Sistema de Perguntas e Respostas com TensorFlow.js

Descrição do Projeto

Este projeto é uma aplicação web que utiliza o modelo de Perguntas e Respostas (Question Answering) da biblioteca **TensorFlow.js**. A aplicação permite que o usuário insira um texto de contexto e, com base nele, faça perguntas em linguagem natural. O sistema então utiliza um modelo pré-treinado de QnA para identificar e retornar a melhor resposta possível dentro do texto.

Tecnologias Utilizadas

- **HTML5/CSS3** – Estrutura e estilo da interface.
- **JavaScript (Vanilla)** – Lógica da aplicação.
- **TensorFlow.js** – Biblioteca de machine learning no navegador.
- **@tensorflow-models/qna** – Modelo de perguntas e respostas baseado em BERT.

Estrutura dos Arquivos

```
/ProjetoWeb1

- index.html           # Interface principal
- style.css            # Estilização da interface
- app.js               # Lógica JavaScript e carregamento do modelo QnA
```

Como Funciona

1. O modelo `qna` é carregado diretamente no navegador com `@tensorflow-models/qna`.
2. O usuário insere um texto de contexto.
3. O usuário faz uma pergunta relacionada ao contexto.
4. O sistema retorna até duas respostas mais prováveis com o nível de confiança.
5. Caso nenhuma resposta seja encontrada, dicas são exibidas para melhorar os resultados.

Funcionalidades

- Carregamento dinâmico do modelo QnA.
- Campo para entrada de contexto e pergunta.
- Resposta baseada no texto com precisão estimada.
- Sugestões para perguntas e contextos mais eficazes.
- Tempo de processamento exibido ao usuário.
- Feedback quando a pergunta ou contexto for alterado.

Exemplo de Uso

Contexto:

> A água ferve a 100°C ao nível do mar, mas esse ponto de ebulição varia com a altitude. Em locais mais altos, a pressão atmosférica é menor, o que reduz o ponto de ebulição da água.

Pergunta: A que temperatura a água ferve ao nível do mar?
Resposta esperada: 100°C

Como Executar Localmente

1. Clone o repositório:

   ```
   git clone https://github.com/GuiRBeira/ProjetoWeb1.git
   ```

2. Abra o arquivo `index.html` no navegador.
3. Insira um contexto e uma pergunta.

Possíveis Problemas

- O modelo pode demorar alguns segundos para carregar, especialmente em conexões lentas.
- O modelo pode não entender perguntas vagas, ambíguas ou subjetivas.
- Perguntas devem estar claramente relacionadas ao contexto fornecido.

Explicação do Código HTML

Abaixo está a explicação do código `index.html`, que define a interface do usuário para o sistema de Perguntas e Respostas utilizando o TensorFlow.js:

- **¡!DOCTYPE html¿**
  Define o tipo de documento como HTML5.
- **¡html lang=”pt-BR»**
  Inicia o documento HTML e define o idioma como português do Brasil.
- **¡head¿**
  Contém metadados da página:
  - `<meta charset="UTF-8»` define a codificação de caracteres.
  - `<meta name="viewport"...>` torna o layout responsivo.
  - `<title>` define o título da aba do navegador.
  - `<link rel="stylesheet"...>` importa o arquivo CSS externo.
- **¡body¿**
  Contém todo o conteúdo visível da página, dividido em seções:
  - **Header** (`<header>`)
    Contém o título da aplicação e um menu de navegação com links fictícios: Início, Sobre e Contato.
  - **Main** (`<main>`)
    Área principal da página com a seção de análise de texto.
  - **Contexto** (`<textarea id="context»`)
    Campo onde o usuário insere o texto base em inglês.
  - **Pergunta** (`<input id="question»`)
    Campo para inserir a pergunta em inglês, relacionada ao contexto.
  - **Botões** (`<button>`)
    Botões para submeter a pergunta e alternar o tema da página.
  - **Resposta** (`<div id="answer»`)
    Área onde a resposta do modelo será exibida.
  - **Informações de Debug** (`<div id="debug-info»`)
    Mostra o tempo de execução e mensagens auxiliares.
  - **Rodapé** (`<footer>`)
    Contém informações de direitos autorais.
  - **Scripts TensorFlow.js e modelo QnA**
    Os dois `<script>` carregam a biblioteca `TensorFlow.js` e o modelo pré-treinado de Perguntas e Respostas.
  - **Script Principal** (`script.js`)
    Arquivo que contém toda a lógica da aplicação: carregamento do modelo, captura de entrada do usuário e exibição de resultados.

1 Explicação do Código CSS

1.1 Reset Geral

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

Remove margens e preenchimentos padrão de todos os elementos e define o modelo de caixa como `border-box` para facilitar o controle do layout.

1.2 Variáveis de Tema

As variáveis definidas no seletor `:root` permitem a personalização das cores do site:

- `--bg-color`: cor de fundo padrão
- `--text-color`: cor principal do texto
- `--primary-color`: cor principal da marca
- `--header-bg`, `--footer-bg`: cores de fundo para cabeçalho e rodapé
- Entre outras para mensagens, destaque, e áreas como debug

Além disso, há uma versão para **tema escuro**, aplicada quando a classe `.dark` está presente no `body`.

