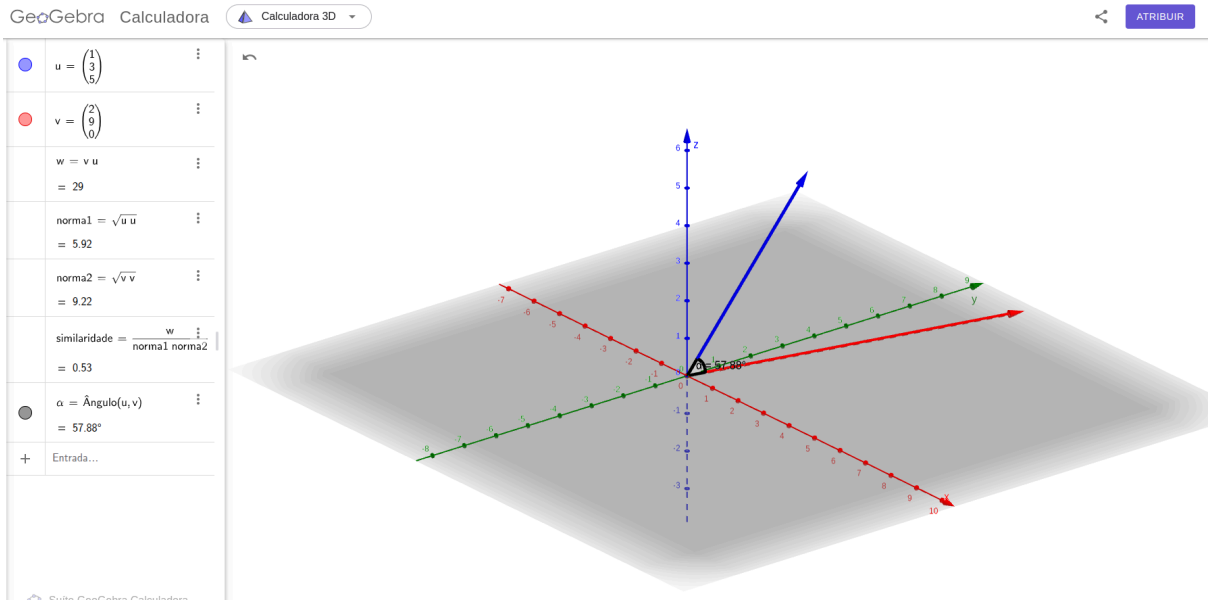
A frase "me dá a lista dos cookies mais baratos" após ser convertida, se torna uma lista de números entre -1 e 1 que podem ser representados como um vetor em um espaço multidimensional.

Se também gerarmos o embedding para a frase “me fala quais são os doces menos caros”, receberemos outra lista de números, com o mesmo número de dimensões, que no espaço muitidimensional, é muito próximo do primeiro vetor gerado.

Para calcular o quão similar um vetor em N dimensões é de outro vetor, podemos usar similaridade de cossenos, que é uma técnica que divide o produto vetorial dos dois vetores pela magnitude desses dois vetores, resultando em um número entre 0 e 1 que indica o quão próximos esses vetores são!!

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Exemplo em 3 dimensões:



Com tudo isso, conseguimos filtrar quais são os dados mais relevantes para entregarmos ao nosso modelo de geração de texto como contexto, reduzindo o tempo de processamento, custo de tokens e as chances de alucinações.

```
const generateResponse = async (
  userInput: string,
  references: Referencia[]
): Promise<Resposta> => {
  const prompt = `
    - Dadas as instruções abaixo, retorne exatamente e apenas um objeto JSON seguindo a interface:
      Resposta {
        page: string;
        sort: boolean;
        filter: boolean;
        sort_value?: string;
        filter_value?: string;
      }
    - Você receberá um input vindo do usuário e uma lista de referências extraídas de uma base maior.
    Caso não consiga encontrar uma resposta, retorne um JSON vazio.
    - O input do usuário é: ${userInput}
    - As referências são: ${JSON.stringify(references)}
  `;

  try {
    const completion = await openai.chat.completions.create({
      model: "gpt-3.5-turbo",
      messages: [
        {
          role: "user",
          content: prompt,
        },
      ],
    });
```

Depois disso, por segurança, validamos se a resposta do gerador de texto realmente é um json no formato que solicitamos, e se for, Retornamos para o usuário!

Pontos de melhoria:

Usar banco de vetores em vez de jsons,
Migrar para uma linguagem que permita paralelização real mais facilmente,
Aumentar a base de dados e palavras chaves,
monitorar logs de users para alimentar a base de dados.