**Trabalho Final 2023**

**Introdução à Inteligência Artificial na Área da Saúde**

[Emilie Ga Eun Kim](mailto:emilie.kim@ufcspa.edu.br) e [Mariana Luisa Goncalves Silva](mailto:mariana.goncalves@ufcspa.edu.br)

<https://colab.research.google.com/drive/1eUgnK8B604bpet2AZ8ZIN5_-Z6PJGdwe?authuser=0#scrollTo=G1scXZ13PsZA>

para fazer nossa base de conhecimento:

artigo: <https://www.cell.com/action/showPdf?pii=S0960-9822%2819%2930204-0>

site da usp: <https://sites.icmc.usp.br/andre/research/neural/>

SEM AIML -> modelo pré-treinado

**crie um chatbot especialista em redes neurais em inteligência artificial**

**Para criar um chatbot especializado em redes neurais artificiais com uma base de conhecimento, podemos usar uma combinação de modelos pré-treinados e uma base de dados de perguntas e respostas específicas. Neste exemplo, usarei o modelo BERT para responder a perguntas e uma base de conhecimento simulada.**

**Certifique-se de ter a biblioteca transformers instalada:**

terminal bash:

pip install transformers

**Aqui está um exemplo básico:**

terminal python:

from transformers import pipeline

import random

# Base de conhecimento simulada (você pode expandir isso com informações mais detalhadas)

base\_de\_conhecimento = {

"O que é uma rede neural?": "Uma rede neural é um modelo computacional inspirado no sistema nervoso biológico, composta por camadas de neurônios artificiais interconectados.",

"Como funcionam as redes neurais?": "Redes neurais processam informações passando-as por camadas de neurônios, onde cada conexão entre neurônios possui um peso ajustável.",

"Quais são os tipos de redes neurais?": "Existem vários tipos, incluindo redes feedforward, redes recorrentes (RNNs) e redes convolucionais (CNNs), cada uma adequada para tarefas específicas.",

# Adicione mais perguntas e respostas conforme necessário

}

# Modelo BERT para question-answering

qa\_pipeline = pipeline('question-answering', model='bert-base-uncased', tokenizer='bert-base-uncased')

# Diálogo com o chatbot

print("Chatbot Especialista em Redes Neurais Artificiais:")

print("Olá! Como posso ajudar você hoje? (Digite 'sair' para encerrar)")

while True:

user\_input = input("Você: ")

if user\_input.lower() == 'sair':

print("Chatbot: Até logo!")

break

# Verificar se a pergunta está na base de conhecimento

if user\_input in base\_de\_conhecimento:

response = base\_de\_conhecimento[user\_input]

else:

# Se não estiver na base de conhecimento, usar o modelo BERT para responder

response = qa\_pipeline(context=" ".join(base\_de\_conhecimento.values()), question=user\_input)['answer']

print(f"Chatbot: {response}")

**Neste exemplo, o chatbot verifica se a pergunta do usuário está na base de conhecimento simulada. Se estiver, ele fornece uma resposta direta. Se não estiver, utiliza o modelo BERT para gerar uma resposta com base na base de conhecimento geral sobre redes neurais.**

**Lembre-se de que esta é uma abordagem simplificada e você pode expandir a base de conhecimento e melhorar o modelo de resposta conforme necessário para atender às suas necessidades específicas.**

**TEMOS QUE FAZER NA MÃO UMA PRÓPRIA COM LIVROS E ARTIGOS**

**crie uma base de conhecimento para o tema**

Certamente, criar uma base de conhecimento envolve fornecer informações sobre o tema de redes neurais. Aqui está uma base de conhecimento simples sobre redes neurais que você pode expandir conforme necessário:

1. \*\*Definição de Redes Neurais Artificiais:\*\*

As redes neurais artificiais são modelos computacionais inspirados na estrutura e função do sistema nervoso humano. Elas consistem em camadas de neurônios interconectados, conhecidos como unidades ou neurônios artificiais.

2. \*\*Arquitetura Básica:\*\*

- \*\*Neurônios:\*\*

Neurônios artificiais são as unidades básicas de uma rede neural. Cada neurônio recebe entradas, aplica pesos a essas entradas e passa o resultado por uma função de ativação.

- \*\*Camadas:\*\*

As redes neurais são organizadas em camadas, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída.

- \*\*Conexões:\*\*

As conexões entre os neurônios têm pesos que são ajustados durante o treinamento da rede.