1.3 Estilização Base

Define o estilo geral da página, incluindo fonte, cor de fundo e texto, além de uma transição suave:

```
body {
    font-family: 'Segoe UI', Tahoma, sans-serif;
    background-color: var(--bg-color);
    color: var(--text-color);
    line-height: 1.6;
    padding: 20px;
    transition: all 0.3s ease;
}
```

1.4 Cabeçalho, Navegação e Rodapé

Estiliza os elementos `header` e `footer` com cor de fundo escura, texto branco e arredondamento de bordas. O menu de navegação usa flexbox para alinhar os itens horizontalmente.

1.5 Conteúdo Principal (main)

Centraliza o conteúdo com largura máxima de 900px e adiciona sombreamento, espaçamento interno e cantos arredondados.

1.6 Títulos

Os títulos são diferenciados por tamanho e cor:

- `h1`: título principal
- `h2`: destacado com a cor primária
- `h3`: subtítulos com margens

1.7 Inputs e Formulários

Campos de entrada como `textarea` e `input` são estilizados com preenchimento, bordas arredondadas e efeitos ao focar.

1.8 Botões

Botões recebem cor de fundo primária, sombra, bordas arredondadas e efeito de hover com leve translação vertical.

1.9 Seções Específicas

- `.answer-section`: área para exibir respostas
- `.debug-section`: área para depuração
- `.highlight`: destaque de texto
- `.error`: mensagens de erro com fundo vermelho claro
- `.tips`: sugestões ou dicas
- `.confidence`, `.processing-time`: informações auxiliares

1.10 Animação de Carregamento

Define um spinner com a classe `.loading` que gira continuamente com a animação `@keyframes spin`.

1.11 Responsividade

Usa `@media (max-width: 600px)` para adaptar o layout em telas pequenas:

- Menu de navegação se torna vertical
- Botões ocupam 100% da largura

Este CSS oferece uma estrutura moderna e personalizável para sites, incluindo suporte a temas claros/escuros, responsividade e acessibilidade visual.

2 Explicação do Código JavaScript (Sistema de Perguntas e Respostas com TensorFlow.js)

Este código implementa um sistema interativo de Perguntas e Respostas (QnA) utilizando a biblioteca `TensorFlow.js`, permitindo que usuários insiram um texto de contexto e façam perguntas com base nele. A aplicação processa a entrada e retorna respostas encontradas no texto.

2.1 Carregamento do Modelo

`async function loadQnAModel()`

Esta função é executada ao carregar a página. Ela:

- Exibe uma mensagem de carregamento ao usuário.
- Usa a função `qna.load()` para carregar o modelo pré-treinado da API `@tensorflow-models/qna`.
- Em caso de sucesso, atualiza a interface informando que o modelo está pronto.
- Em caso de erro, exibe uma mensagem de erro detalhada.

2.2 Processamento da Pergunta

`async function processQuestion()`

Esta função é chamada quando o usuário clica em "Perguntar"ou pressiona Enter. Ela:

- Valida se o contexto e a pergunta foram fornecidos.
- Exibe uma mensagem de "processando".
- Chama `qnaModel.findAnswers(question, context)`.
- Mede o tempo de execução com `performance.now()`.
- Exibe as respostas com nível de confiança e tempo de resposta.
- Trata erros que possam ocorrer durante a inferência.

2.3 Exibição das Respostas

`function displayAnswers(question, answers, processingTime)`

Responsável por exibir dinamicamente as respostas encontradas. Se nenhuma for encontrada, exibe dicas para melhorar os resultados. Caso haja respostas:

- Lista as respostas com o texto e a confiança (%).
- Exibe o tempo de processamento.
- Sugere boas práticas para melhor desempenho.

2.4 Eventos de Interface

`function setupEventListeners()`

Adiciona interações à interface:

- Clique no botão "Perguntar"chama `processQuestion()`.
- Pressionar Enter no campo de pergunta também aciona a função.
- Alterações nos campos de entrada atualizam mensagens na interface.

2.5 Inicialização do Sistema

`document.addEventListener('DOMContentLoaded', init);`

A função `init()` chama `setupEventListeners()` e `loadQnAModel()` após o DOM estar totalmente carregado.

2.6 Versão Alternativa do Submit

`document.getElementById('submit-button').addEventListener('click', ...)`

Implementa uma versão alternativa do botão de envio, que:

- Valida a presença do modelo e entradas.
- Chama `model.findAnswers()`.
- Mostra respostas formatadas e o tempo de execução.
- Exibe mensagens de erro, se necessário.

2.7 Tema Claro/Escuro

`function alternarTema()`

Permite alternar entre o tema claro e escuro. O estado do tema é salvo no `localStorage`, garantindo persistência entre sessões.

`window.addEventListener('DOMContentLoaded', ...)`

Recupera o tema salvo e aplica ao carregar a página.

2.8 Resumo

Este código fornece um sistema completo de QnA baseado em navegador com:

- Carregamento de modelo em tempo real com `TensorFlow.js`.
- Interface amigável com validações e dicas.
- Suporte a tema escuro.
- Medição de desempenho do modelo.

Dependência principal: `@tensorflow-models/qna` (importada externamente no HTML).

Considerações Finais

Este projeto demonstra como é possível usar modelos de NLP diretamente no navegador com **TensorFlow.js**, sem necessidade de backend ou servidores externos. Ele é ideal para fins educacionais ou prototipagem de sistemas interativos baseados em IA.