

ChatBook

Este documento descreve o desenvolvimento de um chatbot temático voltado para recomendação de livros, criado por Caio César e Raphael Imbelloni. O projeto utiliza a biblioteca “Flask” para criar uma aplicação web interativa e a biblioteca “NLTK” para processamento de linguagem natural. Abaixo estão os passos detalhados do desenvolvimento, organizados por etapas.

1. Configuração do Ambiente e Importação de Bibliotecas:

Descrição: Inicialmente, foi configurado o ambiente de desenvolvimento com as dependências necessárias.

Passos:

1.1- Instalação das bibliotecas necessárias:

- “Flask” para criar a aplicação web.
- “nltk” para processamento de linguagem natural.
- Comando executado: “pip install flask nltk”.

1.2- Importação das bibliotecas no código:

```
from flask import Flask, render_template, request, jsonify  
  
import nltk  
  
from nltk.tokenize import word_tokenize  
  
from nltk.corpus import stopwords  
  
from nltk.tokenize import RegexpTokenizer
```

1.3- Download dos recursos do NLTK:

- punkt para tokenização.

```
nltk.download("punkt")
```

- stopwords para filtragem de palavras irrelevantes.

```
nltk.download("stopwords")
```

2. Inicialização da Aplicação Flask:

Descrição: Configuração da aplicação Flask para servir as páginas web e processar requisições.

Passos:

2.1- Criação da instância da aplicação Flask:

```
app = Flask(__name__)
```

2.2- Definição da rota principal ('/') para renderizar a interface do usuário:

```
@app.route("/")
def index():
    return render_template("index.html")
```

2.3- Configuração do modo de depuração para desenvolvimento:

```
if __name__ == "__main__":
    app.run(debug=True)
```

3- Criação do Banco de Recomendações:

Descrição: Definição de uma estrutura de dados estática para armazenar as recomendações de livros por gênero.

Passos:

3.1- Criação de um dicionário `recomendacoes` com gêneros literários como chaves e listas de livros como valores:

```
recomendacoes = {
    fantasia: ["O Senhor dos Anéis", "Eragon", "As Crônicas de Nárnia"],
    romance: ["Orgulho e Preconceito", "Como Eu Era Antes de Você", "A Culpa é das Estrelas"],
    # ... outros gêneros
}
```

3.2- Inicialização de variáveis globais para rastrear o último gênero recomendado e as recomendações já fornecidas:

```
ultimo_genero_recomendado = None  
recomendacoes_dadas = set()
```

4. Implementação da Lógica de Processamento de Mensagens:

Descrição: Desenvolvimento da função principal para processar mensagens do usuário e gerar respostas com base em palavras-chave e contexto.

Passos:

4.1- Criação da rota `/chat` para receber mensagens via POST:

```
@app.route("/chat", methods=["POST"])  
def chat():  
    global ultimo_genero_recomendado  
    data = request.get_json()  
    if not data or 'message' not in data:  
        return jsonify({'response': 'Erro: nenhuma mensagem recebida.'}), 400  
    user_message = data['message']  
    resposta = gerar_resposta(user_message)  
    return jsonify({"response": resposta})
```

4.2- Desenvolvimento da função `gerar_resposta` para processar mensagens:

- Conversão da mensagem para minúsculas e tokenização com `RegexTokenizer`:

```
mensagem = mensagem.lower()  
tokenizer = RegexTokenizer(r'\w+')  
tokens = tokenizer.tokenize(mensagem)
```

- Filtragem de palavras irrelevantes usando `stopwords` do NLTK:

```
stop_words = set(stopwords.words('portuguese'))  
palavras_filtradas = [palavra for palavra in tokens if palavra not in stop_words]
```

4.3- Definição de um dicionário de categorias para mapear palavras-chave a gêneros:

```
categorias = {  
    "fantasia": ["ficção", "fantasia", "aventura"],  
    "romance": ["romance", "amor", "relacionamento"],  
    # ... outras categorias  
}
```

5. Lógica de Recomendação e Contexto:

Descrição: Implementação da lógica para recomendar livros com base nas palavras-chave e manter o contexto da conversa.