3. \*\*Treinamento de Redes Neurais:\*\*

- \*\*Algoritmos de Otimização:\*\*

O treinamento de redes neurais envolve otimizar os pesos das conexões para minimizar uma função de custo. Algoritmos como o Gradiente Descendente são comumente usados.

- \*\*Função de Ativação:\*\*

Funções como a sigmoide, tangente hiperbólica e ReLU são usadas para introduzir não-linearidades nas redes.

4. \*\*Tipos de Redes Neurais:\*\*

- \*\*Redes Neurais Feedforward:\*\*

Informação flui em uma direção, da entrada para a saída, sem ciclos.

- \*\*Redes Neurais Recorrentes (RNNs):\*\*

Permitem a retroalimentação, permitindo a consideração de contextos temporais.

- \*\*Redes Neurais Convolucionais (CNNs):\*\*

Especializadas em processar dados com estrutura de grade, como imagens.

5. \*\*Aplicações Práticas:\*\*

- \*\*Aprendizado Supervisionado:\*\*

Classificação e regressão.

- \*\*Aprendizado Não Supervisionado:\*\*

Agrupamento e redução de dimensionalidade.

- \*\*Processamento de Linguagem Natural (PLN):\*\*

Compreensão de linguagem, tradução automática, resumo de texto, etc.

6. \*\*Desafios e Considerações:\*\*

- \*\*Overfitting:\*\*

Ajustar muito bem aos dados de treinamento, mas não generalizar bem.

- \*\*Vanishing Gradient Problem:\*\*

Dificuldade de atualizar pesos em camadas mais profundas.

- \*\*Interpretabilidade:\*\*

Redes neurais complexas podem ser difíceis de interpretar.

7. \*\*Ferramentas e Frameworks:\*\*

- \*\*TensorFlow e PyTorch:\*\*

Frameworks populares para construir e treinar redes neurais.

- \*\*Keras:\*\*

Interface de alto nível que roda sobre TensorFlow e Theano.

Esta base de conhecimento fornece uma visão geral das redes neurais, cobrindo desde conceitos fundamentais até aplicações práticas e considerações importantes. Lembre-se de que este é apenas um ponto de partida, e você pode expandi-lo com informações mais detalhadas, exemplos, referências e atualizações conforme a tecnologia evolui.

COM AIML

AIML (Artificial Intelligence Markup Language) é uma linguagem de marcação específica para a criação de chatbots ou agentes de conversação. Ela foi desenvolvida com o objetivo de criar sistemas de inteligência artificial que pudessem simular conversas humanas de maneira mais natural.

A AIML é baseada em XML e segue uma abordagem baseada em regras para definir padrões de entrada e as respostas correspondentes. Cada regra AIML consiste em um padrão (a expressão que o usuário pode fornecer) e uma resposta associada (a resposta que o sistema deve dar).

A biblioteca AIML em Python (geralmente chamada de PyAIML ou simplemente AIML) permite implementar chatbots usando esta linguagem de marcação. Essa biblioteca oferece funcionalidades para criar, treinar e interagir com chatbots AIML.

crie um chatbot especialista em redes neurais artificiais com uma base de conhecimento do assunto aiml

**Para criar um chatbot especializado em redes neurais artificiais com uma base de conhecimento usando AIML (Artificial Intelligence Markup Language), você pode usar a biblioteca AIML em Python. AIML é uma linguagem de marcação que facilita a criação de chatbots e agentes de conversação.**

**Certifique-se de ter a biblioteca AIML instalada:**

terminal bash:

!pip install python-aiml

**importar drive**

from google.colab import drive

drive.mount('/content/drive')

#get\_ipython().run\_line\_magic('cd', '"/content/drive/MyDrive/4o semestre/inteligência artificial/trabalho final IA/"')

%cd "/content/drive/MyDrive/4o semestre/inteligência artificial/trabalho final IA/"

**Aqui está um exemplo básico de um chatbot AIML para redes neurais artificiais:**

terminal python:

import aiml

# Criar o kernel AIML

kernel = aiml.Kernel()

# Caminho do arquivo AIML

base\_conhecimento\_aiml\_file = '/content/drive/MyDrive/4o semestre/inteligência artificial/trabalho final IA/base.aiml'

# Carregar os arquivos AIML

kernel.learn(base\_conhecimento\_aiml\_file)

# Diálogo com o chatbot

print("Chatbot Especialista em Redes Neurais Artificiais:")

print("Olá! Como posso ajudar você hoje? (Digite 'sair' para encerrar)")

# Esse trecho de código representa a estrutura básica de um ciclo de interação com o usuário. O chatbot solicita uma entrada, processa a entrada usando o kernel AIML para gerar uma resposta e exibe essa resposta. O loop continua até que o usuário escolha encerrar digitando 'sair'.

