Trabalho Final de PLN
Aluna: Rafaela Melo Ferreira
Disciplina: Tópicos Especiais em Recuperação da Informação (NLP)
Prof.º: Altigran da Silva / André Carvalho

1. Introdução

Esse trabalho consistiu em desenvolver uma LLM (*Large Language Model*) capaz de responder perguntas sobre a legislação acadêmica de Graduação da Universidade Federal do Amazonas (UFAM). Para tanto, foi necessário realizar as seguintes etapas: i) download da base de dados e extração dos textos dos PDFs; ii) pré-processamento; iii) geração da base de dados sintética; iv) treinamento do modelo de linguagem com LoRA/QLoRA; v) implementação de RAG (*Retrieval-Augmented Generation*). Essas etapas estão descritas nas seções a seguir.

2. Pré-processamento

O primeiro passo foi realizar o download das legislações, em seguida, usando o Tesseract, foi feita a extração dos textos dos PDFs. O código apresentado na Figura 1 foi utilizado para extrair os textos. Os textos foram salvos em arquivos TXT para facilitar o processo.

```python
def pdf_to_text(pdf_path, output_folder):
    # Convertendo PDF para imagens
    images = convert_from_path(pdf_path)

    text = ""
    for i, image in enumerate(images):
        # Convertendo cada imagem em texto
        page_text = pytesseract.image_to_string(image, lang='por')  # Para português
        text += page_text + "\n\n"

    return text

def process_multiple_pdfs(input_folder, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    for filename in os.listdir(input_folder):
        if filename.endswith(".pdf"):
            pdf_path = os.path.join(input_folder, filename)
            print(f"Processando {pdf_path}...")
            extracted_text = pdf_to_text(pdf_path, output_folder)

            # Salvar o texto extraído em um arquivo de texto
            text_output_path = os.path.join(output_folder, f'{os.path.splitext(filename)[0]}_texto_extraido.txt')
            with open(text_output_path, 'w', encoding='utf-8') as f:
                f.write(extracted_text)
```

Figura 1. Código para extração de textos dos PDFs.

3. Geração da base

Para gerar a base, primeiramente usou-se uma abordagem para gerar a partir de modelos como GPT-2 e GPT-3, porém os resultados não foram satisfatórios. Também tentou-se trabalhar com a API do GPT para geração da base sintética, porém também não se obteve sucesso.

Por conta disso, a base foi gerada com auxílio do GPT-4, ao fazer o upload dos arquivos de texto extraídos dos PDFs e solicitar que fossem criadas perguntas e respostas referentes às legislações. Foram geradas 372 perguntas/respostas com base nas legislações. Por limitações do GPT-4, não foi possível gerar 1000 exemplos.

4. Treinamento do modelo de linguagem com LoRA/QLoRA

```
# Carregar o modelo GPT-2
model = GPT2LMHeadModel.from_pretrained("pierreguillou/gpt2-small-portuguese")

# Configurar LoRA
lora_config = LoraConfig(
    task_type=TaskType.CAUSAL_LM,
    r=8,
    lora_alpha=32,
    lora_dropout=0.1,
    target_modules=["attn.c_attn", "attn.c_proj"]
)

model = get_peft_model(model, lora_config)

# Configurar os parâmetros de treinamento
training_args = TrainingArguments(
    output_dir="/content/drive/MyDrive/tp3/results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    num_train_epochs=20,
    weight_decay=0.01,
)

# Criar o trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)

# Treinar o modelo
trainer.train()
```

Figura 2. Código de treinamento do modelo.

Conforme apresentado na Figura 2, para adaptar o modelo GPT-2 para a tarefa de respostas a perguntas sobre legislação acadêmica, foi utilizada a técnica LoRA, que permite adaptar modelos grandes de forma eficiente, adicionando uma quantidade mínima de novos parâmetros. Abaixo estão alguns detalhes sobre a implementação:

- **Carregamento do Modelo GPT-2:** foi utilizado um modelo GPT-2 pré-treinado em português.
- **Configuração do LoRA:** foi configurado o LoRA para adaptar as camadas de projeção de queries e valores (q_proj e v_proj) do modelo GPT-2. Essas camadas de projeção fazem parte da arquitetura de mecanismos de atenção, que é uma técnica que permite que o modelo aprenda a focar em diferentes

partes da entrada para cada posição da saída, o que é importante para capturar dependências contextuais em sequências de dados.
- Argumentos de Treinamento: foram definidos os parâmetros de treinamento, incluindo a taxa de aprendizado, tamanho do batch e número de épocas.
- Inicialização do Trainer: foi utilizada a classe Trainer da biblioteca transformers para gerenciar o processo de treinamento.
- Treinamento: o treinamento do modelo foi realizado utilizando os dados pré-processados.

Com relação às métricas de treinamento, training loss e validation loss começaram bem altas, mas terminaram em 0.77 e 0.73, respectivamente. Também foi adicionada uma métrica de perplexidade após o treinamento, mas como é possível ver no notebook, apresentou um valor muito alto.

5. Implementação de RAG

O sistema RAG combina recuperação de documentos relevantes com geração de texto para fornecer respostas às perguntas. A implementação do sistema RAG envolveu as seguintes etapas:

- **Configuração do Pipeline de QA**: foi utilizado um pipeline de geração de texto com o modelo treinado.
- **Carregamento de Documentos**: foram carregados os documentos de texto que serviram como base de conhecimento.
- **Geração de Embeddings**: foram geradas embeddings para os documentos utilizando um modelo de embeddings.
- **Configuração do Índice FAISS**: foi utilizada a biblioteca FAISS para criar um índice de similaridade que permite buscar documentos relevantes com base nos embeddings.

No código é possível visualizar alguns exemplos de perguntas e respostas.

6. Desafios

- **Conjunto de dados:** o fato de serem apenas 372 exemplos pode ter influenciado no resultado final.
- **Configuração dos hiperparâmetros:** os hiperparâmetros escolhidos podem não ter sido os mais adequados para este problema específico.
- **Limitação do Google Colab:** o treinamento foi realizado com auxílio da GPU no Google Colab, porém em alguns momentos a memória estourou e foi necessário até usar mais de uma conta do Google para finalizar o trabalho.