Passos:

5.1- Verificação de palavras que indicam que o usuário já conhece ou leu uma sugestão:

```
if any(p in palavras_filtradas for p in ["li", "lido", "já", "conheço", "sim", "com certeza",  
"claro", "outro", "outra", "exemplo", "sugestão", "sugestao", "ja"]):
```

```
    if ultimo_genero_recomendado and ultimo_genero_recomendado in  
    recomendacoes:
```

```
        sugestoos = [livro for livro in recomendacoes[ultimo_genero_recomendado] if  
        livro not in recomendacoes_dadas]
```

```
    if sugestoos:
```

```
        nova_sugestao = sugestoos[0]
```

```
        recomendacoes_dadas.add(nova_sugestao)
```

```
        return f"Entendi! Que tal então '{nova_sugestao}'?"
```

```
    else:
```

```
        return "Acho que já sugeri todos os que conheço desse gênero! Quer tentar  
        outro estilo de leitura?"
```

5.2- Resposta a feedback positivo do usuário (ex.: "gostei", "legal"):

```
if any(p in palavras_filtradas for p in ["gostei", "curti", "legal", "bom", "massa", "show",  
"adorei", "interessante"]):
```

```
    if ultimo_genero_recomendado and ultimo_genero_recomendado in  
    recomendacoes:
```

```

    sugestoes = [livro for livro in recomendacoes[ultimo_genero_recomendado] if
livro not in recomendacoes_dadas]

    if sugestoes:

        nova_sugestao = sugestoes[0]

        recomendacoes_dadas.add(nova_sugestao)

        return f"Fico feliz que tenha gostado! Caso queira ajuda em algo mais é só
falar!"

```

5.3- Identificação de gêneros mencionados pelo usuário e recomendação de livros:

```

for genero, palavras in categorias.items():

    if any(p in palavras_filtradas for p in palavras):

        ultimo_genero_recomendado = genero

        recomendacoes_dadas = set()

        livro = recomendacoes[genero][0]

        recomendacoes_dadas.add(livro)

        return f"Recomendo '{livro}'! Já ouviu falar?"

```

5.4- Respostas padrão para interações genéricas (ex.: "oi", "recomenda"):

```

if any(p in palavras_filtradas for p in ["recomenda", "indica", "sugere"]):

    return "Claro! Me diga que tipo de livro você gosta: romance, mistério, aventura,
etc."

# ... outras condições

```

6. Testes e Depuração:

Descrição: Validação do funcionamento do chatbot em diferentes cenários de entrada.

Passos:

6.1- Execução da aplicação no modo de depuração (`app.run(debug=True)`).

6.2- Testes manuais com entradas como:

- Solicitações de gêneros ("Quero um livro de fantasia").

- Feedback positivo ("Gostei da sugestão").
- Pedidos de novas sugestões ("Já li esse, tem outro?").
- Mensagens genéricas ("Oi, quais livros você recomenda?").

6.3- Correção de bugs relacionados a:

- Tokenização incorreta de palavras com acentos.
- Respostas fora de contexto quando palavras-chave não eram detectadas.

7. Considerações Finais:

O chatbot foi projetado para ser simples, mas funcional, com uma interface web que permite interação fluida com os usuários. Ele utiliza técnicas básicas de NLP para identificar intenções e manter o contexto da conversa, recomendando livros de forma personalizada. A colaboração entre **Caio César** e **Raphael Imbelloni** foi essencial para dividir as tarefas de configuração, lógica de backend e criação de respostas amigáveis.

7.1- Possíveis Melhorias:

- Adicionar mais livros e gêneros ao banco de recomendações.
- Implementar aprendizado de preferências do usuário ao longo da conversa.

Esta documentação reflete o processo completo de desenvolvimento do ChatBook.