# loop infinito, que continuará executando até ser explicitamente interrompido pelo comando break. Isso permite que o chatbot interaja repetidamente com o usuário.

while True:

# Solicita ao usuário que insira uma mensagem. O valor inserido pelo usuário é armazenado na variável user\_input.

user\_input = input("Você: ")

# Verifica se a entrada do usuário (convertida para minúsculas usando lower()) é igual a 'sair'. Se for, o chatbot exibe uma mensagem de despedida, imprime "Até logo!" e interrompe o loop usando break, encerrando a interação.

if user\_input.lower() == 'sair':

print("Chatbot: Até logo!")

break

# Usar o kernel AIML para responder

# Usa o kernel AIML para gerar uma resposta com base na entrada do usuário. A função respond do kernel AIML busca padrões na base de conhecimento AIML que correspondam à entrada do usuário e retorna a resposta correspondente.

response = kernel.respond(user\_input)

# Exibe a resposta gerada pelo chatbot. A resposta é formatada como "Chatbot: [resposta]" e é apresentada ao usuário.

print(f"Chatbot: {response}")

**Neste exemplo, o chatbot AIML verifica se a pergunta do usuário está na base de conhecimento AIML simulada. Se estiver, ele fornece uma resposta direta. Se não estiver, ele utiliza o kernel AIML para gerar uma resposta com base em regras AIML predefinidas.**

**Lembre-se de que AIML é uma linguagem mais antiga e pode não ter o mesmo poder ou flexibilidade que abordagens baseadas em modelos mais recentes. Se você estiver interessado em construir um chatbot mais avançado, pode ser benéfico considerar outras abordagens, como o uso de modelos de linguagem pré-treinados.**

EXPLICAÇÃO DA FUNÇÃO KERNEL

No AIML (Artificial Intelligence Markup Language), o "kernel" refere-se à instância central que interpreta as instruções AIML e gerencia a interação do chatbot. O kernel AIML é uma espécie de "cérebro" do chatbot AIML, responsável por interpretar as entradas do usuário, procurar correspondências nos padrões AIML e gerar as respostas correspondentes.

Vamos explorar o uso do kernel AIML em mais detalhes:

1. \*\*Inicialização do Kernel:\*\*

No seu código Python, você inicializa o kernel AIML usando a classe `aiml.Kernel()`:

```python

import aiml

# Criar o kernel AIML

kernel = aiml.Kernel()

```

2. \*\*Aprendizado (Learn):\*\*

O kernel AIML precisa aprender com uma base de conhecimento AIML antes de poder interagir com os usuários. Isso é feito usando o método `learn()` para carregar os padrões AIML de um arquivo específico:

```python

# Caminho do arquivo AIML

base\_conhecimento\_aiml\_file = 'base.aiml'

# Carregar os arquivos AIML

kernel.learn(base\_conhecimento\_aiml\_file)

```

O aprendizado é uma etapa importante para que o chatbot tenha uma base de conhecimento inicial.

3. \*\*Interagir com o Usuário:\*\*

Após o aprendizado, o kernel AIML está pronto para interagir com os usuários. O método `respond()` é usado para obter respostas com base nas entradas do usuário:

```python

while True:

user\_input = input("Você: ")

if user\_input.lower() == 'sair':

print("Chatbot: Até logo!")

break

# Usar o kernel AIML para responder

response = kernel.respond(user\_input)

print(f"Chatbot: {response}")

```

O loop permite uma interação contínua com o usuário, onde o chatbot responde com base nas entradas fornecidas.

4. \*\*Aprimoramento Contínuo:\*\*

À medida que você estende a base de conhecimento AIML adicionando mais padrões e respostas ao arquivo AIML, o kernel AIML se torna mais especializado e capaz de lidar com uma variedade maior de perguntas.

O kernel AIML é uma parte central e vital para a funcionalidade do chatbot AIML, fornecendo a capacidade de aprender com a base de conhecimento e responder a consultas dos usuários com base nos padrões AIML definidos.